Politechnika Wrocławska

FIELD OF SCIENCE: Engineering and Technology

DISCIPLINE OF SCIENCE: Information and Communication Technology

DOCTORAL DISSERTATION

Normalizing flow models for modeling uncertainty in machine learning tasks

mgr inż. Patryk Wielopolski

Supervisor:
dr hab. inż. Maciej Zięba, prof. uczelni

Keywords: machine learning, normalizing flows, uncertainty modeling

WROCŁAW 2024

Personal Acknowledgments

First and foremost, I would like to extend my deepest gratitude to my Ph.D. supervisor, *Maciej Zięba*. This Ph.D. journey would not have been possible without your support. One of my key criteria for pursuing a doctorate was having an exceptional supervisor, and you are exactly that. I am incredibly grateful for your patience and understanding during all my highs and lows, from moments of doubt and pauses to re-energized comebacks. Together, we made this possible—perhaps not in the two years we initially aimed for, but three is still an accomplishment I am very proud of.

I also want to express thanks to my collaborators and colleagues at *genwro.ai*. In particular, I owe immense gratitude to *Oleksii*, with whom I have had the pleasure of closely collaborating over the past year and a half. I believe we have both gained invaluable insights from each other, and I eagerly look forward to observing your progress through your doctorate journey. A special thank you also goes to *Michał* for sharing the ups and downs of our Ph.D. adventures and providing mutual support along the way.

I am grateful for the support of my family and friends. To my beloved parents: *Mamo, Tato – dziękuję za wasze nieustające wsparcie, celebrowanie moich sukcesów, wysiłek w odkrywaniu, kiedy i gdzie odbędzie się kolejna konferencja, a także przypominanie mi, jak ważny jest odpoczynek.* To my dear friends, *Dominika*, *Bartek*, and *Aleks*, thank you for simply being there. I know I can always rely on you, ask for help, and have the deep conversations we all enjoy so much. Your presence has been a source of immense emotional support, and I am deeply grateful for it.

A special thank you to *Martyna* for your continuous support, for making me laugh countless times, and for making sure I rest (or sometimes work) in the most wonderful places—whether it's Berlin, Canary Islands, London, Venice, Porto, Azores, Lisbon, Mallorca, Iceland, Edinburgh, New York or any other place in the world. You've shown me the joy in balancing life and work, empathized during tough publishing moments, and fueled my curiosity. And, of course, for all the questions that have kept me thinking.

Lastly, I want to acknowledge *myself*—for the inner work, self-reflection, and the systems I've built to support my personal growth. This process is ongoing, but I feel I've made great strides during this Ph.D. journey.

Table of Contents

Abstract

In recent years, machine learning models have achieved remarkable results across a wide range of tasks, including classification, regression, and data generation, spanning various modalities such as tabular data, text, images, and point clouds. However, many of these applications have focused primarily on deterministic predictions, often overlooking the importance of modeling uncertainty in those predictions. This dissertation aims to address this limitation by utilizing Normalizing Flow models to effectively model uncertainty across a spectrum of machine learning tasks.

This thesis consists of seven papers and is divided into three parts. The first part, covered in Publications [I-III], focuses on applying normalizing flows to discriminative tasks. Publication [I] introduces TreeFlow, a novel approach that enhances tree-based parametric models with more expressive probability distributions for regression tasks. Publication [II] presents NodeFlow, an end-to-end probabilistic regression framework for tabular data that improves upon TreeFlow's two-stage learning process. Publication [III] demonstrates the versatility of these methods by adapting them to personalized natural language processing, yielding significant improvements in emotion recognition and hate speech detection.

The second part of the thesis, encompassing Publications [IV-VI], explores the application of normalizing flows to generative tasks. Publication [IV] introduces the Flow Plugin Network (FPN), an innovative architecture that integrates normalizing flows with pre-trained generative models. Publications [V, VI] present PluGeN, a sophisticated approach for multi-label conditional generation that leverages pre-trained models, demonstrating its potential in generating images and 3D point clouds with attribute combinations not seen during training.

The third part, covered in Publication [VII], investigates the intersection of discriminative and generative tasks within the domain of Explainable AI (XAI). It introduces a novel method for generating Probabilistically Plausible Counterfactual Explanations using Normalizing Flows (PPCEF), enhancing the plausibility and interpretability of AI explanations.

All the publications listed here were presented at top- and high-tier conferences or published in high-impact journals. The research demonstrates the versatility of normalizing flows across various machine learning paradigms, contributing to the development of more accurate, controllable, and interpretable AI systems. This work opens up several promising avenues for future research, including scaling these methods to larger datasets, integrating them with other emerging paradigms, and extending their application to new domains.

Streszczenie

W ostatnich latach modele uczenia maszynowego osiągnęły niezwykłe wyniki w szerokim zakresie zadań, w tym klasyfikacji, regresji i generowania danych, obejmujących różne modalności, takie jak dane tabelaryczne, tekst, obrazy i chmury punktów. Jednak wiele z tych zastosowań skupiało się głównie na przewidywaniach deterministycznych, często pomijając znaczenie modelowania niepewności w tych przewidywaniach. Niniejsza rozprawa ma na celu rozwiązanie tego ograniczenia poprzez wykorzystanie modeli normalizacji przepływu do efektywnego modelowania niepewności w szerokim spektrum zadań uczenia maszynowego.

Ta rozprawa składa się z siedmiu artykułów i jest podzielona na trzy części. Pierwsza część, omówiona w publikacjach [I-III], koncentruje się na stosowaniu modeli normalizacji przepływu do zadań dyskryminacyjnych. Publikacja [I] wprowadza TreeFlow, nowatorskie podejście, które wzmacnia parametryczne modele oparte na drzewach o bardziej ekspresyjne rozkłady prawdopodobieństwa dla zadań regresji. Publikacja [II] przedstawia NodeFlow, kompleksowe podejście do regresji probabilistycznej dla danych tabelarycznych, które ulepszają dwuetapowy proces trenowania TreeFlow. Publikacja [III] demonstruje wszechstronność tych metod poprzez dostosowanie ich do spersonalizowanego przetwarzania języka naturalnego, co daje znaczące usprawnienia w rozpoznawaniu emocji i wykrywaniu mowy nienawiści.

Druga część pracy, obejmująca Publikacje [IV-VI], bada zastosowanie normalizacji przepływów do zadań generatywnych. Publikacja [IV] wprowadza Flow Plugin Network (FPN), innowacyjną architekturę, która integruje modele normalizacji przepływu z wstępnie wyszkolonymi modelami generatywnymi. Publikacje [V, VI] przedstawiają PluGeN, wyrafinowane podejście do wieloetykietowej generacji warunkowej, które wykorzystuje wstępnie wyszkolone modele, demonstrując jego potencjał w generowaniu obrazów i chmur punktów 3D z kombinacjami atrybutów niewidocznymi podczas treningu.

Trzecia część, omówiona w Publikacji [VII], bada przecięcie się zadań dyskryminacyjnych i generatywnych w domenie Explainable AI (XAI). Wprowadza nową metodę generowania prawdopodobnych, kontrfaktycznych wyjaśnień przy użyciu normalizacji przepływów (PPCEF), zwiększając wiarygodność i interpretowalność wyjaśnień AI.

Wszystkie wspomniane publikacje zostały zaprezentowane na prestiżowych konferencjach lub ukazały się w renomowanych czasopismach naukowych. Badania te pokazują, jak wszechstronne zastosowanie mają modele normalizacji przepływu w różnych dziedzinach uczenia maszynowego. Przyczyniają się one do tworzenia systemów sztucznej inteligencji, które są nie tylko dokładniejsze, ale także bardziej kontrolowane i łatwiejsze do zrozumienia. Ta praca otwiera wiele obiecujących kierunków dla przyszłych badań. Wśród nich można wymienić zastosowanie tych metod do większych zbiorów danych, połączenie ich z nowymi koncepcjami w dziedzinie AI, a także rozszerzenie ich wykorzystania na nowe obszary.

List of Publications

[I] **P. Wielopolski**, M. Zięba
"TreeFlow: Going Beyond Tree-based Parametric Probabilistic Regression"
ECAI 2023 : 26th European Conference on Artificial Intelligence : September 30–October 4, 2023, Kraków, Poland : proceedings / eds. Kobi Gal [et al.]. Amsterdam : IOS Press, cop. 2023. s. 2631-2638
2023; CORE A; 140 MEiN points

[II] **P. Wielopolski**, O. Furman, M. Zięba
"NodeFlow: Towards End-to-end Flexible Probabilistic Regression on Tabular Data"
Entropy. 2024. vol. 26, nr 7, art. 593, s. 1-16
2024; IF: 2.1; 100 MEiN points

[III] P. Miłkowski, K. Karanowski, **P. Wielopolski**, J. Kocoń, P. Kazienko, M. Zięba
"Modeling Uncertainty in Personalized Emotion Prediction with Normalizing Flows"
23nd IEEE International Conference on Data Mining Workshops (ICDMW), 1–4 December 2023 Shanghai, China : proceedings / eds. Jihe Wang [et al.]. Piscataway, NJ : Institute of Electrical and Electronics Engineers, cop. 2023. s. 757-766
2023; CORE A; 200 MEiN points

[IV] **P. Wielopolski**, M. Koperski, M. Zięba
"Flow Plugin Network for conditional generation"
Intelligent Information and Database Systems 15th Asian Conference, ACIIDS 2023 : Phuket, Thailand, July 24-26, 2023 : Proceedings, Pt. 2 / ed. Ngoc Thanh Nguyen [et al.]. Singapore : Springer, cop. 2023. s. 221-232
2023; CORE B; 70 MEiN points

[V] M. Wołczyk, M. Proszewska, Ł. Maziarka, M. Zięba, **P. Wielopolski**, R. Kurczab, M. Śmieja
"PluGeN: Multi-Label Conditional Generation from Pre-trained Models"
Proceedings of the 36th AAAI Conference on Artificial Intelligence : February 22 - March 1, 2022 : virtual conferece / eds. Gabriel Pedroza [et al.]. Palo Alto, USA : AAAI Press, cop. 2022. s. 8647-8656
2022; CORE A*; 200 MEiN points

[VI] M. Proszewska, M. Wołczyk, M. Zięba, **P. Wielopolski**, Ł. Maziarka, M. Śmieja
"Multi-Label Conditional Generation From Pre-Trained Models"
IEEE Transactions on Pattern Analysis and Machine Intelligence. 2024. vol. 46, nr 9, s. 6185-6198
2024; IF: 20.8; 200 MEiN points

[VII] **P. Wielopolski**, O. Furman, J. Stefanowski, M. Zięba
"Probabilistically Plausible Counterfactual Explanations with Normalizing Flows"
Accepted to ECAI 2024 : 27th European Conference on Artificial Intelligence : October 19–October 24, 2024, Santiago de Compostela, Spain
2024; CORE A; 140 MEiN points

CHAPTER 1

Introduction

Machine Learning Machine Learning (ML) has revolutionized various areas of our lives by providing powerful tools for extracting patterns and insights from complex data [1, 2]. Over the past decade, ML techniques have demonstrated remarkable success in diverse domains, including decision-making systems [3], natural language processing [4], and computer vision [5]. As these methods continue to evolve, there is an increasing recognition of the importance of not only making predictions but also quantifying the uncertainty associated with those predictions [6, 7].

Uncertainty in Machine Learning In machine learning, uncertainty refers to the inherent variability and lack of confidence in model predictions [8]. It encompasses various sources of ambiguity, including noise in the data, limitations of the model, and the stochastic nature of the processes being modeled. Uncertainty modeling aims to quantify and represent this variability in a principled manner [6]. Rather than relying solely on point estimates, probabilistic approaches to machine learning explicitly capture uncertainty by modeling the distribution of possible outcomes [9, 10]. These methods provide a framework for reasoning about the reliability and limitations of ML models, forming the foundation for more robust and interpretable machine learning systems.

Why is Modeling Uncertainty Important? Understanding and quantifying uncertainty is essential for robust decision-making in real-world applications, where the consequences of errors can be significant [7]. Effective uncertainty quantification enhances model interpretability, improves decision-making processes, and provides valuable insights into the reliability of predictions across various domains [8]. In both discriminative and generative tasks, uncertainty modeling enables more informed decisions and realistic outputs [6, 11]. Moreover, in the context of Explainable AI (XAI), it contributes to the development of more transparent and trustworthy systems [12]. Proper uncertainty modeling thus not only improves the performance and reliability of machine learning models but also enhances their applicability in critical real-world scenarios.

Current State in Modeling Uncertainty The landscape of uncertainty modeling in machine learning is characterized by a range of approaches, from simple point-based estimates to more sophisticated probabilistic methods. Commonly used techniques include frequentist methods like bootstrap [13], Bayesian approaches using variational inference or MCMC [14], ensemble techniques [15], and Gaussian processes [16]. These methods often rely on simplifying assumptions, typically employing Gaussian-based or other parameterized distributions to make uncertainty quantification tractable [10]. Despite these simplifications, such approaches have found wide application across various domains. They

have been successfully applied to regression problems on tabular data [17, 18], computer vision tasks such as object detection and semantic segmentation [7], and in generative modeling using variational autoencoders [19]. The prevalence of these methods in practical applications can be attributed to their relative simplicity, computational efficiency, and well-established theoretical foundations. However, as the complexity of real-world data and tasks increases, the limitations of these conventional approaches become more apparent.

Limitations in Modeling Uncertainty Current approaches to modeling uncertainty face several key limitations. Many traditional methods, including tree-based models like NGBoost and CatBoost [20, 21], rely on parametric distributions, limiting their ability to capture complex, real-world data distributions. Deep learning methods offer more flexibility but often struggle with computational efficiency and scalability [15, 14]. Most techniques face challenges with high-dimensional samples and distinguishing between different sources of uncertainty [8]. The interpretability of uncertainty estimates, particularly in complex models, remains a significant challenge [22, 12]. To address these limitations, recent research has explored more flexible approaches to uncertainty modeling.

Normalizing Flows In recent years, Normalizing Flows [23] has emerged as a promising solution to address the limitations in uncertainty modeling. These models offer a unique approach to distribution estimation, distinguishing themselves from other deep generative techniques such as Variational Autoencoders (VAEs) [19], Generative Adversarial Networks (GANs) [24], and Diffusion Models [25]. At their core, normalizing flows leverage the principle of change of variables to transform a simple base distribution into a complex target distribution [23].

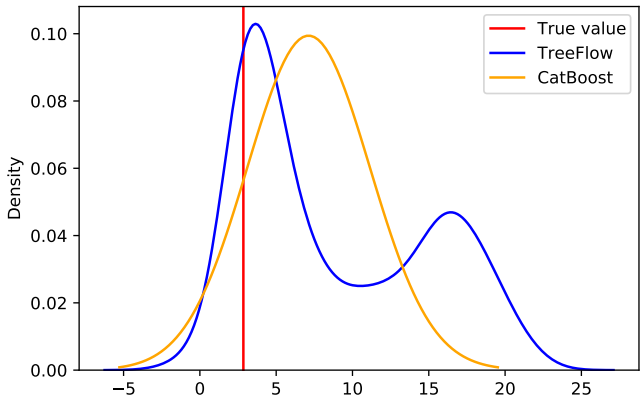
Formally, a normalizing flow is defined as a composition of invertible and differentiable transformations, $f = f_K \circ f_{K-1} \circ ... \circ f_1$, applied to a random variable z with a known probability density $p_Z(z)$. The resulting probability density $p_X(x)$ of the transformed variable $x = f(z)$ can be computed exactly through the change of variables formula:

$$p_X(x) = p_Z(f^{-1}(x)) \left| \det \frac{\partial f^{-1}(x)}{\partial x} \right|, \tag{1.1}$$

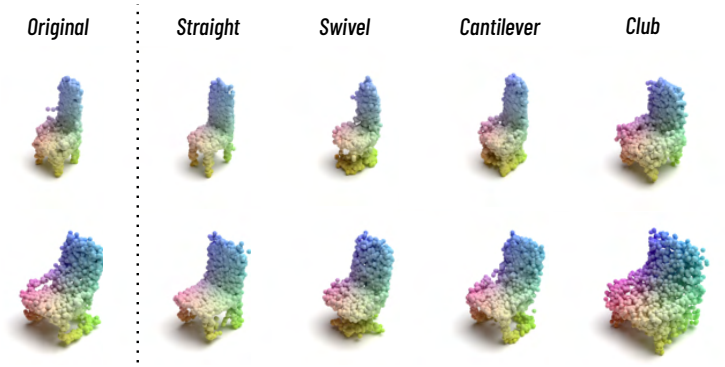
where $p_Z(z)$ typically denotes a simple base distribution such as a standard normal distribution.

Key Properties and Advantages of Normalizing Flows This formulation endows normalizing flows with several key properties that make them particularly suitable for addressing the challenges in uncertainty modeling [26]. First and foremost, normalizing flows provides exact computations of probability densities, enabling precise likelihood evaluation. This feature, coupled with the invertible nature of the transformations, allows for both efficient sampling and exact density estimation, overcoming the limitations of approximate methods often used in traditional uncertainty quantification.

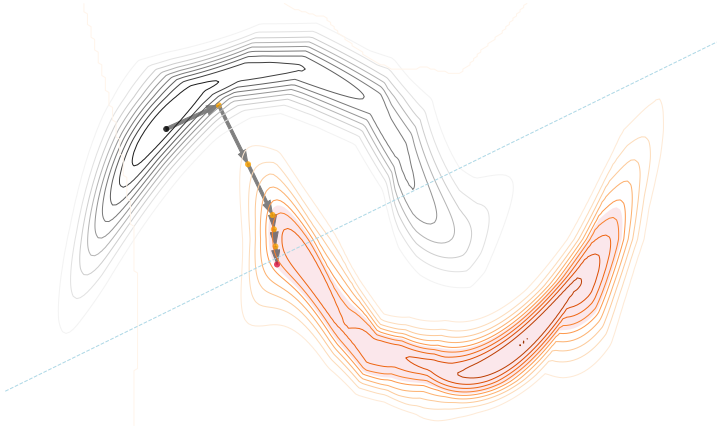
The careful design of these transformations empowers normalizing flows to model complex, high-dimensional probability distributions, addressing the challenge of capturing intricate data distributions that many parametric models struggle with [27, 28, 29, 30]. Moreover, the bidirectional mapping between the base and target distributions facilitates both inference and generation tasks, making normalizing flows a versatile tool across various machine learning paradigms.



(a) Normalizing Flows for discriminative tasks.



(b) Normalizing Flows for generative tasks.



(c) Normalizing Flows for combined tasks.

Figure 1.1: Application of Normalizing Flows for uncertainty modeling across diverse machine learning tasks. (a) Discriminative tasks: Enhancing probabilistic regression for tabular and text data. The innovations led to a more reliable uncertainty modeling for both domains. (b) Generative tasks: Advancing computer vision applications. These improvements enabled better control of generative model outputs, more effective object manipulation, and novel generation capabilities previously unattainable. (c) Combined discriminative and generative tasks: Leveraging generative capabilities of Normalizing Flows to explain discriminative models through plausible counterfactual explanations.

This flexibility and power make normalizing flows particularly well-suited for tackling the limitations in current uncertainty modeling approaches. They offer the potential to model complex, multimodal distributions without the need for simplifying assumptions, to provide scalable solutions for high-dimensional data, and to enable more nuanced uncertainty quantification in both discriminative and generative tasks. These unique capabilities of normalizing flows open up new avenues for research and applications across various domains of machine learning, setting the stage for the core investigations of this thesis.

Motivation The core of this thesis is driven by the aforementioned unique properties of normalizing flows, particularly their ability to perform exact density estimation and facilitate efficient sampling. The primary research question guiding this work is: *"How can the properties of normalizing flows be leveraged to enhance uncertainty modeling across a range of machine learning tasks?"* As illustrated in Figure 1.1, this work explores the application of normalizing flows across a spectrum of machine learning tasks, focusing on both discriminative (Figure 1.1a) and generative contexts (Figure 1.1b). Additionally, it explores innovative hybrid approaches that merge these paradigms (Figure 1.1c) within the domain of Explainable AI.

Contributions This thesis makes several key contributions to the field through seven papers exploring various aspects of normalizing flows for uncertainty modeling. These papers are organized into three chapters, each dedicated to a different machine learning task. The research presented in this thesis has been published in highly regarded venues, including machine learning conferences rated A* to B by CORE and journals with Impact Factors ranging up to 20.8. Of these seven papers, I am the first author of four, demonstrating significant personal contributions to the field. The impact of this work is evident in its accumulation of 1050 MEiN points and 12 citations as of October 2024, according to Google Scholar.

This thesis explores the application of normalizing flows in both discriminative and generative models, as well as in innovative hybrid approaches, demonstrating their versatility and potential across various machine learning paradigms. The following paragraphs provide an overview of the key contributions made in each chapter, outlining how normalizing flows are applied to specific machine learning tasks.

Chapter 2 explores the application of normalizing flows to discriminative tasks, focusing on probabilistic regression. It introduces TreeFlow, a novel approach that enhances tree-based parametric models with more expressive probability distributions [I]. TreeFlow integrates normalizing flows with tree-based models, significantly improving uncertainty estimates in regression tasks (Figure 1.1a) and achieving notable advancements in univariate regression benchmarks. The chapter then presents NodeFlow, an end-to-end probabilistic regression framework for tabular data [II]. NodeFlow improves upon TreeFlow's two-stage training process, offering a unified approach for modeling complex distributions and achieving state-of-the-art results in multivariate probabilistic regression. Finally, the chapter demonstrates the versatility of these methods by adapting them to personalized natural language processing [III]. This application yields significant improvements in emotion recognition and hate speech detection, highlighting the broad effectiveness of the proposed methodologies.

In Chapter 3, the focus shifts to the realm of generative tasks, specifically exploring how normalizing flows can enhance conditional generation and manipulation of complex

data types like images and 3D point clouds. The chapter's first major contribution is the Flow Plugin Network (FPN) [IV], an innovative architecture that seamlessly integrates normalizing flows with pre-trained generative models such as Variational Autoencoders. FPN's plug-in network design enables refined control over generated outputs, facilitating conditional generation, object attribute manipulation (Figure 1.1b), and even classification tasks. The chapter then introduces PluGeN, a sophisticated approach for multi-label conditional generation that leverages pre-trained models [V, VI]. By extending the FPN framework to modern architectures like Generative Adversarial Networks, PluGeN tackles the challenge of generating images and 3D point clouds with attribute combinations not seen during training. Through advanced attribute disentanglement, PluGeN demonstrates its potential to address complex generative challenges, pushing the boundaries of what's possible in conditional generation and data manipulation.

Chapter 4 delves into the intersection of discriminative and generative tasks, demonstrating the versatility of normalizing flows in addressing challenges within the domain of Explainable AI. While XAI encompasses various approaches to making machine learning models interpretable, this chapter focuses specifically on counterfactual explanations for discriminative models. In this context, I introduce a novel method for generating Probabilistically Plausible Counterfactual Explanations using normalizing flows (PPCEF) [VII]. This approach pushes the boundaries of current methods by ensuring that the generated counterfactuals not only achieve the desired class change but also align with the underlying data distribution modeled by the normalizing flow model, thereby enhancing their plausibility (Figure 1.1c).

Chapter 5 discusses the contributions to the research community. It highlights my efforts in founding a research group, reviewing for conferences, conducting workshops, mentoring students, giving talks, and collaborating at local, national, and international levels during my Ph.D. journey. Chapter 6 summarizes the thesis, highlighting key contributions to advancing machine learning through versatile applications of normalizing flows in discriminative and generative modeling and Explainable AI. Finally, Appendix A provides the complete texts of the related publications.

CHAPTER 2

Part I: Normalizing Flows for Discriminative Tasks

2.1 Research Scope

Discriminative tasks in machine learning encompass a wide range of problems where the primary objective is to predict specific outcomes or labels based on input features [10]. These tasks, which include classification and regression, form the backbone of numerous real-world applications, from medical diagnosis to financial forecasting [31, 32]. As the complexity and stakes of these applications continue to grow, there is an increasing need for models that not only provide accurate predictions but also offer reliable estimates of uncertainty [7].

The importance of uncertainty modeling in discriminative tasks cannot be overstated. In high-stakes domains such as healthcare or financial forecasting mentioned previously, understanding the confidence level of a model's predictions is crucial for making informed decisions and mitigating risks. Traditional approaches to uncertainty estimation in discriminative tasks have often relied on simplistic assumptions, such as Gaussian distributions for regression tasks or softmax probabilities for classification [10]. While these methods provide a basic measure of uncertainty, they often fail to capture the true complexity of real-world data distributions.

Existing approaches to uncertainty modeling in discriminative tasks include Bayesian Neural Networks [33], Monte Carlo Dropout [6], and ensemble methods [15]. While these techniques have shown promise, they often struggle with computational efficiency, scalability, or the ability to model complex, multi-modal distributions. Moreover, many of these methods provide only approximate uncertainty estimates, which may not be sufficient for critical applications requiring precise probabilistic outputs.

Normalizing flows offers a promising avenue for addressing these limitations in discriminative tasks. By leveraging their ability to model complex probability distributions through a series of invertible transformations, normalizing flows can potentially provide more accurate and flexible uncertainty estimates. This chapter explores novel applications of normalizing flows to various discriminative tasks, focusing on probabilistic regression and emotion prediction in natural language processing.

The research presented in this chapter aims to bridge the gap between the expressive power of normalizing flows and the practical needs of discriminative tasks. By introducing innovative architectures that combine normalizing flows with traditional machine learning models, I seek to enhance the accuracy and reliability of uncertainty estimates across a range of applications. This work not only advances the state-of-the-art in probabilistic modeling for discriminative tasks but also paves the way for more robust and interpretable machine learning systems in critical domains.

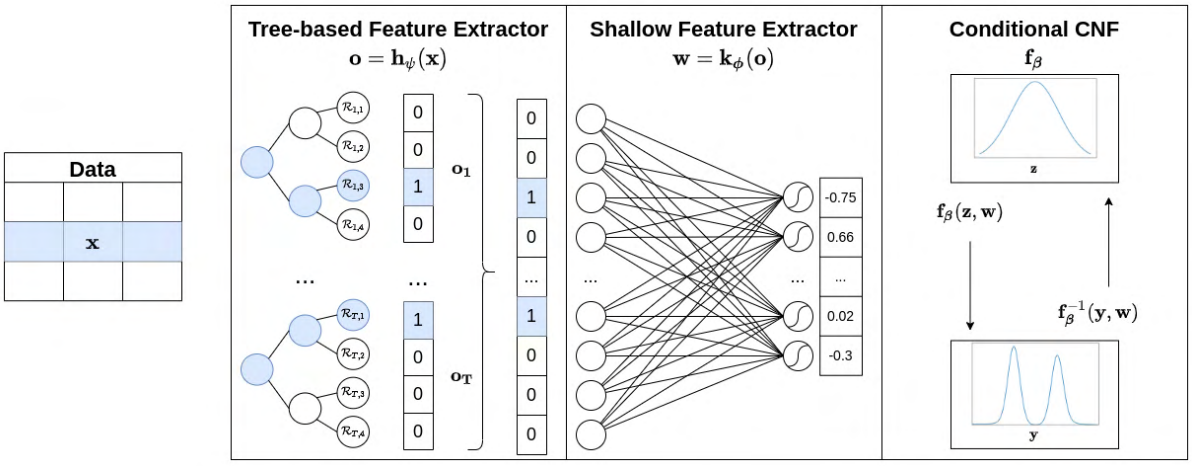


Figure 2.1: TreeFlow architecture. The proposed model consists of three components: Tree-based Feature Extractor, Shallow Feature Extractor, and Conditional CNF module.

2.2 TreeFlow: Going Beyond Tree-based Parametric Probabilistic Regression

This section covers results from Publication [I], which explores improving uncertainty modeling in probabilistic regression tasks by combining tree-based models with normalizing flows.

Motivation Tree-based models have long been a popular choice for regression tasks due to their interpretability and ability to handle complex feature interactions [17, 18, 34]. However, probabilistic extensions of the traditional tree-based methods [20, 21, 35] often rely on simplistic probabilistic assumptions, limiting their ability to capture complex uncertainty patterns in real-world data. The motivation behind TreeFlow is to address this limitation by combining the strengths of tree-based models with the flexibility of normalizing flows, thereby enabling more expressive and accurate uncertainty modeling in regression tasks.

Contents TreeFlow introduces a novel framework that extends traditional tree-based parametric probabilistic regression models to accommodate more flexible probability distributions. The core idea is to use a tree-based model as a feature extractor and combine it with a conditional variant of normalizing flow. This approach allows for modeling complex distributions for regression outputs, going beyond the typical Gaussian assumptions.

The architecture of TreeFlow (presented in Figure 2.1) consists of three main components: a Tree-based Feature Extractor, a Shallow Feature Extractor, and a conditional Continuous Normalizing Flow (CNF) module. The Tree-based Feature Extractor captures complex relationships in tabular data through a tree-like structure, while the Shallow Feature Extractor maps the high-dimensional binary output to a low-dimensional representation. The CNF module then utilizes this representation as a conditioning factor to model flexible probability distributions for the regression outputs.

The training process of TreeFlow employs a two-stage approach. First, the Tree-based

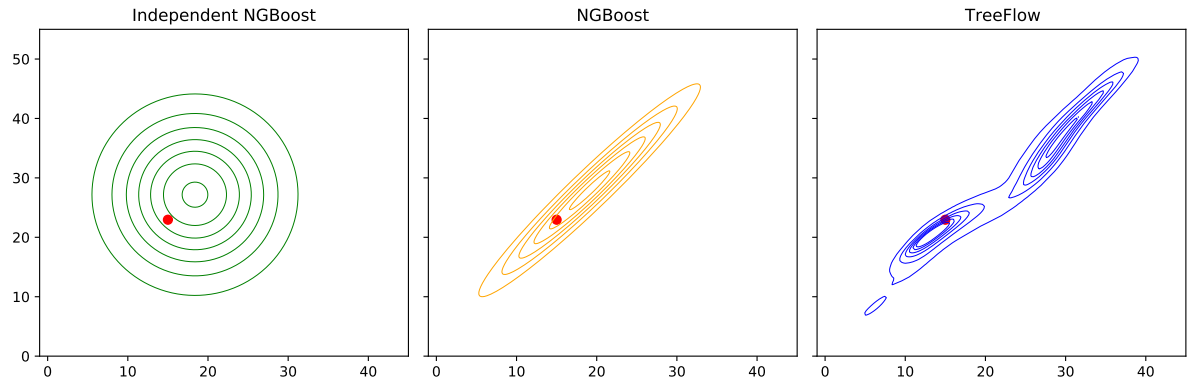


Figure 2.2: The estimated probability density functions for TreeFlow and NGBoost, a benchmark method, demonstrate that the proposed approach effectively models multi-modal distributions, surpassing the limitations of a fixed unimodal Gaussian distribution.

Feature Extractor is trained using a surrogate criterion specific to the type of tree-based architecture. Then, the remaining components are trained end-to-end by optimizing the negative log-likelihood of the conditional probability distribution modeled by the CNF.

Extensive experiments were conducted on various regression benchmarks, including datasets with varying volumes, feature characteristics, and target dimensionality. The results demonstrate TreeFlow's superior performance compared to traditional tree-based regression baselines, particularly on datasets with multi-modal target distributions as presented in the paper and in Figure 2.2.

Summary TreeFlow represents a significant advancement in probabilistic regression modeling by combining the strengths of tree-based models and normalizing flows. This novel approach enables more accurate and flexible uncertainty estimation, particularly for complex, non-gaussian data distributions. The experimental results validate the effectiveness of TreeFlow across a range of regression tasks, highlighting its potential to improve decision-making in various real-world applications.

Contribution I was primarily responsible for conducting the literature review, designing the method, carrying out experiments, analyzing data, and writing the manuscript. My supervisor, Maciej Zięba, contributed by providing guidance on experimental design and analysis, assisting with result interpretation, and offering feedback during the writing and revision stages. The work was presented at CORE A conference ECAI 2023 in October 2023 and has received 4 citations as of October 2024.

2.3 NodeFlow: Towards End-to-end Flexible Probabilistic Regression on Tabular Data

This section covers results from Publication [II], which overcomes the limitation of Publication [I] by introducing an end-to-end training approach for integrating tree-based methods with normalizing flows.

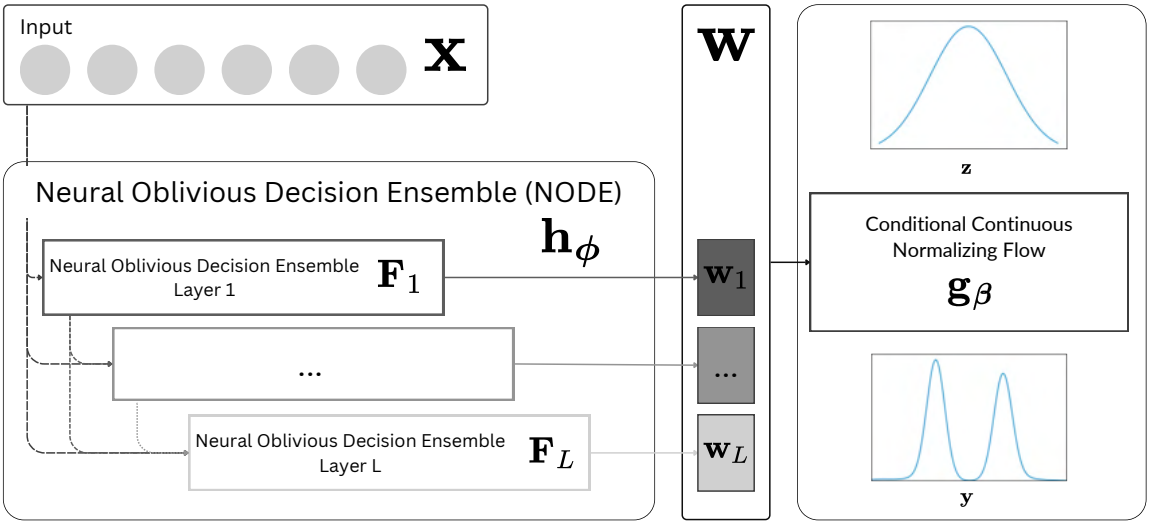


Figure 2.3: NodeFlow architecture. The method leverages a Neural Oblivious Decision Ensemble (NODE) to process the input vector, extracting a hierarchical representation. This representation conditions a Continuous Normalizing Flow (CNF), enabling flexible modeling of the probabilistic distribution of the multidimensional response vector.

Motivation While TreeFlow demonstrated significant improvements in probabilistic regression, its two-stage training process potentially limited the model's ability to fully optimize the interaction between the tree-based feature extractor and the normalizing flow component. NodeFlow addresses this limitation by proposing an end-to-end approach to flexible probabilistic regression on tabular data, aiming to further enhance the modeling capabilities and performance in uncertainty estimation.

Contents NodeFlow introduces an innovative framework that seamlessly integrates Neural Oblivious Decision Ensembles (NODEs) with Conditional Continuous Normalizing Flows (CNFs) in an end-to-end trainable architecture as presented in Figure 2.3.

At its core, NodeFlow utilizes a Neural Oblivious Decision Ensemble to process the input vector, extracting a hierarchical representation of tabular data. This NODE component combines differentiable oblivious decision trees in a multi-layer structure, enabling complex feature interactions. Building upon this foundation, NodeFlow incorporates a Conditional Continuous Normalizing Flow component. This CNF leverages the NODE output as a conditioning factor to model flexible probability distributions for regression outputs. The synergy between these two components allows NodeFlow to capture intricate relationships within the data while providing accurate and adaptable uncertainty estimates. A key strength of NodeFlow is its end-to-end training approach using gradient-based optimization. This unified training process facilitates joint learning of feature representation and uncertainty modeling, enhancing the system's overall performance.

To validate its effectiveness, comprehensive experiments were conducted on both univariate and multivariate regression benchmarks. These tests pitted NodeFlow against state-of-the-art probabilistic regression methods. The results clearly demonstrated NodeFlow's superior performance, with particularly impressive outcomes on multivariate tasks and datasets characterized by complex underlying distributions.

Summary NodeFlow represents a significant advancement in probabilistic regression for tabular data by offering an end-to-end trainable framework combining the strengths of Neural Oblivious Decision Ensembles and Normalizing Flows. This approach enables more accurate and flexible uncertainty modeling, particularly for high-dimensional, complex regression tasks. The experimental results validate NodeFlow's effectiveness across a range of benchmarks, showcasing its potential to improve decision-making in various real-world applications involving tabular data.

Contribution I was primarily responsible for conducting the literature review, designing the methodology, creating and implementing the initial version of the experiments, analyzing the data, and writing the manuscript. Oleksii Furman played a key role in executing the experiments and refining the experimental design. My supervisor, Maciej Zięba provided invaluable feedback throughout all phases of the research. The work was published in Entropy (IF: 2.1) in July 2024.

2.4 Modeling Uncertainty in Personalized Emotion Prediction with Normalizing Flows

This section covers results from Publication [III], which adapts the ideas from Publications [I] and [II] to the domain of personalized natural language processing. It introduces a probabilistic framework that combines end-to-end training and normalizing flows for improved emotion prediction.

Motivation Emotion recognition in text is a challenging task due to its inherent subjectivity and the variability in individual perceptions. Traditional approaches often rely on generalized point-prediction models that fail to capture personal biases and contextual nuances. The motivation behind this work is to develop a probabilistic framework that can model the uncertainty in personalized emotion prediction, taking into account individual differences and the inherent ambiguity in emotional responses to text.

Contents This work introduces Emotional Normalizing Flow, a novel probabilistic framework designed to model uncertainty in personalized emotion prediction. The approach is centered around three key components: a profile extractor, a text encoder, and a conditional normalizing flow, as presented in Figure 2.4.

The profile extractor generates a representation of the individual, capturing characteristics that are crucial for understanding their emotional responses. Meanwhile, the text encoder transforms the input text into an embedding using transformer-based language models. These two components are then integrated by the conditional normalizing flow, which models the conditional probability distribution of emotional responses, combining the personalized user representation with the text embedding. This framework supports both generalized and personalized approaches to emotion prediction, with the personalized method incorporating specific user information to enhance prediction accuracy. The entire model is trained end-to-end by optimizing the negative log-likelihood of the conditional probability distribution.

Extensive experiments across multiple datasets, including emotion recognition and hate speech detection tasks, reveal the superiority of the personalized approach over

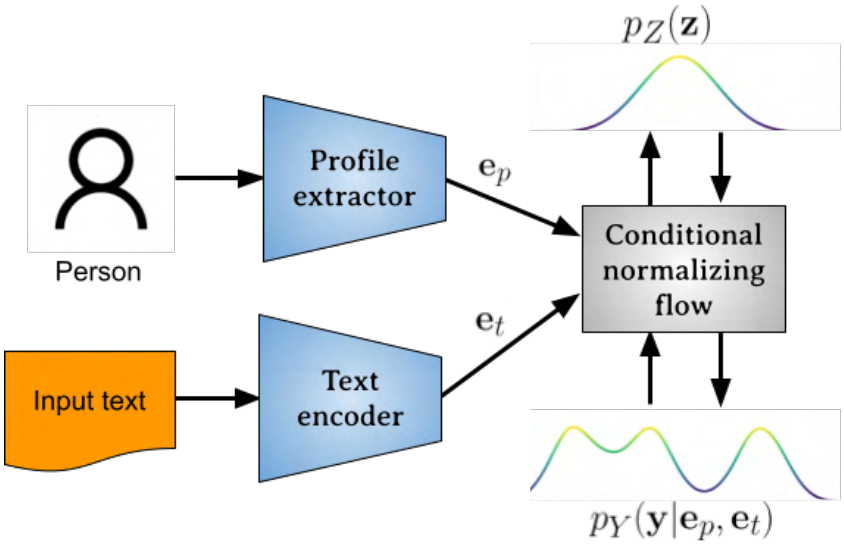


Figure 2.4: Emotional Normalizing Flow architecture: Personalized user profiles and text embeddings are combined through a conditional normalizing flow to model the probability distribution of emotional responses.

generalized models. Additionally, the use of normalizing flows proves more effective than simpler probabilistic baselines such as Gaussian Mixture Models.

Summary Emotional Normalizing Flow represents a significant advancement in personalized emotion prediction by providing a flexible probabilistic framework that can capture individual differences and model complex uncertainty patterns. The proposed approach demonstrates superior performance in both emotion recognition and hate speech detection tasks, highlighting its potential to improve the understanding and analysis of subjective emotional responses in natural language processing applications.

Contribution I contributed to conceptualizing the method and assessing its applicability to the domain, provided feedback on the method's implementation, designed the experiments, and contributed to the writing and review of the manuscript. The article was published in the CORE A conference, the 23rd IEEE International Conference on Data Mining (ICDM 2023), specifically in the Sentiment Elicitation from Natural Text for Information Retrieval and Extraction (SENTIRE) Workshop in December 2023 has received a 1 citation as of October 2024.

CHAPTER 3

Part II: Normalizing Flows for Generative Tasks

3.1 Research Scope

Chapter 3 investigates the application of normalizing flows in generative modeling tasks, focusing on conditional generation and manipulation of complex data such as images and 3D point clouds. The research aims to enhance the flexibility and control of existing generative models by integrating normalizing flows, thereby expanding their applicability across diverse tasks.

Conventional generative models like Variational Autoencoders (VAEs) [19] and Generative Adversarial Networks (GANs) [24] have shown success in generating high-quality samples but often struggle with conditional generation, particularly for high-dimensional and complex data structures. State-of-the-art models such as StyleGAN [36], PointFlow [37] or LION [38], while capable of generating photorealistic images or 3D point clouds, lack native support for fine-grained control over individual attributes within the generated outputs. This limitation poses significant challenges for applications requiring precise and flexible generation in fields like graphics, design, and scientific simulations.

To address these challenges, this chapter introduces two novel methodologies: the Flow Plugin Network (FPN) and PluGeN (Plugin Generative Network). These approaches leverage plugin network framework [39] and normalizing flows to learn invertible mappings between complex distributions and simpler, tractable ones. By enabling more sophisticated manipulation of the latent space of pre-trained generative models, FPN and PluGeN facilitate conditional generation and the disentanglement of latent representations. This research examines the adaptation of these methods to different data modalities, including both 2D images and 3D point clouds, and assesses their effectiveness in generating samples and manipulating attributes with rare or previously unseen attribute combinations.

The significance of this work lies in its proposal of a comprehensive framework for enhancing generative models through normalizing flows, offering novel solutions to existing challenges in conditional generation and attribute manipulation. By advancing the state of the art in generative modeling, this research establishes a foundation for future improvements in the control and expressiveness of generative models, with potential applications spanning various domains where high-quality, controlled generation is essential.

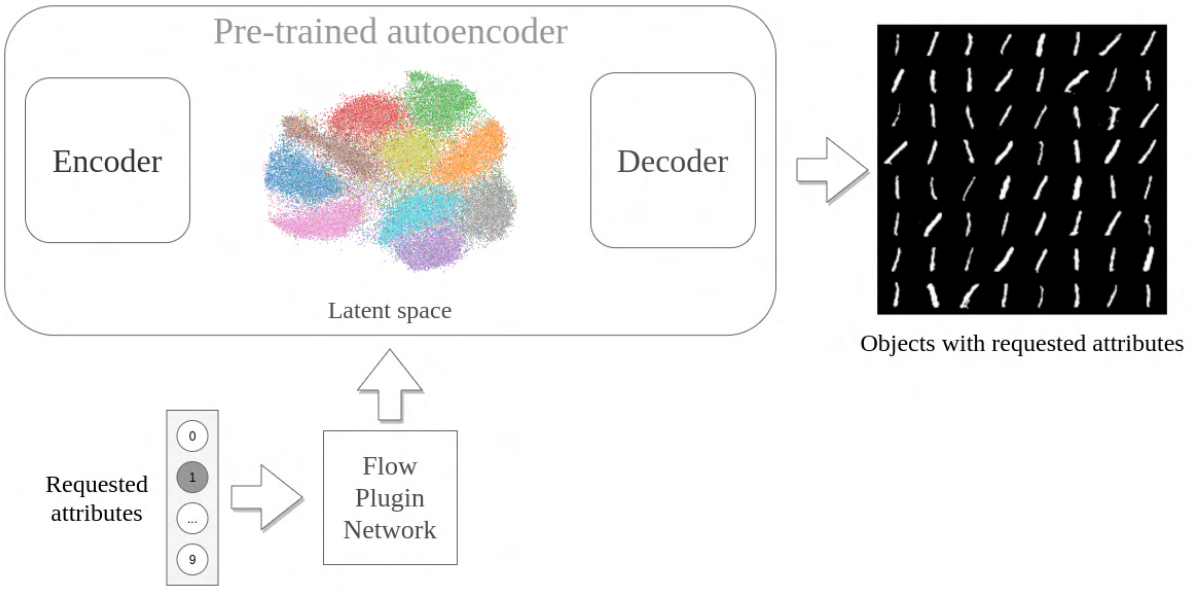


Figure 3.1: The Flow Plugin Network, when connected to the latent space of a pre-trained autoencoder (whether generative or non-generative), facilitates the generation of objects with specified attributes.

3.2 Flow Plugin Network for Conditional Generation

This section covers results from Publication [IV], which explores enabling conditional object generation from pre-trained generative models using normalizing flows utilized as a plug-in network.

Motivation While modern generative models like GANs and VAEs can produce high-quality samples, they often lack fine-grained control over the generated outputs. Conditional variants of these models typically require training from scratch, which is computationally expensive and time-consuming. The Flow Plugin Network (FPN) aims to address this limitation by enabling conditional generation capabilities in pre-trained generative models without modifying their original parameters.

Contents The Flow Plugin Network (FPN) introduces a novel architecture that extends the functionality of pre-trained generative models by integrating a conditional normalizing flow within the latent space. This approach builds upon a pre-existing encoder-decoder framework, leveraging the latent representations generated by the base model. Central to the FPN is the conditional normalizing flow, which establishes a bidirectional mapping between a simple base distribution and the latent space of the generative model. This mapping is conditioned on specific attributes, enabling the flow to incorporate additional information into the transformation process.

In FPN, the primary objective is to model the conditional distribution $p(z|y)$, where z denotes the latent vector and y represents the desired attributes. This modeling capability facilitates both conditional generation and attribute manipulation. The training process involves encoding a dataset of samples to obtain their corresponding latent representations. The conditional flow is then trained to map these latent representations to a simple base distribution, with the mapping conditioned on the attributes. The op-

timization process focuses on minimizing the conditional negative log-likelihood of the latent representations. It is crucial to note that throughout this process, the weights of the base generative model remain fixed, ensuring that the original model's capabilities are preserved.

Once trained, the FPN is capable of performing conditional generation by sampling from the base distribution and passing the sample through the conditional flow, followed by the decoder, to produce an output that exhibits the desired attributes. Attribute manipulation is achieved by encoding an input into the latent space, mapping it to the base distribution, modifying the attributes as needed, and then decoding back to the output space to reflect these changes.

Extensive evaluations on multiple datasets, including MNIST, CelebA, and ShapeNet, demonstrate the effectiveness of the FPN architecture. These experiments illustrate the model's ability to extend the capabilities of pre-trained generative models, facilitating tasks such as conditional image and 3D point cloud generation, attribute manipulation, and even classification. The results underscore the FPN's potential to enhance the flexibility and functionality of existing generative models by enabling conditional generation capabilities while maintaining the integrity of the original model's latent space.

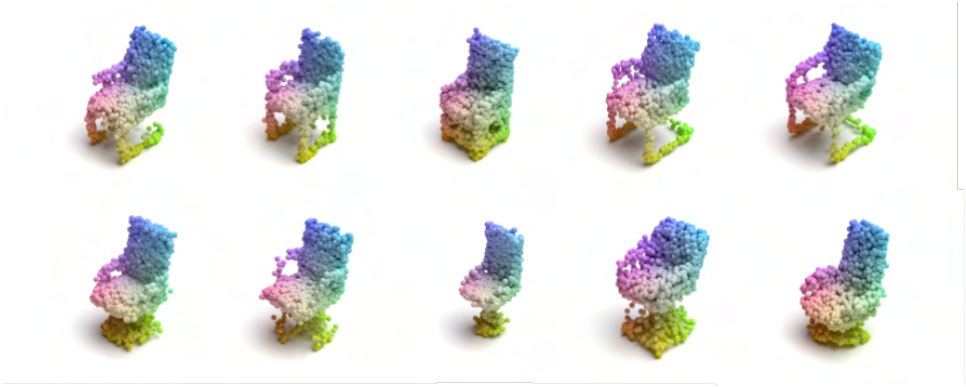
Summary The Flow Plugin Network provides a flexible approach for adding conditional generation capabilities to pre-trained generative models. By leveraging normalizing flows, FPN enables fine-grained control over generated outputs without requiring retraining of the base model. This allows for efficient adaptation of existing models to new tasks or attribute-based generation requirements.

Contribution I was responsible for conducting the literature review, designing the research methodology, executing the experiments, analyzing and interpreting the data, and writing the manuscript. Maciej Zięba and Michał Koperski contributed by providing guidance on methodological and experimental design, assisting with result interpretation, and offering critical feedback during the manuscript's writing and revision stages. This work was presented at the 15th Asian Conference on Intelligent Information and Database Systems, a CORE B conference, in July 2023, and has been cited twice as of October 2024.

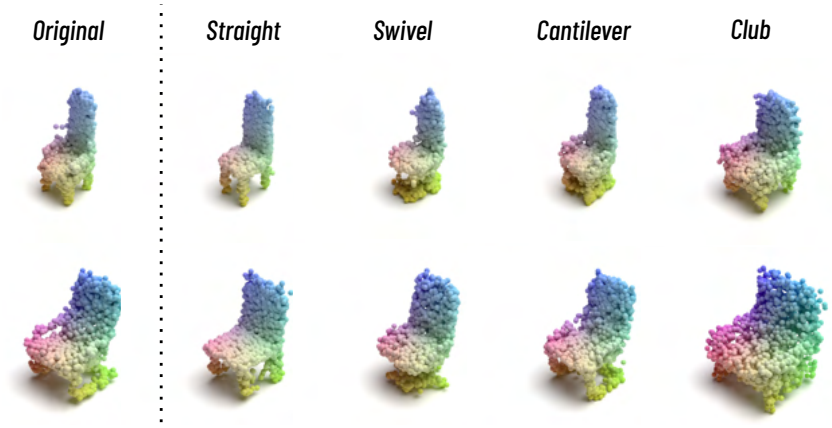
3.3 PluGeN: Multi-Label Conditional Generation From Pre-Trained Models

This section covers results from Publications [V, VI], which extends the ideas from Publication [IV] and enables multi-label conditional generation of rare attribute combinations from both VAEs and GANs models.

Motivation While FPN demonstrated the potential of using normalizing flows for conditional generation, it was limited in its ability to handle multiple attributes simultaneously. PluGeN (Plugin Generative Network) extends this concept to enable multi-label conditional generation and manipulation, providing even greater control over generated outputs. Additionally, PluGeN aims to disentangle the latent space to allow for more precise attribute control and generation of rare attribute combinations.



(a) PluGeN for conditional generation.



(b) FPN for attribute manipulation.

Figure 3.2: The samples generated using PluGeN (a) illustrate different chair types: cantilever armchairs in the first row and swivel armchairs in the second row. The results of point cloud attribute manipulation for the Flow Plug-in Network (FPN) (b) show transformations to straight, swivel, cantilever, and club chair types.

Contents PluGeN introduces several key enhancements and novel concepts that build upon and improve the FPN approach. One of the primary innovations is the transformation of the entangled latent representation of the base model into a factorized latent space, where each dimension corresponds to a distinct, interpretable attribute. This disentangled representation allows for more precise control over individual attributes, as the conditional probability distribution within the latent space is modeled as a product of independent factors.

Additionally, PluGeN offers a flexible attribute parameterization system, providing continuous parameterization for both binary and continuous attributes. This enables smooth interpolation and extrapolation of attribute values, enhancing the model's adaptability. Moreover, PluGeN is designed to be trained on partially labeled datasets, which makes it particularly suitable for real-world scenarios where complete labeling is often impractical. This semi-supervised training capability is complemented by improved disentanglement, as PluGeN factorizes the true data distribution into independent components. This allows the model to generate samples with rare or unseen combinations of attributes, broadening its applicability.

The effectiveness of the PluGeN approach is demonstrated through evaluations on various datasets, including CelebA for facial attribute manipulation, FFHQ for high-quality

face generation, ShapeNet for 3D point clouds, and a dataset of chemical compounds for molecular property optimization. These experiments highlight PluGeN's ability to generate high-quality samples with precise attribute control, manipulate existing samples effectively, and even produce samples with uncommon attribute combinations.

Summary PluGeN advances the concept of flow-based plug-in networks for conditional generation by enabling multi-label control, attribute disentanglement, and semi-supervised training. This provides a powerful and flexible tool for enhancing the capabilities of pre-trained generative models across various domains.

Contribution As the author of Publications [V, VI], I contributed to the development of the method and the formulation of the experimental methodology. I was primarily responsible for conducting all experiments involving 3D point clouds, including those related to the PluGeN and FPN methods, as well as all FPN-related experiments on image datasets. Publication [V], presented at the 36th AAAI Conference on Artificial Intelligence in February 2022, has garnered five citations as of October 2024. Publication [VI], on the other hand, appeared in Transactions on Pattern Analysis and Machine Intelligence in March 2024.

CHAPTER 4

Part III: Normalizing Flows for Combined Tasks

4.1 Research Scope

In the rapidly evolving landscape of machine learning, the boundaries between discriminative and generative tasks are becoming increasingly blurred. This chapter explores a novel approach that leverages the strengths of both paradigms, focusing on the domain of Explainable AI. By combining these traditionally distinct areas, we aim to enhance the interpretability and transparency of complex machine learning models, addressing a critical need in the deployment of AI systems across sensitive domains.

At the heart of this research lies the development of a new method for generating counterfactual explanations, a key tool in XAI [22]. Counterfactual explanations provide insights into model decisions by illustrating how input features would need to change to alter the model's prediction [40]. However, traditional approaches to counterfactual generation often focus solely on finding minimal changes to flip model predictions without considering whether these changes result in plausible data points [22].

To address this limitation, we introduce Probabilistically Plausible Counterfactual Explanations using Normalizing Flows (PPCEF). This novel approach integrates normalizing flows into the process of counterfactual generation. By modeling the underlying data manifold of the training set, normalizing flows allows us to generate counterfactuals that not only achieve the desired class change but also align with the true data distribution, enhancing their plausibility and interoperability.

The use of normalizing flows in this context represents a significant advancement in the field of Explainable AI (XAI). These models provide unique flexibility in modeling complex data distributions, surpassing traditional methods such as Kernel Density Estimation or Gaussian Mixture Models. By leveraging these capabilities, the PPCEF method achieves a delicate balance between the discriminative task of altering model predictions and the generative task of modeling the underlying training data distribution.

The implications of this research extend beyond the realm of technical innovation. By providing a principled method for generating plausible counterfactual explanations, the work addresses crucial ethical and practical considerations in the deployment of AI systems. This is particularly relevant in fields such as finance, healthcare, and legal systems, where the transparency and interpretability of AI-driven decisions can have significant real-world impacts.

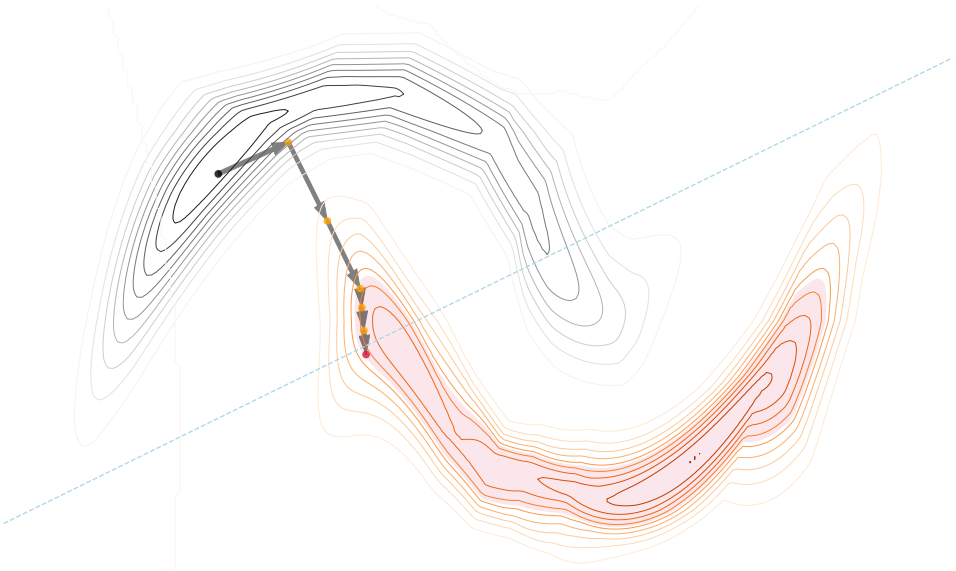


Figure 4.1: Probabilistically Plausibile Counterfactual Explanation Estimation Process on the Moons Dataset. The image shows an evolution of an instance from the initial instance (black dot) to the final counterfactual (red dot) against the linear classifier's decision boundary (blue line) and density threshold contours, highlighting the method's trajectory towards achieving target classification and probabilistic plausibility condition.

4.2 Probabilistically Plausible Counterfactual Explanations with Normalizing Flows

This section covers results from Publication [VII], which explores the application of normalizing flows in generating probabilistically plausible counterfactual explanations for complex machine learning models.

Motivation The work on Probabilistically Plausible Counterfactual Explanations with Normalizing Flows (PPCEF) addresses the critical need for interpretable and reliable AI explanations as these systems impact crucial life decisions. Existing methods for generating counterfactuals often produce explanations that are mathematically valid but not necessarily realistic or useful, as they fail to ensure that suggested changes align with real-world data distributions. By using normalizing flows to enhance density estimation in counterfactual generation, the approach aims to produce explanations that not only alter model predictions but also remain plausible, improving the trustworthiness and practical value of AI explanations.

Contents The work introduces a novel method for generating probabilistically plausible counterfactual explanations using normalizing flows (as presented in Figure 4.1). Building on Artelt and Hammer's approach [41], which incorporates a probabilistic plausibility constraint, the method utilizes normalizing flows to model complex, high-dimensional data distributions with exact likelihood computation, offering advantages over alternatives like Gausian Mixture Models (GMMs) and Kernel Density Estimators (KDEs).

The method frames counterfactual generation as an unconstrained optimization problem. It includes a validity term to ensure the counterfactual produces the desired change

in model prediction, a proximity term to keep the counterfactual close to the original input in feature space, and a plausibility term that uses density estimates from the normalizing flow to ensure the counterfactual is located in a high-density region of the data distribution.

This formulation supports efficient gradient-based optimization, enabling a balance between validity, proximity, and plausibility. The approach is effective across various datasets and model types, demonstrating that it generates counterfactuals that are not only valid and proximate but also more plausible than those produced by existing methods. The practical implications of this method for improving model interpretability and decision transparency are also discussed, addressing the trade-offs involved in balancing these objectives.

Summary The work on Probabilistically Plausible Counterfactual Explanations with Normalizing Flows advances Explainable AI by integrating normalizing flows to accurately model complex data distributions, thus generating counterfactual explanations that are both valid and highly plausible. The study presents an unconstrained optimization framework that balances validity, proximity, and plausibility, demonstrating that this method outperforms existing approaches in producing more realistic counterfactuals, especially in high-dimensional datasets. The approach combines discriminative and generative modeling techniques to enhance model interpretability and decision transparency, offering a significant improvement in counterfactual plausibility and utility across various datasets and models.

Contribution I conducted the literature review, designed the methodology, oversaw and debugged the implementation of methods and experiments, visualized the results, and wrote the manuscript. Oleksii Furman contributed by focusing on method implementation, conducting experiments, and co-authoring the manuscript. Maciej Zięba and Jerzy Stefnaowski provided crucial guidance on method design, experimental design, and analysis. They also assisted with interpreting results and offered valuable feedback during the writing and revision stages. This work is scheduled to be presented at the CORE A conference, ECAI 2024, in October 2024.

CHAPTER 5

Contribution to Research Community

While the Ph.D. journey primarily focuses on publishing, the community is an inseparable part of research. In this section, I'd like to discuss my efforts and successes in contributing to the research community at local, national, and international levels.

I'd like to start with personally my most important contribution. Together with my supervisor, we created a research group called *genwro.ai*, and I was responsible for co-leading the group for the last 1.5 years. The tasks, while they seemed trivial, like setting up Slack and biweekly meetings, working with other Ph.D. students to publish the webpage [1], creating a logo (see Figure 5.1), and ensuring we all found time for a small integration party, led to something I believe constitutes one of my biggest successes in the non-publishing part of my Ph.D. – collaborative and supportive research environment. It's personally crucial for me as it represented a reconnection with research colleagues after the challenging COVID times (which were still present at the beginning of my Ph.D. journey in October 2021). As I complete my studies, I feel immense pride in our entire group, particularly noting the influx of talented students from engineering degrees, master's programs, and Ph.D. studies joining and initiating impressive new projects.

Regarding giving back to the international community, I served as a reviewer for top AI conferences such as NeurIPS, ICML, AISTATS, ECAI, and KDD. Moreover, I had the opportunity to review more local initiatives like the PPRAI conference and the Eastern European Machine Learning Summer School. Moving forward to national initiatives, I conducted workshops titled "Conditional object generation using pre-trained models and plug-in networks," during which I introduced normalizing flows (and their application as FPN) to AI Master's students in Poland. These workshops were held twice during the AI Tech Summer School in Gdańsk (2022) and Warsaw (2023).

Furthermore, on the local level, as a continuation of my five-year-long involvement during engineering and master studies with the Scientific Research Group of Mathematical Statistics "Gauss" at the Faculty of Pure and Applied Mathematics at Wrocław University of Science and Technology, I served for one year as a mentor. During this time, I delivered a presentation on academic and industry career paths (based on my experience) to group members. Moreover, I organized a series of workshops called "Hack the Hackathon!" where colleagues and I shared our experiences of winning multiple hackathons. Finally, I had the pleasure of conducting laboratory classes for the Machine Learning course in the Artificial Intelligence Master's program at the Department of Artificial Intelligence of Wrocław University of Science and Technology. It's wonderful to observe the growth of these students, and I hope that the experience I shared during these classes will benefit them, especially since I still have the opportunity to observe the progress of some students closely.

[1] `https://genwro.ai.pwr.edu.pl/`

Figure 5.1: Logo for genwro.ai, a research group I co-founded with my Ph.D. supervisor, prof. Maciej Zięba.

Contributing to the community also involves giving talks and presentations. While it's difficult to recall all the presentations, here is a list of some of the most significant ones:

- Ogólnopolska Matematyczna Konferencja Studentów "OMatKo!!!", 11.2021, Online, Poland, "Can the lady in the photo smile?" (Talk)
- Ogólnopolska Matematyczna Konferencja Studentów "OMatKo!!!", 11.2021, Online, Poland, "Generation on demand" (Poster)
- 26th European Conference on Artificial Intelligence ECAI 2023, 10.2023, Krakow, Poland, "TreeFlow: Going Beyond Tree-based Parametric Probabilistic Regression" accepted and presented at ECAI 2023 (Talk and Poster)
- ML in PL Conference 2023, 11.2023, Warsaw, Poland, "TreeFlow: Going Beyond Tree-based Parametric Probabilistic Regression" (Talk and Poster)

I am very proud to share that the audience appreciated my contributions by awarding me the Best Contributed Talk Award at ML in PL Conference 2023 and 3rd Place in the competition for the best lecture delivered during the Ogólnopolska Matematyczna Konferencja Studentów "OMatKo!!!" in 2021.

Finally, in terms of collaboration, I've contributed to four initiatives:

- Contractor in NCN Grant no. 2021/43/B/ST6/02853 "Generative flow models for modeling uncertainty in machine learning tasks"; 10.2022 - 06.2023 & 10.2023 - Present
- Joint Publications [V, VI] with GMUM (Group of Machine Learning Research) from Jagiellonian University during the period of 10.2021 - 03.2024
- Joint Publication [III] with CLARIN-PL from Wrocław University of Science and Technology during the period 06.2022 - 12.2023
- Joint Publication [VII] (and more not published yet) with Jerzy Stefanowski from Poznan University of Technology during the period 01.2024 - now

In summary, my contributions to the research community took many forms and were at various levels, from local to international. As I move forward in my career, I aim to continue building on these efforts, nurturing talent, sharing knowledge, and contributing to the growth and evolution of the research community in the fields of AI and machine learning. This journey has not only been about producing new knowledge but also about fostering a supportive, collaborative, and innovative research environment that I hope will continue to thrive long after my formal studies are completed.

CHAPTER 6

Conclusions

This doctoral thesis has explored the application of normalizing flows across a spectrum of machine learning tasks, demonstrating their versatility and potential to advance the field in multiple directions. The research presented here makes several key contributions to the areas of discriminative modeling, generative modeling, and their intersection in Explainable AI.

In the realm of discriminative tasks, I introduced novel architectures that integrate normalizing flows with traditional machine learning models to enhance probabilistic regression. TreeFlow and NodeFlow represent significant advancements in flexible uncertainty modeling for tabular data, overcoming the limitations of conventional tree-based methods. Applying these techniques to personalized emotion prediction further demonstrates their broad applicability and potential impact on real-world problems.

For generative tasks, my work on Flow Plugin Networks (FPN) and PluGeN showcases how normalizing flows can extend the capabilities of pre-trained generative models. These approaches enable fine-grained control over generated outputs and facilitate the disentanglement of latent representations, opening new possibilities for conditional generation and attribute manipulation in both 2D and 3D domains.

My research on probabilistically plausible counterfactual explanations (PPCEF) leverages normalizing flows at the intersection of discriminative and generative paradigms to enhance the interpretability of machine learning models. By generating counterfactuals that are both valid and plausible, this work addresses a critical need in Explainable AI, potentially improving trust and transparency in high-stakes decision-making scenarios.

The methodologies developed in this thesis not only advance the state-of-the-art in their respective areas but also demonstrate the broader potential of applicability of normalizing flows across various machine learning tasks. By providing flexible and powerful tools for modeling complex probability distributions, normalizing flows offers a pathway to more accurate, controllable, and interpretable machine learning systems.

Looking forward, this research opens several promising avenues for future work:

- Scaling normalizing flow-based methods to larger datasets and more complex domains, potentially leveraging recent advancements in efficient flow architectures.
- Exploring the integration of normalizing flows with other emerging paradigms in machine learning, such as self-supervised learning or few-shot learning.
- Investigating the theoretical properties of these hybrid models, particularly in terms of their expressiveness and generalization capabilities.
- Extending the application of these techniques to new domains, such as time-series analysis, cluster analysis, or multi-modal learning.

- Further developing the use of normalizing flows in Explainable AI, potentially exploring their application to other forms of model interpretation beyond counterfactual explanations.

In conclusion, this thesis has showcased the remarkable power and versatility of normalizing flows across a wide range of machine learning tasks. The research presented here makes significant contributions to the advancement of more sophisticated and reliable AI systems. As the field of machine learning continues to evolve, the methods developed in this work offer promising pathways for creating models that are not only more powerful but also more transparent and adaptable to practical, real-world applications.

Acknowledgments

The conducted work was supported by the National Centre of Science (Poland) Grant No. 2021/43/B/ST6/02853. Moreover, I gratefully acknowledge Polish high-performance computing infrastructure PLGrid (HPC Center: ACK Cyfronet AGH) for providing computer facilities and support within computational grant no. PLG/2023/016636.

Bibliography

[1] Y. LeCun, Y. Bengio, and G. E. Hinton, "Deep learning," *Nat.*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science*, vol. 349, no. 6245, pp. 255–260, 2015.

[3] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. P. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nat.*, vol. 529, no. 7587, pp. 484–489, 2016.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 5998–6008, 2017.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, pp. 1106–1114, 2012.

[6] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016* (M. Balcan and K. Q. Weinberger, eds.), vol. 48 of *JMLR Workshop and Conference Proceedings*, pp. 1050–1059, JMLR.org, 2016.

[7] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 5574–5584, 2017.

[8] E. Hüllermeier and W. Waegeman, "Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods," *Mach. Learn.*, vol. 110, no. 3, pp. 457–506, 2021.

[9] K. P. Murphy, *Machine learning - a probabilistic perspective*. Adaptive computation and machine learning series, MIT Press, 2012.

[10] C. M. Bishop, *Pattern recognition and machine learning, 5th Edition*. Information science and statistics, Springer, 2007.

[11] A. Brock, J. Donahue, and K. Simonyan, "Large scale GAN training for high fidelity natural image synthesis," in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, OpenReview.net, 2019.

[12] U. Bhatt, A. Xiang, S. Sharma, A. Weller, A. Taly, Y. Jia, J. Ghosh, R. Puri, J. M. F. Moura, and P. Eckersley, "Explainable machine learning in deployment," in *FAT* '20: Conference on Fairness, Accountability, and Transparency, Barcelona, Spain, January 27-30, 2020*, pp. 648–657, ACM, 2020.

[13] B. Efron and R. Tibshirani, *An Introduction to the Bootstrap*. Springer, 1993.

[14] A. K. David M. Blei and J. D. McAuliffe, "Variational inference: A review for statisticians," *Journal of the American Statistical Association*, vol. 112, no. 518, pp. 859–877, 2017.

[15] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 6402–6413, 2017.

[16] C. E. Rasmussen and C. K. I. Williams, *Gaussian processes for machine learning*. Adaptive computation and machine learning, MIT Press, 2006.

[17] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pp. 785–794, ACM, 2016.

[18] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T. Liu, "LightGBM: A Highly Efficient Gradient Boosting Decision Tree," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 3146–3154, 2017.

[19] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[20] T. Duan, A. Anand, D. Y. Ding, K. K. Thai, S. Basu, A. Y. Ng, and A. Schuler, "NGBoost: Natural Gradient Boosting for Probabilistic Prediction," in *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, vol. 119 of *Proceedings of Machine Learning Research*, pp. 2690–2700, PMLR, 2020.

[21] A. Malinin, L. Prokhorenkova, and A. Ustimenko, "Uncertainty in Gradient Boosting via Ensembles," in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, OpenReview.net, 2021.

[22] R. Guidotti, "Counterfactual explanations and how to find them: literature review and benchmarking," *Data Mining and Knowledge Discovery*, pp. 1–55, 04 2022.

[23] D. J. Rezende and S. Mohamed, "Variational Inference with Normalizing Flows," in *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, vol. 37 of *JMLR Workshop and Conference Proceedings*, pp. 1530–1538, JMLR.org, 2015.

[24] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, pp. 2672–2680, 2014.

[25] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[26] G. Papamakarios, E. T. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan, "Normalizing flows for probabilistic modeling and inference," *J. Mach. Learn. Res.*, vol. 22, pp. 57:1–57:64, 2021.

[27] L. Dinh, D. Krueger, and Y. Bengio, "NICE: non-linear independent components estimation," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings*, 2015.

[28] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using Real NVP," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, OpenReview.net, 2017.

[29] G. Papamakarios, I. Murray, and T. Pavlakou, "Masked autoregressive flow for density estimation," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 2338–2347, 2017.

[30] T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, "Neural ordinary differential equations," in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 6572–6583, 2018.

[31] Z. Obermeyer and E. Emanuel, "Predicting the future — big data, machine learning, and clinical medicine," *The New England journal of medicine*, vol. 375, pp. 1216–1219, 09 2016.

[32] T. Blasco, J. S. Sánchez, and V. García, "A survey on uncertainty quantification in deep learning for financial time series prediction," *Neurocomputing*, vol. 576, p. 127339, 2024.

[33] E. Goan and C. Fookes, *Bayesian Neural Networks: An Introduction and Survey*, pp. 45–87. Cham: Springer International Publishing, 2020.

[34] L. O. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "CatBoost: unbiased boosting with categorical features," in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 6639–6649, 2018.

[35] O. Sprangers, S. Schelter, and M. de Rijke, "Probabilistic gradient boosting machines for large-scale probabilistic regression," in *KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14-18, 2021*, pp. 1510–1520, ACM, 2021.

[36] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pp. 4401–4410, Computer Vision Foundation / IEEE, 2019.

[37] G. Yang, X. Huang, Z. Hao, M. Liu, S. J. Belongie, and B. Hariharan, "PointFlow: 3D Point Cloud Generation With Continuous Normalizing Flows," in *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pp. 4540–4549, IEEE, 2019.

[38] X. Zeng, A. Vahdat, F. Williams, Z. Gojcic, O. Litany, S. Fidler, and K. Kreis, "LION: latent point diffusion models for 3d shape generation," in *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

[39] M. Koperski, T. K. Konopczynski, R. Nowak, P. Semberecki, and T. Trzcinski, "Plugin networks for inference under partial evidence," in *IEEE Winter Conference on Applications of Computer Vision, WACV 2020, Snowmass Village, CO, USA, March 1-5, 2020*, pp. 2872–2880, IEEE, 2020.

[40] S. Wachter, B. D. Mittelstadt, and C. Russell, "Counterfactual explanations without opening the black box: Automated decisions and the GDPR," *CoRR*, vol. abs/1711.00399, 2017.

[41] A. Artelt and B. Hammer, "Convex density constraints for computing plausible counterfactual explanations," in *Artificial Neural Networks and Machine Learning - ICANN 2020 - 29th International Conference on Artificial Neural Networks, Bratislava, Slovakia, September 15-18, 2020, Proceedings, Part I*, vol. 12396 of *Lecture Notes in Computer Science*, pp. 353–365, Springer, 2020.

CHAPTER A

Full Texts of Publications

This chapter presents the publications referenced in the main text, arranged in the following order:

[I] **P. Wielopolski**, M. Zięba *"TreeFlow: Going Beyond Tree-based Parametric Probabilistic Regression"*

[II] **P. Wielopolski**, O. Furman, M. Zięba *"NodeFlow: Towards End-to-end Flexible Probabilistic Regression on Tabular Data"*

[III] P. Miłkowski, K. Karanowski, **P. Wielopolski**, J. Kocoń, P. Kazienko, M. Zięba *"Modeling Uncertainty in Personalized Emotion Prediction with Normalizing Flows"*

[IV] **P. Wielopolski**, M. Koperski, M. Zięba *"Flow Plugin Network for conditional generation"*

[V] M. Wołczyk, M. Proszewska, Ł. Maziarka, M. Zięba, **P. Wielopolski**, R. Kurczab, M. Śmieja *"PluGeN: Multi-Label Conditional Generation from Pre-trained Models"*

[VI] M. Proszewska, M. Wołczyk, M. Zięba, **P. Wielopolski**, Ł. Maziarka, M. Śmieja *"Multi-Label Conditional Generation From Pre-Trained Models"*

[VII] **P. Wielopolski**, O. Furman, J. Stefanowski, M. Zięba *"Probabilistically Plausible Counterfactual Explanations with Normalizing Flows"*

ECAI 2023
K. Gal et al. (Eds.)

doi:10.3233/FAIA230570

TreeFlow: Going Beyond Tree-Based Parametric Probabilistic Regression

Patryk Wielopolski[a, b;*] **and Maciej Zięba**[a, c]

[a]Wrocław University of Science and Technology
[b]DataWalk
[c]Tooploox

Abstract. The tree-based ensembles are known for their outstanding performance in classification and regression problems characterized by feature vectors represented by mixed-type variables from various ranges and domains. However, considering regression problems, they are primarily designed to provide deterministic responses or model the uncertainty of the output with Gaussian or parametric distribution. In this work, we introduce TreeFlow, the tree-based approach that combines the benefits of using tree ensembles with the capabilities of modeling flexible probability distributions using normalizing flows. The main idea of the solution is to use a tree-based model as a feature extractor and combine it with a conditional variant of normalizing flow. Consequently, our approach is capable of modeling complex distributions for the regression outputs. We evaluate the proposed method on challenging regression benchmarks with varying volume, feature characteristics, and target dimensionality. We obtain the SOTA results for both probabilistic and deterministic metrics on datasets with multi-modal target distributions and competitive results on unimodal ones compared to tree-based regression baselines.

1 Introduction

The modern tree-based models achieve outstanding results for problems where the data representation is tabular, the number of training examples is limited, and the input feature vector is represented by mixed-type variables from various ranges and domains. Most of such algorithms focus on providing deterministic predictions, paying no attention to the probabilistic nature of the provided output. However, for many practical applications, it is impossible to deliver an exact target value based on the given input factors. Consider the regression problem of predicting the future location of the vehicle that is approaching a roundabout [29]. Having past coordinates and other information aggregated in current and past states, we cannot unambiguously predict which of the three remaining exits from the roundabout will be taken by the tracked object. Therefore, it is more beneficial to provide multimodal probability distribution for future locations instead of a single deterministic prediction oscillating around one mode.

Due to the tractable closed-form, the standard approaches assume to model regression uncertainty using Gaussian or parametric distributions [17]. The well-known deterministic gradient boosting machine method adopted those approaches to tree-based structures [6, 13, 25]. Consequently, they can capture the uncertainty of the regression outputs with a standard family of distributions.

* Corresponding Author. Email: patryk.wielopolski@pwr.edu.pl

The major limitation of the current approaches is modeling regression outputs using only Gaussians. Moreover, it is not trivial to extend them to a mixture of Gaussians to capture the multi-modalities of the predictions. Creating multivariate extensions of such models is also ineffective, especially for higher dimensionality, due to the need to estimate the complete covariance matrix.

To reduce the limitations of existing methods, we introduce TreeFlow - a novel tree-based approach for modeling probabilistic regression. The proposed method combines the benefits of using tree-based structures as feature extractors with the normalizing flows [22] capable of modeling flexible data distributions. We introduce the novel concept of combining forest structure with a conditional flow variant to model uncertainty for regression output. Thanks to that approach, we can model complex non-Gaussian or in general non-parametric data distributions even for high-dimensional predictions. We confirm the quality of the proposed model in the experimental part, where we show the superiority of our method over the baselines.

To summarize, our contributions are as follows:

- According to our knowledge, for the first time, tree-based models are used to model non-parametric probabilistic regression for both uni- and multi-variate predictions.
- We propose a novel approach for combining tree-based models with conditional flows via binary representation of the forest structure.
- We obtain the SOTA results for both probabilistic (NLL, CRPS) and deterministic (RMSE) metrics on datasets with multi-modal target distributions and competitive results on unimodal ones compared to tree-based regression baselines.

2 Background

Assume we have a dataset $\mathcal{D} = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1..N}$ where $\mathbf{x}_n = (x_n^1, \ldots, x_n^D)$ is a D-dimensional random vector of features and $\mathbf{y}_n = (y_n^1, \ldots, y_n^P)$ is a P-dimensional vector of targets. We consider regression problems, thus, we assume that $y_n^p \in \mathbb{R}$. Additionally, when $P = 1$ we will refer to that as a univariate regression problem, and when $P \geq 2$ as a multivariate regression problem.

For the probabilistic regression task, we aim at modeling conditional probability distribution $p(\mathbf{y}|\mathbf{x})$. Assuming some parametrization of the regression model $\boldsymbol{\theta}$, the problem of training the probabilistic model can be expressed as minimisation of the conditional negative log likelihood function (NLL) given by $Q(\boldsymbol{\theta}) = -\sum_{n=1}^{N} \log p(\mathbf{y}_n|\mathbf{x}_n, \boldsymbol{\theta})$. During the training procedure we aim at finding $\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} Q(\boldsymbol{\theta})$.

Decision Tree Ensembles Decision Tree [2] recursively partition feature space $\mathbb{R}^D$ into K disjoint regions $\mathcal{R}_k$ (tree leaves) and for each region assign value w_k. Formally, the model can be written as $h(\mathbf{x}) = \sum_{k=1}^{K} w_k \mathbf{1}_{\{\mathbf{x} \in \mathcal{R}_k\}}$.

Decision Tree Ensembles are constructed of multiple, usually shallow decision trees, whose results are differently aggregated depending on the training mode. In general, we distinguish two main approaches: independent model training with average or majority voting such as Random Forest [1], and iterative model training with additive aggregation such as Gradient Boosting Machine (GBM) [8].

For the univariate probabilistic regression, GBM optimizes the loss function given by negative log likelihood (NLL). Then it assumes the target variable has a Gaussian distribution, i.e.,

$$p(y|\mathbf{x}, \boldsymbol{\theta}^{(t)}) = \mathcal{N}(y|\mu^{(t)}, \sigma^{(t)}), \tag{1}$$

where $\{\mu^{(t)}, \log \sigma^{(t)}\} = F^{(t)}(\mathbf{x})$ and $F^{(t)}(\mathbf{x})$ is an output of t-th tree from GBM model consisted of T trees. In the multivariate case, it assumes Multivariate Normal distribution and uses the parametrization trick with Cholesky decomposition of the covariance matrix which reduces the number of parameters.

In practice, NGBoost [6] supports both uni- and multi-variate Gaussian distributions and estimates each parameter using one underlying model. CatBoost [13] supports only univariate Gaussians but estimates all distribution parameters using only one model. Moreover, it provides deterministic multivariate regression with the same property, which keeps the total number of trees relatively small. In this case, the loss function is Multioutput Root Mean Squared Error (MultiRMSE).

Normalizing Flows Normalizing flows [22] represent the group of generative models that can be efficiently trained via direct likelihood estimation thanks to the application of a change-of-variable formula. Practically, they utilize a series of (parametric) invertible functions: $\mathbf{y} = \mathbf{f}_n \circ \cdots \circ \mathbf{f}_1(\mathbf{z})$. Assuming given base distribution $p(\mathbf{z})$ for $\mathbf{z}$, the log likelihood for $\mathbf{y}$ is given by $\log p(\mathbf{y}) = \log p(\mathbf{z}) - \sum_{n=1}^{N} \log \left| \det \frac{\partial \mathbf{f}_n}{\partial \mathbf{z}_{n-1}} \right|$. In practical applications $p(\mathbf{y})$ represents the distribution of observable data and $p(\mathbf{z})$ is usually assumed to be Gaussian with independent components.

The sequence of discrete transformations can be replaced by continuous alternative by application of Continuous Normalizing Flows (CNFs) [3, 9] where the aim is to solve the differential equation of the form $\frac{\partial \mathbf{z}}{\partial t} = \mathbf{g}_{\boldsymbol{\beta}}(\mathbf{z}(t), t)$, where $\mathbf{g}_{\boldsymbol{\beta}}(\mathbf{z}(t), t)$ represents the function of dynamics, described by parameters $\boldsymbol{\beta}$. Our goal is to find solution of the equation in t_1, $\mathbf{y} := \mathbf{z}(t_1)$, assuming the given initial state $\mathbf{z} := \mathbf{z}(t_0)$ with a known prior. The transformation function $\mathbf{f}_{\boldsymbol{\beta}}$ is defined as:

$$\mathbf{y} = \mathbf{f}_{\boldsymbol{\beta}}(\mathbf{z}) = \mathbf{z} + \int_{t_0}^{t_1} \mathbf{g}_{\boldsymbol{\beta}}(\mathbf{z}(t), t) dt. \tag{2}$$

The inverted form of the transformation can be easily computed using the formula: $\mathbf{f}_{\boldsymbol{\beta}}^{-1}(\mathbf{y}) = \mathbf{y} - \int_{t_0}^{t_1} \mathbf{g}_{\boldsymbol{\beta}}(\mathbf{z}(t), t) dt$. The log-probability of $\mathbf{y}$ can be computed by:

$$\log p(\mathbf{y}) = \log p(\mathbf{f}_{\boldsymbol{\beta}}^{-1}(\mathbf{y})) - \int_{t_0}^{t_1} \operatorname{Tr}\left(\frac{\partial \mathbf{g}_{\boldsymbol{\beta}}}{\partial \mathbf{z}(t)}\right) dt, \tag{3}$$

where $\mathbf{f}_{\boldsymbol{\beta}}^{-1}(\mathbf{y}) = \mathbf{z}$.

CNFs are rather designed to model complex probability distributions for low-dimensional data, what was confirmed in various applications including point cloud generation [27], future prediction [29] or probabilistic few-shot regression [23]. Compared to models like RealNVP [5] or Glow [11], they can be successfully applied to one-dimensional data and achieve better results for tabular datasets.

3 TreeFlow

Tree-based methods obtain superior results on tabular datasets and have developed multiple techniques to deal with categorical variables, null values, etc. but are limited to distributions with explicitly provided probability distribution functions, e.g., Gaussian. We want to overcome this limitation by introducing TreeFlow - method for uni- and multi-variate tree-based probabilistic regression with non-Gaussian and multi-modal target distributions. The main idea of the solution is to combine the benefits of using tree ensembles with the capabilities of modeling flexible probability distributions using conditional normalizing flows.

The architecture of TreeFlow is provided in fig. 1. The proposed model consists of three components: Tree-based Feature Extractor, Shallow Feature Extractor, and conditional CNF module. The role of the first component is to extract the vector of binary features from the structure of the tree-based ensemble model for a given input observation $\mathbf{x}$. The problem of extracting a unified vector from the tree-based ensemble model is non-trivial due to the complex structure and a large number of base learners. Motivated by the fact that crucial information extracted from input examples is stored in the leaves, we propose a binary occurrence representation that is the most lightweight approach assuming thousands of trees. Formally it could be written as $\mathbf{h}_{\boldsymbol{\psi}}(\mathbf{x}) = [\mathbf{o_1}, \ldots, \mathbf{o_T}]$, where $\mathbf{h}_{\boldsymbol{\psi}}(\mathbf{x})$ is Tree-based Feature Extractor with parameters $\boldsymbol{\psi}$ and $\mathbf{o_i} = [\mathbf{1}_{\{\mathbf{x} \in \mathcal{R}_{i,1}\}}, \ldots, \mathbf{1}_{\{\mathbf{x} \in \mathcal{R}_{i,K}\}}]$ where $\mathcal{R}_{i,k}$ is a region of kth leaf of ith decision tree in the forest structure.

The size of vector $\mathbf{o}$ is significantly larger than the size of the regression variable $\mathbf{y}$. If we deliver directly the large sparse binary vector as a CNF conditioning component: (i) the number of CNF parameters grows significantly, (ii) the conditioning component dominates training, and (iii) the ordinary differential equation (ODE) solver slows down significantly and behaves in an unstable way. Therefore, we use an additional Shallow Feature Extractor $\mathbf{k}_{\boldsymbol{\phi}}(\cdot)$, that is represented by a neural network responsible for mapping high-dimensional binary vector $\mathbf{o}$ returned by $\mathbf{h}_{\boldsymbol{\psi}}(\mathbf{x})$ to low-dimensional feature representation $\mathbf{w} = \mathbf{k}_{\boldsymbol{\phi}}(\mathbf{o})$. The low-dimensional representation $\mathbf{w}$ of the sparse embedding $\mathbf{o}$ is further passed to the conditional CNF module as a conditioning factor. We postulate to use the variant of the conditional flow-based model provided in [27, 23], where $\mathbf{w}$ is delivered to the function of dynamics, $\mathbf{g}_{\boldsymbol{\beta}}(\mathbf{z}(t), t, \mathbf{w})$. The transformation function is given by eq. (2) is represented as:

$$\mathbf{y} = \mathbf{f}_{\boldsymbol{\beta}}(\mathbf{z}, \mathbf{w}) = \mathbf{z} + \int_{t_0}^{t_1} \mathbf{g}_{\boldsymbol{\beta}}(\mathbf{z}(t), t, \mathbf{w}) dt. \tag{4}$$

The inverse form of the transformation $\mathbf{f}_{\boldsymbol{\beta}}(\cdot)$ given the same $\mathbf{w}$ in both directions is simply: $\mathbf{z} = \mathbf{f}_{\boldsymbol{\beta}}^{-1}(\mathbf{y}, \mathbf{w}) = \mathbf{y} - \int_{t_0}^{t_1} \mathbf{g}_{\boldsymbol{\beta}}(\mathbf{z}(t), t, \mathbf{w}) dt$. For a given model, we can easily calculate the log-probability of regression output $\mathbf{y}$, given the input $\mathbf{x}$ [9]:

$$\log p(\mathbf{y}|\mathbf{w}) = \log p(\mathbf{f}_{\boldsymbol{\beta}}^{-1}(\mathbf{y}, \mathbf{w})) - \int_{t_0}^{t_1} \operatorname{Tr}\left(\frac{\partial \mathbf{g}_{\boldsymbol{\beta}}(\mathbf{z}(t), t, \mathbf{w})}{\partial \mathbf{z}(t)}\right) dt, \tag{5}$$

where $\mathbf{w} = \mathbf{k}_{\boldsymbol{\phi}}(\mathbf{o})$, and $\mathbf{o} = \mathbf{h}_{\boldsymbol{\psi}}(\mathbf{x})$. With the model defined in the following way, we can easily calculate the exact value of log-probability for any possible regression outputs. We can also utilize the generative capabilities of the model by generating samples from a known prior $p(\mathbf{z})$ and transforming them into the space of regression outputs using the function given by eq. (4).

We aim at training the model by optimizing the NLL for a log probability defined by eq. (5) and the set of trainable parameters $\boldsymbol{\theta} = \{\boldsymbol{\psi}, \boldsymbol{\phi}, \boldsymbol{\beta}\}$. In the perfect scenario, we should optimize the entire

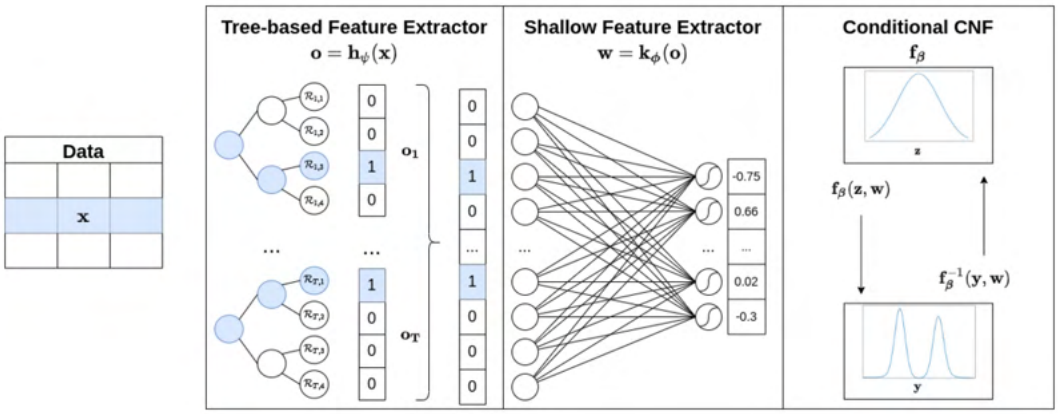


Figure 1. TreeFlow architecture. The proposed model consists of three components: Tree-based Feature Extractor, Shallow Feature Extractor, and Conditional CNF module. The role of the first component is to extract the vector of binary features from the structure of the tree-based ensemble model. The Shallow Feature Extractor is a neural network responsible for mapping high-dimensional binary vectors returned by the Tree-based Feature Extractor to low-dimensional feature space. The resulting vector is further passed to the conditional CNF module as a conditioning factor. The goal of the last component is to model complex probability distribution.

model in an end-to-end fashion, jointly updating the parameters of the Tree-based Feature Extractor ψ, Shallow Feature Extractor ϕ, and conditional CNF β. However, the Shallow Feature Extractor needs to have a constant size input which cannot be easily obtained from our Tree-based Feature Extractor as it learns iteratively. To overcome this limitation, we perform two-staged learning.

In the first stage, only the parameters of Tree-based Feature Extractor ψ^* are trained by optimizing the surrogate criterion specific to the type of tree-based architecture. In our work, we utilize the CatBoost model as it out-of-the-box supports categorical features and null values. Therefore, following [13] we train the Tree-based feature extractor by optimizing NLL loss for a standard Gaussian regression output given by eq. (1). For the multivariate case, we use the protocol from [20] and train the feature extractor by optimizing MultiRMSE.

Given the Tree-based Feature Extractor parameters, we train the remaining components of our model in an end-to-end fashion. Formally, given the estimated parameters ψ^* for $\mathbf{h}_{\psi}(\mathbf{x})$ we train the model by optimizing NLL with log-probability given by eq. (5) with respect to remaining parameters ϕ and β using the standard gradient-based approach.

The two-stage training has a couple of advantages compared to the end-to-end approach. First, any trained tree-based ensemble can be used as a feature extractor. Second, extracting the forest structure together with optimizing the parameters of the remaining components of the system in an end-to-end fashion is non-trivial and requires handcrafting the training procedure for a particular type of tree-based learner.

4 Related works

One of the best-known examples of gradient boosting methods is XGBoost [4] which iteratively combines weak regression trees to obtain accurate predictions. Further extensions to this method consist of LightGBM [10] and CatBoost [20] which introduce multiple novel techniques to obtain even better point estimates. Recently they have been extended to a probabilistic framework to model the whole probability distributions.

One such approach is NGBoost (Natural Gradient Boosting) [6] algorithm, which can model any probabilistic distribution with a defined probability density function, e.g., Univariate Gaussian, Exponential, or Laplace. It simultaneously estimates the distribution parameters by optimizing a proper scoring rule, e.g., negative log likelihood (NLL) or Continuous Ranked Probability Score (CRPS). The variant of NGBoost that utilizes Multivariate Gaussian to model multidimensional predictions was presented in [19]. RoNGBa [21] is an extension of NGBoost, which improves the performance of NGBoost via a better choice of hyperparameters. This framework has also been adapted to the CatBoost [13] with support to only univariate Gaussian distributions, but contrary to the NGBoost, the model outputs all distribution parameters from one model. There is also a group of approaches that were developed in parallel to NGBoost consisting of XGBoostLSS [15] and CatBoostLSS [16] which make a connection to well-established statistical framework Generalized Additive Models for Shape, Scale, and Location (GAMLSS) [26]. Like NGBoost, these models use one XGBoost or CatBoost model per parameter, but their training consists of two phases: independent model learning for each parameter and iterative parameter correction. One of the most recent approaches is Probabilistic Gradient Boosting Machine (PGBM) [25] which treats the leaf weights in each tree as random variables. This approach is capable to model different sets of posterior distributions but is limited to only distributions parameterized with location and scale parameters.

Besides the tree-based probabilistic models, several works investigate the problem of probabilistic regression. In [24] the authors model conditional density estimators for multivariate data with conditional sum-product networks that combines tree-based structures with deep models. In [7] the authors combine the transformer model with flows for density estimation. The flow models were also applied for future prediction problems in [29]. In [23] and [14] the authors propose to integrate flows with Gaussian Processes for probabilistic regression.

TreeFlow, to our best knowledge, is the first tree-based model for uni-, and multi-variate probabilistic regression, that is capable to model any distribution for regression outputs.

5 Experiments

This section evaluates our method on four different setups - univariate regression on synthetic data, univariate regression on mixed-type data, univariate regression on numerical data, and multivariate regression. Our goal is a quantitative and qualitative analysis of TreeFlow in comparison to the baseline models.

In all experiments, we measure target distribution fit using the negative log likelihood metric in the quantitative part. It is a natural choice as we expect to deal with heavy-tailed and multimodal distributions. Additionally, we calculate the CRPS metric which is defined as the mean squared difference between the forecasted probabilities and the actual outcomes, over all possible thresholds. It is not the best-suited metric for multimodal distributions, although it is often used for probabilistic forecasting and we would like to understand differences between TreeFlow and baselines. Moreover, we investigate point estimates that are usually necessary from the application point of view. For that purpose, we use the standard Root Mean Squared Error (RMSE) metric and introduce Root Mean Squared Error at K (RMSE@K) metric. The latter is a version of the RMSE metric that is adjusted for multimodal distributions and takes into account multiple predictions from the model. More details and justifications are provided in the Appendix in sec. A.1. In the qualitative part, we analyze and discuss the characteristics of obtained probability distributions. Finally, we perform the ablation study whose goal was to justify the design choices. The results of this part are presented in the Appendix in sec. D.

5.1 Univariate regression on synthetic data

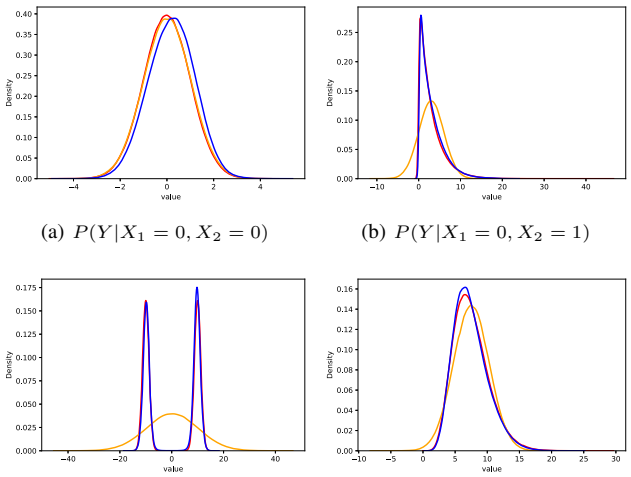
This experiment is one of the motivating examples. Here, we want to evaluate the capabilities of TreeFlow to model data when the true probability distribution is known.

Dataset and methodology We have created a dataset with two conditioning binary variables. For each possible combination of features, we have proposed different continuous distributions: Normal, Exponential, Mixture of Gaussians, and Gamma (see fig. 2 and details in Appendix, in sec. B). After that, we trained TreeFlow and CatBoost models. Finally, we calculated negative log likelihood and visualized the obtained probability distributions.

Results After five repetitions of the experiment, we obtained negative log likelihood for CatBoost equal 2.52 ± 0.01, and for TreeFlow equal 2.02 ± 0.00. We can observe, that our method effortlessly obtained better results and, contrary to CatBoost, it was able to correctly model all probability distributions (see fig. 2). This is due to its flexibility in modeling probability distributions resulting from the usage of the CNF component.

5.2 Univariate regression on mixed-type data

Our goal is to evaluate and verify our approach to univariate regression problems with mixed-type data. This experiment is the main motivating example of this paper, as tree-based methods cannot model non-gaussian target distributions, and normalizing flows cannot deal with categorical variables without any additional data preparation step.



(a) $P(Y|X_1 = 0, X_2 = 0)$ (b) $P(Y|X_1 = 0, X_2 = 1)$

(c) $P(Y|X_1 = 1, X_2 = 0)$ (d) $P(Y|X_1 = 1, X_2 = 1)$

Figure 2. Comparison of the estimated probability distributions for univariate regression on synthetic data experiment. We can observe that contrary to CatBoost, TreeFlow was able to correctly model all underlying true probability distributions. Legend: Red - True probability distribution; Blue - TreeFlow; Orange - CatBoost.

Datasets and methodology To the best of our knowledge, there is no established standard benchmark for regression problems with mixed-type datasets. Thus, we propose seven datasets from the well-known data platform - Kaggle. They have various numbers of samples ranging from a few thousand to a hundred thousand, a different number of categorical and numerical variables. All details of the datasets can be found in tab. 6.

In terms of the methodology, we follow the standard $80\%/20\%$ training/testing holdout split. We also split the training dataset to train and validation datasets in the same proportion for the best epoch/iteration selection purposes. All experiments are run 5 times and results are averaged.

For obtaining point estimates from TreeFlow we analyze three approaches: (i) Samples averaging (Avg) - the simple average of samples, (ii) RMSE@1 - usage of the most probable sample, (iii) RMSE@2 - usage of the two most probable samples. Finally, we provide ablation studies regarding the design of the Tree-based Feature Extractor and the Shallow Feature Extractor (see Appendix sec. D).

Baselines Currently, the only approach to work with such problems is a CatBoost which deals out-of-the-box with mixed-type datasets and support modeling target variable with Gaussian distribution. Additionally, we evaluate PGBM with one hot encoding for categorical variables as the representative method for standard tree methods without support for categorical variables. Moreover, this method is also capable of utilizing various parametric distributions. We perform the evaluation on both probabilistic (NLL, CRPS) and deterministic (RMSE / RMSE@K) metrics.

Results The results of the conducted experiments for probabilistic metrics are provided in tab. 1 and for deterministic metrics in tab. 2. Our method obtains better negative log likelihood scores for most of the datasets and for most of them better CRPS values than reference methods. Furthermore, for the majority of datasets, there

Table 1. Comparison of TreeFlow with existing methods in terms of negative log likelihood (NLL) and Continuous Ranked Probability Score (CRPS) on univariate regression problems with mixed-type data. Our method outperformed both CatBoost and PGBM approaches on most of the datasets thanks to its flexibility in modeling non-gaussian distributions. One Hot Encoding was used for categorical variables for PGBM. Extended information about datasets is provided in tab. 6.

Dataset	NLL			CRPS		
	CatBoost	PGBM	TreeFlow	CatBoost	PGBM	TreeFlow
Avocado	-0.40 ± 0.01	-0.45 ± 0.01	**-0.47 ± 0.03**	0.0992 ± 0.0018	0.0870 ± 0.0013	**0.0854 ± 0.0024**
BigMart	-0.05 ± 0.02	**-0.10 ± 0.02**	-0.08 ± 0.02	0.1270 ± 0.0021	**0.1259 ± 0.0023**	0.1294 ± 0.0027
Diamonds	-1.80 ± 0.02	-1.41 ± 0.76	**-1.94 ± 0.03**	0.0222 ± 0.0002	0.0447 ± 0.0474	**0.0210 ± 0.0005**
Diamonds 2	-1.89 ± 0.02	-1.24 ± 0.83	**-2.14 ± 0.05**	0.0217 ± 0.0002	0.0461 ± 0.0504	**0.0197 ± 0.0005**
Laptop	-0.89 ± 0.08	**-0.97 ± 0.09**	-0.74 ± 0.13	0.0572 ± 0.0049	**0.0474 ± 0.0034**	0.0563 ± 0.0043
Pak Wheel	-1.40 ± 0.05	-0.53 ± 0.02	**-1.60 ± 0.03**	0.0362 ± 0.0006	0.0813 ± 0.0009	**0.0327 ± 0.0007**
Sydney	-0.54 ± 0.04	0.20 ± 1.02	**-0.66 ± 0.01**	0.0726 ± 0.0011	0.2383 ± 0.2646	**0.0721 ± 0.0008**

Table 2. Comparison of TreeFlow with existing methods in terms of root mean squared error (RMSE) on univariate regression problems with mixed-type data. TreeFlow in approach @2 significantly outperforms other baseline methods by taking advantage of multimodal distribution modeling property.

Dataset	RMSE				
	CatBoost	PGBM	TreeFlow(Avg)	TreeFlow(@1)	TreeFlow(@2)
Avocado	0.1939 ± 0.0043	**0.1624 ± 0.0024**	0.1676 ± 0.0058	0.1769 ± 0.0087	0.1713 ± 0.0066
BigMart	0.2284 ± 0.0039	**0.2274 ± 0.0040**	0.2335 ± 0.0045	0.2514 ± 0.0087	0.2480 ± 0.0083
Diamonds	0.0419 ± 0.0007	0.0403 ± 0.0006	0.0407 ± 0.0009	0.0445 ± 0.0015	**0.0343 ± 0.0017**
Diamonds 2	0.0421 ± 0.0006	0.0492 ± 0.0010	0.0398 ± 0.0006	0.0460 ± 0.0014	**0.0364 ± 0.0004**
Laptop	0.1028 ± 0.0092	**0.0848 ± 0.0063**	0.1014 ± 0.0082	0.1015 ± 0.0076	0.0958 ± 0.0058
Pak Wheel	0.0783 ± 0.0009	0.1630 ± 0.0018	0.0729 ± 0.0018	0.0796 ± 0.0021	**0.0654 ± 0.0047**
Sydney	0.1528 ± 0.0057	0.1561 ± 0.0047	0.1518 ± 0.0051	0.1721 ± 0.0041	**0.1361 ± 0.0066**

is a substantial improvement in the results. In terms of point estimates, TreeFlow in @2 approach obtains superior results in most of the datasets by the ability to provide multiple predictions for a particular sample that could be modeled by multimodal distributions. The detailed discussion regarding differences between point estimates for TreeFlow is provided in the Appendix in sec. A.1 Moreover, we investigated that the target distributions provided by TreeFlow had more realistic properties such as a heavy tail, multimodality, or does not provide any probability mass for impossible values, e.g., negative values when modeling price as a target variable. The latter example is presented in fig. 3. We analyzed estimated probability density functions for the Wine Reviews datasets for CatBoost and TreeFlow. Both methods predicted similar values for the PDF function; however, only TreeFlow was able to model heavy-tailed distribution and recognize that negative price values are highly unlikely.

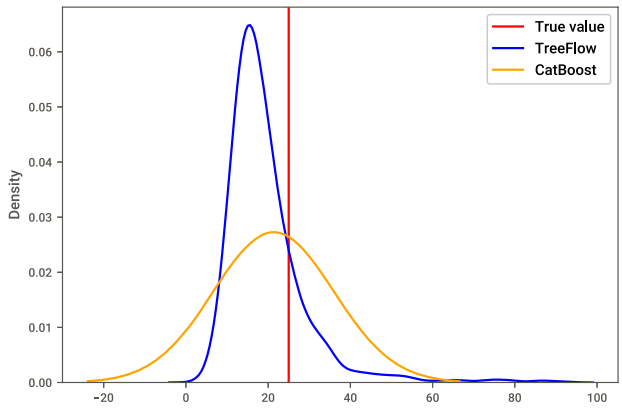


Figure 3. Estimated probability density functions for the Wine Reviews datasets from the univariate regression on mixed-type data experiment. Both methods predicted similar values for the PDF function; however, the properties of the obtained distributions are entirely different. Contrary to the CatBoost approach, TreeFlow was able to model heavy-tailed distribution and recognize that negative price values are highly unlikely.

5.3 Univariate regression on numerical data

We focus on univariate regression problems with only numerical variables in this setup. We aim to evaluate our method on standard probabilistic regression benchmarks in both probabilistic and deterministic approach. Finally, we investigate the properties of the obtained target distributions.

Datasets and methodology We use established in the reference methods [6, 13] probabilistic regression benchmark with the exclusion of the Boston dataset due to ethical issues. It contains nine varying-size datasets from the UCI Machine Learning Repository. We follow the same protocol as used in the reference papers. We create 20 random folds for all datasets except Protein (5 folds) and Year MSD (1 fold). We keep 10% of samples as a test set for each of these folds, and the remaining 90% of data we split into an 80%/20% train/validation for the best epoch selection purposes.

Baselines We selected four tree-based baseline models: NGBoost [6], RoNGBa [21], CatBoost [13], PGBM [25], and one non tree-based method - Deep Ensemble [12] which was used in [13] as a reference method.

Results The quantitative results for negative log likelihood (NLL) are presented in tab. 3 and for RMSE in tab. 4. In terms of the probabilistic metric, our approach outperforms baseline methods on three

Table 3. Comparison of TreeFlow with existing methods in terms of negative log likelihood (NLL) on univariate regression problems with numerical data. Our method outperformed the other approaches on three datasets and obtained competitive results on others. The superior results were obtained thanks to our method's ability to model multimodal distributions.

Dataset	Deep. Ens.	CatBoost	NGBoost	RoNGBa	PGBM	TreeFlow
Concrete	3.06 ± 0.18	3.06 ± 0.13	3.04 ± 0.17	2.94 ± 0.18	**2.75 ± 0.21**	3.02 ± 0.15
Energy	1.38 ± 0.22	1.24 ± 1.28	0.60 ± 0.45	**0.37 ± 0.28**	1.74 ± 0.04	0.85 ± 0.35
Kin8nm	**-1.20 ± 0.02**	- 0.63 ± 0.02	-0.49 ± 0.02	-0.60 ± 0.03	-0.54 ± 0.04	-1.03 ± 0.06
Naval	**-5.63 ± 0.05**	-5.39 ± 0.04	-5.34 ± 0.04	-5.49 ± 0.04	-3.44 ± 0.04	-5.54 ± 0.16
Power	2.79 ± 0.04	2.72 ± 0.12	2.79 ± 0.11	2.65 ± 0.08	**2.60 ± 0.02**	2.65 ± 0.06
Protein	2.83 ± 0.02	2.73 ± 0.07	2.81 ± 0.03	2.76 ± 0.03	2.79 ± 0.01	**2.02 ± 0.02**
Wine	0.94 ± 0.12	0.93 ± 0.08	0.91 ± 0.06	0.91 ± 0.08	0.97 ± 0.20	**-0.56 ± 0.62**
Yacht	1.18 ± 0.21	0.41 ± 0.39	0.20 ± 0.26	1.03 ± 0.44	**0.05 ± 0.28**	0.72 ± 0.40
Year MSD	3.35 ± NA	3.43 ± NA	3.43 ± NA	3.46 ± NA	3.61 ± NA	**3.27 ± NA**

Table 4. Comparison of TreeFlow with existing methods in terms of Root Mean Squared Error (RMSE) on univariate regression problems with numerical data.

Dataset	Deep. Ens.	CatBoost	NGBoost	RoNGBa	PGBM	TreeFlow (Avg)	TreeFlow (@1)	TreeFlow (@2)
Concrete	6.03 ± 0.58	5.21 ± 0.53	5.06 ± 0.61	4.71 ± 0.61	**3.97 ± 0.76**	5.33 ± 0.65	5.41 ± 0.72	5.41 ± 0.71
Energy	2.09 ± 0.29	0.57 ± 0.06	0.46 ± 0.06	**0.35 ±0.07**	**0.35 ± 0.06**	0.64 ± 0.11	0.66 ± 0.13	0.65 ± 0.12
Kin8nm	**0.09 ± 0.00**	0.14 ± 0.00	0.16 ± 0.00	0.14 ± 0.00	0.13 ± 0.01	**0.09 ± 0.00**	0.10 ± 0.01	0.10 ± 0.01
Naval	**0.00 ± 0.00**	**0.00 ± 0.00**	**0.00 ± 0.00**	**0.00 ± 0.00**	**0.00 ± 0.00**	**0.00 ± 0.00**	**0.00 ± 0.00**	**0.00 ± 0.00**
Power	4.11 ± 0.17	3.55 ± 0.27	3.70 ± 0.22	3.47 ± 0.19	**3.35 ± 0.15**	3.71 ± 0.26	3.79 ± 0.26	3.79 ± 0.25
Protein	4.71 ± 0.06	3.92 ± 0.08	4.33 ± 0.03	4.21 ± 0.06	3.98 ± 0.06	4.00 ± 0.27	4.79 ± 0.52	**3.01 ± 0.06**
Wine	0.64 ± 0.04	0.63 ± 0.04	0.62 ± 0.04	0.62 ± 0.05	0.60 ± 0.05	0.66 ± 0.05	0.73 ± 0.06	**0.41 ± 0.09**
Yacht	1.58 ± 0.48	0.82 ± 0.40	**0.50 ± 0.20**	0.90 ± 0.35	0.63 ± 0.21	0.75 ± 0.26	0.75 ± 0.25	0.75 ± 0.26
Year MSD	8.89 ± NA	8.99 ± NA	8.94 ± NA	9.14 ± NA	9.09 ± NA	9.29 ± nan	10.97 ± nan	**8.64 ± NA**

datasets: Protein, Wine, Year MSD, and obtains competitive results on others. For deterministic metrics, we obtain SOTA results for the same three datasets, and for two (kin8nm and naval) we achieve the same results as the current best methods.

To understand the results, we have investigated target distributions. We have compared them with the CatBoost model with default hyperparameters and presented them in fig. 4.

The first subfigure presents results for the Protein dataset. TreeFlow method has discovered that the underlying target distribution has a bimodal character and was able to correctly estimate the high value of the probability density function for the true value. In contrast, the Gaussian-based method did not have such an ability and incorrectly estimated the center of probability mass between two modes. The second subfigure is a representative of naturally occurring integer value datasets: Wine Quality and Year MSD. In this example, our method proposes a multimodal distribution consisting of Gaussian-like and heavy-tailed distributions. Such estimation gives us very rich information for the decision-making process compared to the Gaussian-based approach, which only estimated values around the highest mode and completely ignored information about a minor mode around 5 and a heavy tail for values 8 and 9. The last subfigure is a representative example of the rest of the datasets for which our method obtained similar results to baselines. Both methods proposed Gaussian distribution as a target distribution and assuming that this is a correct target distribution, there is no possibility of obtaining significantly better results.

The above-mentioned analysis also explains the results of the deterministic metrics. We incorporated into the decision-making process additional information about the second modality and it resulted in significant gains in prediction accuracy. To the best of our knowledge, it is the first time when these properties were noticed and exploited.

5.4 *Multivariate regression*

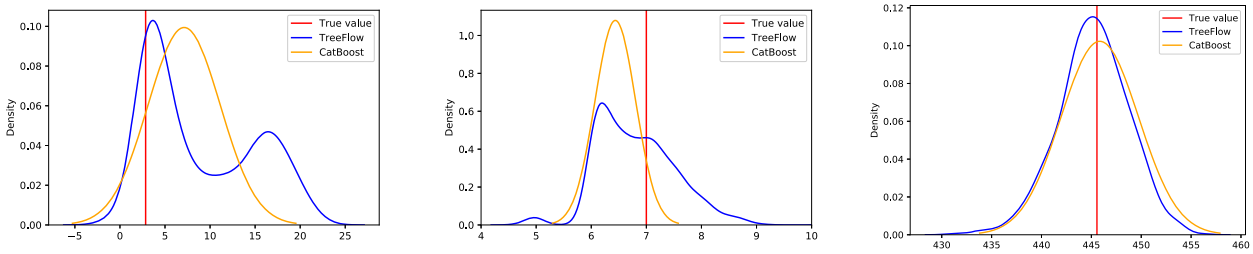
In the last setup, we focus on multivariate regression problems. Our goal is to quantitatively evaluate our method on datasets with various target dimensionality and examine the properties of obtained distributions.

Datasets and methodology Currently, the only tree-based probabilistic multivariate regression problem was approached by [19] which proposes a task of two-dimensional oceanographic velocities prediction [18]. Moreover, we evaluate our method on five more datasets with a broad range of target and feature dimensionality introduced in [28].

For both groups of datasets, we follow the proposed for these datasets experiment methodology. For the Oceanographic dataset, it is the same protocol as in the univariate regression on numerical data experiment. For the second group, it is a standard training/testing holdout split similar to the univariate regression on the mixed-type data experiment. The exact number of samples is provided in the tab. 5.

Baselines For this setup, we selected two baseline models. The first approach uses NGBoost, which assumes Multivariate Gaussian distribution and models correlation between target variables. The second approach also uses NGBoost, but the separate model models each target dimension; thus, it assumes independence between targets. We do not consider other Independent Gaussian approaches as they similarly model target distribution.

Results The results of the experiments are provided in tab. 5. Our method outperforms baselines by a large margin on three datasets. In contrast to NGBoost-based methods, TreeFlow was able to capture non-gaussianity in the target distributions. It can be evident on Parkinsons and US Flight datasets where differences between Independent NGBoost and Multivariate NGBoost were significant. They were probably caused by the ability to model the correlation between target variables and TreeFlow utilized its flexibility to obtain even better results. The other situation is for the Oceanographic dataset, where all results are close. Here, probably true target distribution is similar to the Independent Gaussian distribution; thus, NGBoost and TreeFlow can not achieve better results. In the last dataset - Energy,



(a) Multimodal distribution (b) Multimodal, heavy-tailed dist. (c) Gaussian distribution

Figure 4. Estimated probability density functions for three datasets (Protein, Wine Quality, Power Plant) from the univariate regression on numerical data experiment. Depending on the dataset, TreeFlow is able to model distributions with various properties.

Table 5. Comparison of TreeFlow with existing methods in terms of negative log likelihood on multivariate regression problems. Our method obtains SOTA results on four out of the six datasets thanks to its flexibility in modeling complex distributions. Baseline results for the Oceanographic are taken from the reference paper.

DATASET	IND NGBOOST	NGBOOST	TREEFLOW
PARKINSONS	6.86	5.85	**5.26**
SCM20D	94.40	94.81	**93.41**
WINDTURBINE	-0.65	-0.67	**-2.57**
ENERGY	**166.90**	175.80	180.00
USFLIGHT	9.56	8.57	**7.49**
OCEANOGRAPHIC	7.74±0.02	**7.73±0.02**	7.84±0.01

the best performing model was Independent NGBoost. We suspect that the high dimensionality of the target distribution was too hard to learn for both NGBoost and TreeFlow methods.

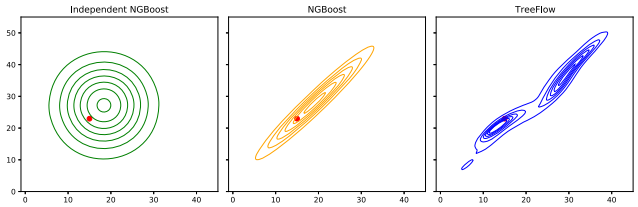


Figure 5. Estimated probability density functions for the Parkinsons datasets from the multivariate regression experiment. TreeFlow is the most flexible method which enables target distribution to correlate between dimensions and has multiple modes.

Moreover, we investigated target distributions for the Parkinsons dataset. The results of all methods are presented in fig. 5. We can easily observe how consecutive methods allow for more flexible distributions. Multivariate NGBoost enables correlation between target variables, while TreeFlow adds multimodality property.

6 Conclusions

In this work, we proposed a novel tree-based approach for probabilistic regression. Our method combines the benefits of ensemble decision trees with the capabilities of flow-based models in modeling complex non-Gaussian, multimodal distributions. We evaluated our approach using four experimental settings for both probabilistic and deterministic metrics and achieve SOTA or comparable results on most of them. We also illustrate some properties of TreeFlow that show the benefits of our approach compared to reference baselines.

Limitations The main trade-off introduced by our method is between computational time and the flexibility of the target distribution. Resource-demanding CNF component limits the scalability of our method, but despite this, we were able to deal with datasets of up to half a million observations. Additionally, our method has multiple hyperparameters, which may be challenging to tune in some cases, but we hope that a broad range of experiments provides good intuitions for end users (see sec. C). Lastly, TreeFlow performs two-staged learning, which might sometimes lead to sub-optimal results. Even so, our method outperforms current baselines, and we hope that TreeFlow will serve as a strong starting point for future end-to-end approaches.

Broader Impact The tree-based models are widely applied in research and industry and often achieve SOTA results. TreeFlow can be seen as an extension of such models, and all ethical considerations, both positive and negative, regarding regression problems apply to our work. However, as we consider target distribution more complex than parametric, our method can better assess uncertainty in the decision-making process or provide realistic probability distributions (see examples in fig. 3, 4, 5). Such properties might be crucial, for example, in medicine or finance applications, and have a largely positive societal impact.

Acknowledgements

The work conducted by Patryk Wielopolski and Maciej Zieba was supported by the National Centre of Science (Poland) Grant No. 2021/43/B/ST6/02853.

Appendix and Code

The code and appendix are available under the GitHub repository: https://github.com/pfilo8/TreeFlow.

References

[1] Leo Breiman, 'Random forests', *Machine Learning*, **45**(1), 5–32, (2001).

[2] Leo Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*, Wadsworth, 1984.

[3] Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud, 'Neural ordinary differential equations', in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, eds., Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, pp. 6572–6583, (2018).

[4] Tianqi Chen and Carlos Guestrin, 'XGBoost: A Scalable Tree Boosting System', in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pp. 785–794. ACM, (2016).

[5] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio, 'Density estimation using Real NVP', in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, (2017).

[6] Tony Duan, Avati Anand, Daisy Yi Ding, Khanh K. Thai, Sanjay Basu, Andrew Y. Ng, and Alejandro Schuler, 'NGBoost: Natural Gradient Boosting for Probabilistic Prediction', in *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 2690–2700. PMLR, (2020).

[7] Rasool Fakoor, Pratik Chaudhari, Jonas Mueller, and Alexander J Smola, 'Trade: Transformers for density estimation', *arXiv preprint arXiv:2004.02441*, (2020).

[8] Jerome H. Friedman, 'Greedy function approximation: A gradient boosting machine.', *Annals of Statistics*, **29**, 1189–1232, (2001).

[9] Will Grathwohl, Ricky T. Q. Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud, 'FFJORD: free-form continuous dynamics for scalable reversible generative models', in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, (2019).

[10] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu, 'LightGBM: A Highly Efficient Gradient Boosting Decision Tree', in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 3146–3154, (2017).

[11] Diederik P. Kingma and Prafulla Dhariwal, 'Glow: Generative flow with invertible 1x1 convolutions', in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 10236–10245, (2018).

[12] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell, 'Simple and scalable predictive uncertainty estimation using deep ensembles', in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 6402–6413, (2017).

[13] Andrey Malinin, Liudmila Prokhorenkova, and Aleksei Ustimenko, 'Uncertainty in Gradient Boosting via Ensembles', in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, (2021).

[14] Juan Maroñas, Oliver Hamelijnck, Jeremias Knoblauch, and Theodoros Damoulas, 'Transforming gaussian processes with normalizing flows', in *International Conference on Artificial Intelligence and Statistics*, pp. 1081–1089. PMLR, (2021).

[15] Alexander März, 'XGBoostLSS - An extension of XGBoost to probabilistic forecasting', *CoRR*, **abs/1907.03178**, (2019).

[16] Alexander März, 'CatBoostLSS - An extension of CatBoost to probabilistic forecasting', *CoRR*, **abs/2001.02121**, (2020).

[17] Kevin P. Murphy, *Machine learning - a probabilistic perspective*, Adaptive computation and machine learning series, MIT Press, 2012.

[18] Michael O'Malley, 'North Atlantic Ocean Drifter Dataset for Multivariate Probabilistic Regression with Natural Gradient Boosting'. Zenodo, (2021).

[19] Michael O'Malley, Adam M. Sykulski, Rick Lumpkin, and Alejandro Schuler, 'Multivariate Probabilistic Regression with Natural Gradient Boosting', *CoRR*, **abs/2106.03823**, (2021).

[20] Liudmila Ostroumova Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin, 'CatBoost: unbiased boosting with categorical features', in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 6639–6649, (2018).

[21] Liliang Ren, Gen Sun, and Jiaman Wu, 'RoNGBa: A Robustly Optimized Natural Gradient Boosting Training Approach with Leaf Number Clipping', *CoRR*, **abs/1912.02338**, (2019).

[22] Danilo Jimenez Rezende and Shakir Mohamed, 'Variational Inference with Normalizing Flows', in *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pp. 1530–1538. JMLR.org, (2015).

[23] Marcin Sendera, Jacek Tabor, Aleksandra Nowak, Andrzej Bedychaj, Massimiliano Patacchiola, Tomasz Trzcinski, Przemysław Spurek, and Maciej Zieba, 'Non-Gaussian Gaussian Processes for Few-Shot Regression', in *NeurIPS*, (2021).

[24] Xiaoting Shao, Alejandro Molina, Antonio Vergari, Karl Stelzner, Robert Peharz, Thomas Liebig, and Kristian Kersting, 'Conditional sum-product networks: Imposing structure on deep probabilistic architectures', in *International Conference on Probabilistic Graphical Models*, pp. 401–412. PMLR, (2020).

[25] Olivier Sprangers, Sebastian Schelter, and Maarten de Rijke, 'Probabilistic gradient boosting machines for large-scale probabilistic regression', in *KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14-18, 2021*, pp. 1510–1520. ACM, (2021).

[26] D Mikis Stasinopoulos, Robert A Rigby, et al., 'Generalized additive models for location scale and shape (GAMLSS) in R', *Journal of Statistical Software*, (2007).

[27] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge J. Belongie, and Bharath Hariharan, 'PointFlow: 3D Point Cloud Generation With Continuous Normalizing Flows', in *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pp. 4540–4549. IEEE, (2019).

[28] Zhongjie Yu, Mingye Zhu, Martin Trapp, Arseny Skryagin, and Kristian Kersting, 'Leveraging probabilistic circuits for nonparametric multi-output regression', in *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence, UAI 2021, Virtual Event, 27-30 July 2021*, volume 161 of *Proceedings of Machine Learning Research*, pp. 2008–2018. AUAI Press, (2021).

[29] Maciej Zieba, Marcin Przewieźlikowski, Marek Śmieja, Jacek Tabor, Tomasz Trzcinski, and Przemysław Spurek, 'RegFlow: Probabilistic Flow-based Regression for Future Prediction', *CoRR*, **abs/2011.14620**, (2020).

entropy

MDPI

Article

NodeFlow: Towards End-to-End Flexible Probabilistic Regression on Tabular Data

Patryk Wielopolski [1,*], Oleksii Furman [1] and Maciej Zięba [1,2]

[1] Department of Artificial Intelligence, Wrocław University of Science and Technology, 50-370 Wrocław, Poland
[2] Tooploox Ltd., 53-601 Wrocław, Poland
* Correspondence: patryk.wielopolski@pwr.edu.pl

Abstract: We introduce NodeFlow, a flexible framework for probabilistic regression on tabular data that combines Neural Oblivious Decision Ensembles (NODEs) and Conditional Continuous Normalizing Flows (CNFs). It offers improved modeling capabilities for arbitrary probabilistic distributions, addressing the limitations of traditional parametric approaches. In NodeFlow, the NODE captures complex relationships in tabular data through a tree-like structure, while the conditional CNF utilizes the NODE's output space as a conditioning factor. The training process of NodeFlow employs standard gradient-based learning, facilitating the end-to-end optimization of the NODEs and CNF-based density estimation. This approach ensures outstanding performance, ease of implementation, and scalability, making NodeFlow an appealing choice for practitioners and researchers. Comprehensive assessments on benchmark datasets underscore NodeFlow's efficacy, revealing its achievement of state-of-the-art outcomes in multivariate probabilistic regression setup and its strong performance in univariate regression tasks. Furthermore, ablation studies are conducted to justify the design choices of NodeFlow. In conclusion, NodeFlow's end-to-end training process and strong performance make it a compelling solution for practitioners and researchers. Additionally, it opens new avenues for research and application in the field of probabilistic regression on tabular data.

Keywords: probabilistic regression; tabular data; normalizing flows; decision tree ensembles; neural decision tree

Citation: Wielopolski, P.; Furman, O.; Zięba, M. NodeFlow: Towards End-to-End Flexible Probabilistic Regression on Tabular Data. *Entropy* **2024**, *26*, 593. https://doi.org/10.3390/e26070593

Academic Editor: Friedhelm Schwenker

Received: 28 May 2024
Revised: 26 June 2024
Accepted: 4 July 2024
Published: 11 July 2024

1. Introduction

Tabular regression involves predicting a continuous target variable based on structured data arranged in a tabular format. It is a vital task in machine learning with applications in various domains, including finance, healthcare, and marketing. In these domains, making reliable and informed decisions is of utmost importance due to potential consequences or impacts and requires not only accurate predictions but also robust uncertainty quantification. These kinds of properties can be obtained by the usage of probabilistic methods that go beyond point estimation by modeling the entire conditional distribution. This approach offers several advantages, including the ability to quantify uncertainty, capture complex data distributions, and provide a more comprehensive understanding of the data.

Regarding deterministic tabular regression, there have been two distinct paths of research in the field of regression on tabular data without any clear conclusion of the best approach to the problem [1,2]. The first path focuses on gradient-boosted trees, exemplified by popular approaches such as XGBoost [3], CatBoost [4], and LightGBM [5]. These methods have demonstrated remarkable performance in point estimation tasks, leveraging ensemble techniques to capture complex relationships in the data. The second research path explores deep learning techniques for regression on tabular data with models such as NODE [6], TabNet [7], or FT-Transfomer [8]. These methods, with their ability to capture intricate patterns and relationships, have shown promise in surpassing the performance of gradient-boosted trees. They offer flexibility in handling various data types, including

categorical variables, and can capture complex interactions among features. However, challenges specific to tabular data, such as feature interactions and interpretability, continue to be active research areas.

In the context of probabilistic tabular regression, recent research predominantly centers on expanding tree-based methods. The development of the new methods has resulted in models such as NGBoost [9], PGBM [10], and a probabilistic extension of CatBoost [11]. However, these methods are predominantly based on parametric distributions, with CatBoost limited to modeling only Gaussian distributions. As a result, a pressing need remains for more flexible approaches that can accurately capture a broader range of complex data distributions encountered in practical scenarios. The recent work on TreeFlow [12] showed that combining tree-based methods with normalizing flows can improve the modeling capabilities; however, a lack of end-to-end optimization might lead to suboptimal results.

To overcome the limitations associated with the absence of end-to-end optimization, we propose NodeFlow, a novel framework for flexible probabilistic regression on tabular data. NodeFlow combines the advantages of tree-based structures, deep learning approaches, and normalizing flows to provide an accurate probabilistic regression approach that can be learned end to end. By combining Neural Oblivious Decision Ensembles (NODEs) and Conditional Continuous Normalizing Flows (CNFs), NodeFlow offers a unique solution that enables the modeling of complex data distributions encountered in probabilistic tasks. Through extensive evaluations and comparative studies on benchmark datasets, we demonstrate the effectiveness of NodeFlow in capturing the underlying data distributions and providing state-of-the-art results for multivariate probabilistic regression problems and competitive performance in univariate regression tasks.

Concluding, our contributions are as follows:

- We introduce NodeFlow, to the best of our knowledge, the first framework to apply an end-to-end, tree-structured deep learning model for probabilistic regression on tabular data;
- We demonstrate NodeFlow's superior performance in multivariate probabilistic regression and competitive results in univariate tasks on benchmark datasets, establishing its effectiveness;
- We conduct a focused ablation study, hyperparameter sensitivity analysis, and computational efficiency assessment, validating NodeFlow's design and scalability.

2. Literature Review

2.1. Tree-Based Regression on Tabular Data

Standard tree-based regression approaches, including XGBoost [3], CatBoost [4], and LightGBM [5], have emerged as state-of-the-art methods for modeling tabular data in regression problems. These frameworks leverage ensemble techniques and advanced optimizations to achieve remarkable performance in various domains. XGBoost is an optimized gradient-boosting framework that combines decision trees to capture complex relationships in tabular data. CatBoost incorporates novel techniques to handle categorical features effectively, while LightGBM utilizes tree-based learning algorithms and efficient data processing strategies. Their widespread adoption and success in diverse applications highlight their effectiveness and prominence in the field of tabular regression modeling, enabling accurate point estimation and capturing intricate patterns within the data.

2.2. Tree-Based Probabilistic Regression on Tabular Data

In recent years, several approaches have been developed for probabilistic regression on tabular data, including NGBoost [9], CatBoost with univariate Gaussian support [11], and the Probabilistic Gradient Boosting Machine (PGBM) [10], each offering unique methods to model probabilistic distributions and improve regression performance. NGBoost is a versatile algorithm that can model various probabilistic distributions using a defined probability density function. It estimates distribution parameters by optimizing scoring rules such as the negative log-likelihood (NLL) or Continuous Ranked Probability Score (CRPS).

RoNGBa [13] is an NGBoost extension that enhances performance through improved hyperparameter selection. CatBoost, a gradient-boosting framework, has also been adapted to probabilistic regression but supports only univariate Gaussian distributions. PGBM treats leaf weights as random variables and can model different posterior distributions, albeit limited to location and scale parameters.

2.3. Deep Learning Regression on Tabular Data

In recent years, deep neural networks have achieved remarkable success in handling unstructured data, but their effectiveness in dealing with tabular data remains inconclusive. Several research papers, including [6–8,14,15], have introduced new deep learning regression methods that demonstrate superiority over tree-based methods. However, recent surveys have produced conflicting results on this topic. Notably, Borisov et al. [1] conducted a study comparing deep models to traditional machine learning methods on selected datasets. They found that deep models consistently outperformed traditional methods, but no single deep model universally outperformed all others. These findings highlight the nuanced performance of deep learning models on tabular data. Additionally, recent benchmarks conducted by Grinsztajn et al. [2] compared tree-based models and deep learning methods, specifically on tabular data. The benchmarks revealed that tree-based models such as XGBoost and random forests remain state-of-the-art for medium-sized datasets (with fewer than 10,000 samples). Notably, even without considering their superior processing speed, tree-based models maintained a competitive edge over deep learning approaches.

Neural Oblivious Decision Ensembles (NODEs), introduced by [6], are a deep learning architecture that extends ensembles of oblivious decision trees. It combines end-to-end gradient-based optimization with multi-layer hierarchical representation learning. DNF-Net, proposed by [7], is a neural architecture incorporating a disjunctive normal form (DNF) structure, allowing efficient and interpretable feature selection. It promotes localized decisions over small feature subsets, enhancing interpretability and mitigating overfitting. TabNet [14] is a deep learning architecture specifically tailored for tabular data. It processes raw tabular data without preprocessing, facilitating seamless integration into end-to-end learning. Sequential attention mechanisms identify crucial features at each decision step, enhancing interpretability and learning efficiency. TabNet also provides interpretable feature attributions and insights into the model's global behavior. Gorishniy et al. [8] proposed FT-Transformer, a modified version of the Transformer architecture designed for tabular data. FT-Transformer incorporates both categorical and continuous features, employs self-attention mechanisms to capture feature relationships, and integrates residual connections akin to ResNet. In addition to these approaches, SAINT (Self-Attention and Intersample Attention Transformer) [15] is a hybrid deep learning approach designed to solve tabular data problems. SAINT integrates attention over both rows and columns, an enhanced embedding method, and a contrastive self-supervised pre-training technique.

2.4. Deep Learning Probabilistic Regression on Tabular Data

Recently, there has been limited research on Probabilistic Deep Learning for tabular data. One notable method in this area is Deep Ensemble [16], which involves training an ensemble of neural networks using negative log-likelihood optimization with a Gaussian distribution as the modeling choice. The authors also incorporate adversarial training to produce smoother predictive estimates. Another approach, MC-Dropout [17], extends the use of dropout to capture model uncertainty during inference. By sampling multiple dropout masks during inference and averaging the predictions over these masks, an ensemble of models is created to capture model uncertainty collectively. Probabilistic Backpropagation [18] treats the neural network weights as random variables and approximates their posterior distribution using a factorized Gaussian distribution. This approximation is updated iteratively utilizing a combination of variational inference and stochastic gradient descent. More recently, TreeFlow [12] introduced a tree-based approach that combined the

advantages of tree ensembles with the flexibility of modeling probability distributions using normalizing flows. By using a tree-based model as a feature extractor and combining it with a conditional variant of normalizing flow, TreeFlow enabled the modeling of complex distributions in regression outputs. While TreeFlow has shown superior performance in some cases, its lack of end-to-end training may result in suboptimal results.

In conclusion, the existing methods for probabilistic regression on tabular data often have limitations in terms of their modeling flexibility or end-to-end training. NodeFlow addresses these limitations by combining the tree-based NODE with the flexibility of CNFs, offering end-to-end training and a unique solution for probabilistic regression on tabular data.

3. NodeFlow

The architecture of NodeFlow is provided in Figure 1. The real-valued input vector $\mathbf{x}$ of dimensionality D is initially processed using a Neural Oblivious Decision Ensemble, consisting of NODE Layers (details of the layer are depicted in Figure 2) arranged in a multi-layer hierarchical structure. It allows the extraction of rich hierarchical representation $\mathbf{w}$. We use that vector as a conditioning factor for the conditional Continuous Normalizing Flow (CNF) in the next step. This component is responsible for the flexible modeling of the conditional probabilistic distribution of vector $\mathbf{y}$. It is worth mentioning that there are no restrictions on the response vector dimensionality. Thus, we could cover both uni- and multivariate regression problems. The whole architecture is trained in an end-to-end fashion using gradient-based optimization.

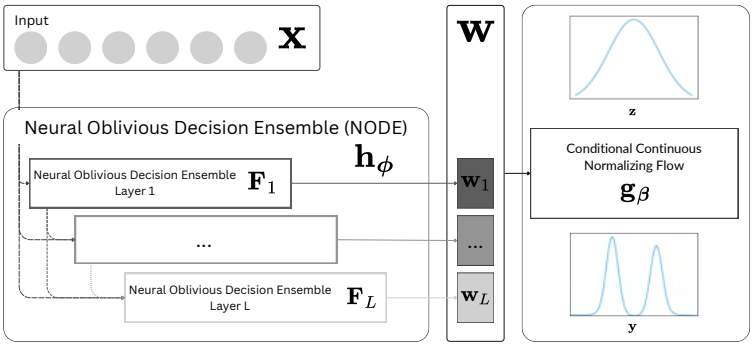


Figure 1. Architectural overview: NodeFlow leverages a Neural Oblivious Decision Ensemble (NODE) to process the input vector, extracting a hierarchical representation. This representation conditions a Continuous Normalizing Flow (CNF), enabling the flexible modeling of the probabilistic distribution of the multidimensional response vector.

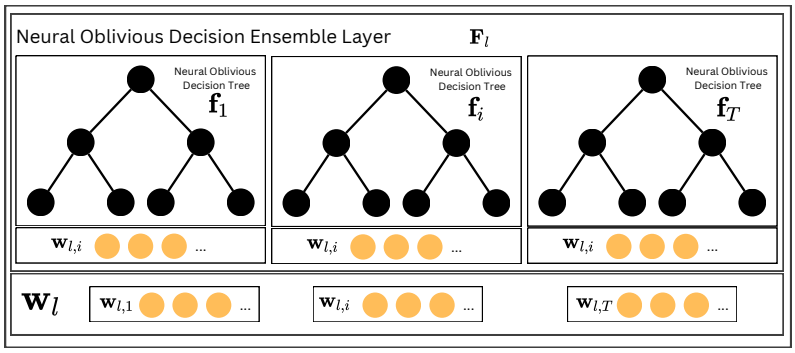


Figure 2. The Neural Oblivious Decision Ensemble (NODE) layer is a key component of NodeFlow's architecture. It comprises several Neural Oblivious Decision Trees, each generating a multidimensional output vector. These vectors are then combined through concatenation to produce the final output of the NODE Layer.

3.1. Extracting Hierarchical Representation with NODE

In order to extract a rich hierarchical representation for a given input $\mathbf{x}$, we utilize Neural Oblivious Decision Ensemble (NODE) $\mathbf{h}_{\boldsymbol{\phi}}(\mathbf{x})$ parametrized by $\boldsymbol{\phi}$, which is a machine learning architecture that combines differentiable oblivious decision trees $\mathbf{f}(\mathbf{x})$ (ODTs). In this section, we start by introducing the ODTs. Then, we discuss the composition of the ODTs into the NODE Layer, and finally, we present the NODE component responsible for the hierarchical representation extraction in NodeFlow.

A single differentiable oblivious decision tree $\mathbf{f}(\mathbf{x})$ of depth d is defined as:

$$\mathbf{f}(\mathbf{x}) = \sum_{j=1}^{2^d} r_j \cdot l_j(\mathbf{x}), \tag{1}$$

where $\mathbf{r} = [r_1, \ldots, r_{2^d}]$ is a 2^d-dimensional vector of real-valued trainable responses for each of the considered leaves in the tree, and $\mathbf{l}(\mathbf{x}) = [l_1(\mathbf{x}), \ldots, l_{2^d}(\mathbf{x})]$ is a 2^d-dimensional vector of real-valued entries from the range $[0, 1]$. The vector is called a "choice vector" and corresponds to the probability of the sample ending up in the specific leaf.

To compute the choice vector, it is requisite to perform a multiplication of the probabilities associated with selecting either the left or right path across successive depth levels within the tree structure. It is important to note that in an oblivious decision tree, only one decision is made at each level of depth, which is referred to as $c_i(\mathbf{x})$ at depth i. The final choice vector $\mathbf{l}$ is derived using the formula:

$$\mathbf{l}(\mathbf{x}) = \begin{bmatrix} c_1(\mathbf{x}) \\ 1 - c_1(\mathbf{x}) \end{bmatrix} \otimes \begin{bmatrix} c_2(\mathbf{x}) \\ 1 - c_2(\mathbf{x}) \end{bmatrix} \otimes \cdots \otimes \begin{bmatrix} c_d(\mathbf{x}) \\ 1 - c_d(\mathbf{x}) \end{bmatrix}, \tag{2}$$

where $\otimes$ denotes the Kronecker product.

To ensure differentiability during training in the tree split, we utilized the α-entmax function [19], which generalizes the Softmax ($\alpha = 1$) and Sparsemax ($\alpha = 2$) functions and allows for the learning of sparse choices through gradient-based learning methods. The feature choice function $c_i(\mathbf{x})$ is then calculated as a two-class entmax function over the transformed output of the feature selection function $k_i(\mathbf{x})$. This can be expressed formally as:

$$c_i(\mathbf{x}) = \text{entmax}_\alpha \left([\frac{k_i(\mathbf{x}) - b_i}{\tau_i}, 0] \right) \tag{3}$$

where b_i and τ_i are learnable threshold and scale parameters, and α is the entmax function's hyperparameter that controls the level of "sparsity" in the output. In addition, the function for selecting differentiable features can be written as follows:

$$k_i(\mathbf{x}) = \sum_{j=1}^{D} x_j \cdot p_j^{(i)}, \tag{4}$$

where $\mathbf{p}^{(i)}$ is the D-dimensional vector of feature selection weights given by the formula $\mathbf{p}^{(i)} = \text{entmax}_\alpha(F_{i,\cdot})$. Moreover, $\mathbf{F} \in \mathbb{R}^{d \times D}$ is called the feature selection matrix, and it is a real-valued, learnable matrix.

In summary, the differentiable oblivious decision tree, denoted as $\mathbf{f}$, is parameterized by the response vector $\mathbf{r}$, threshold values $\boldsymbol{\tau}$, scale factors $\mathbf{b}$, and the feature selection matrix $\mathbf{F}$, facilitating gradient-based learning.

To form the Neural Oblivious Decision Ensemble layer F_l (depicted in Figure 2), we need to concatenate all outputs of the T individual $\mathbf{f}_1, \ldots, \mathbf{f}_T$ ODTs forming the layer. The final output can be written as

$$F_l(\cdot) = [\mathbf{f}_1(\cdot), \ldots, \mathbf{f}_T(\cdot)]. \tag{5}$$

Finally, the NODE architecture $\mathbf{h}_{\boldsymbol{\phi}}(\mathbf{x})$ is composed of L stacked NODE layers in a similar fashion to the DenseNet model. It means that each layer takes the concatenated

outputs of all previous layers as input, allowing the model to learn both low-level and high-level features. It can be written as:

$$\mathbf{w}_0 = \mathbf{x}; \quad \forall_{l\in[1,L]}\ \mathbf{w}_l = [\mathbf{F}_l(\mathbf{w}_{l-1}), \mathbf{w}_{l-1}]. \tag{6}$$

The outputs from each layer are concatenated to create the final representation extracted using NODE, $\mathbf{w} = [\mathbf{w_1}, \ldots, \mathbf{w_L}] = \mathbf{h}_{\boldsymbol{\phi}}(\mathbf{x})$. The representation $\mathbf{w}$ is further delivered to CNFs as a conditioning factor.

3.2. Probabilistic Modeling with CNFs

We consider the conditional variant of CNFs provided in [20,21], where the conditional factor $\mathbf{w} = \mathbf{h}_{\boldsymbol{\phi}}(\mathbf{x})$ is delivered to the function of the dynamics of $\mathbf{z}(t)$, $\mathbf{g}_{\boldsymbol{\beta}}(\mathbf{z}(t), t, \mathbf{w})$, parametrized by $\boldsymbol{\beta}$. In the CNF setting, we aim at finding a solution $\mathbf{y} := \mathbf{z}(t_1)$ for the differential equation, assuming the given initial state $\mathbf{z} := \mathbf{z}(t_0)$ with a known prior, where $\mathbf{z}$ is a random variable, $\mathbf{z}(t_0)$ is a base distribution, and $\mathbf{z}(t_1)$ constitutes our observable data. Moreover, t_0 and t_1 denote the start and end points, respectively, of the continuous transformation process. The transformation function between $\mathbf{y}$ and $\mathbf{z}$ is represented as:

$$\mathbf{y} = \mathbf{u}_{\boldsymbol{\beta},\boldsymbol{\phi}}(\mathbf{z}, \mathbf{x}) = \mathbf{z} + \int_{t_0}^{t_1} \mathbf{g}_{\boldsymbol{\beta}}(\mathbf{z}(t), t, \mathbf{h}_{\boldsymbol{\phi}}(\mathbf{x}))dt. \tag{7}$$

The inverse form of the transformation $\mathbf{u}_{\boldsymbol{\beta},\boldsymbol{\phi}}(\cdot)$ is given by equation:

$$\mathbf{z} = \mathbf{u}^{-1}_{\boldsymbol{\beta},\boldsymbol{\phi}}(\mathbf{y}, \mathbf{x}) = \mathbf{y} - \int_{t_0}^{t_1} \mathbf{g}_{\boldsymbol{\beta}}(\mathbf{z}(t), t, \mathbf{h}_{\boldsymbol{\phi}}(\mathbf{x}))dt. \tag{8}$$

Finally, we can calculate the log-probability of target variable $\mathbf{y}$ given the vector of features $\mathbf{x}$ by the following formula:

$$\log p(\mathbf{y}|\mathbf{x}) = \log p(\mathbf{z}) - \int_{t_0}^{t_1} \mathrm{Tr}\left(\frac{\partial \mathbf{g}_{\boldsymbol{\beta}}(\mathbf{z}(t), t, \mathbf{h}_{\boldsymbol{\phi}}(\mathbf{x}))}{\partial \mathbf{z}(t)}\right)dt, \tag{9}$$

which can be solved analogously to FFJORD [22] by employing the adjoint method to backpropagate through the solution of the neural ODE.

3.3. Training NodeFlow

Using the formula (9) that directly defines log-probability, we can train NodeFlow by directly optimizing the negative log-likelihood function. Let us assume we are given a dataset $\mathcal{D} = (\mathbf{x}_n, y_n)_{n=1..N}$, where $\mathbf{x}_n = (x_n^1, \ldots, x_n^D)$ represents a D-dimensional random feature vector, and $\mathbf{y}_n = (y_n^1, \ldots, y_n^P)$ is the P-dimensional vector of targets. The training of the probabilistic model involves minimizing the conditional negative log-likelihood function (NLL), defined as:

$$Q(\boldsymbol{\beta}, \boldsymbol{\phi}) = -\sum_{n=1}^{N} \log p(\mathbf{y}_n|\mathbf{x}_n, \boldsymbol{\beta}, \boldsymbol{\phi}). \tag{10}$$

The goal during the training process is to find the optimal parameters $\boldsymbol{\beta}^*$ and $\boldsymbol{\phi}^*$ such that:

$$\boldsymbol{\beta}^*, \boldsymbol{\phi}^* = \arg\min_{\boldsymbol{\beta},\boldsymbol{\phi}} Q(\boldsymbol{\beta}, \boldsymbol{\phi}). \tag{11}$$

All model parameters $\boldsymbol{\beta}, \boldsymbol{\phi}$ are trained end to end by optimizing the above-mentioned NLL using the standard gradient-based approach. Such an approach simplifies the modeling process by allowing the entire model to be trained using a single optimization algorithm. Moreover, the model can automatically learn relevant hierarchical representations of the data directly from the raw input data, capturing both low-level and high-level features. This eliminates the need for manual feature engineering, which can be time-consuming and require domain expertise.

4. Experiments

In this section, we present a comprehensive set of experiments to evaluate the performance and effectiveness of NodeFlow in the context of tabular regression problems. We aimed to assess NodeFlow's capabilities in capturing complex data distributions, generating accurate point estimates, and quantifying uncertainty. To achieve this, we conducted evaluations on univariate and multivariate benchmark datasets, comparing NodeFlow with other reference methods. We measured the performance using various evaluation metrics such as the negative log-likelihood (NLL), Continuous Ranked Probability Score (CRPS), and Root-Mean-Square Error (RMSE). Through these experiments, we aimed to demonstrate the performance and flexibility of NodeFlow in probabilistic regression tasks, contributing to the advancement of the field and providing insights for practical applications.

4.1. Methodology

In our evaluation, we adhered to the established probabilistic regression benchmark, as delineated in previous studies [9,11,12], excluding the Boston dataset in consideration of ethical concerns [23]. For univariate regression, we employed nine datasets from the UCI Machine Learning Repository and six datasets for multivariate regression as suggested by [12], with comprehensive dataset details provided in the Appendix A. In alignment with protocols from the referenced literature, we generated 20 random folds for the univariate regression datasets (with the exception of Protein at five folds and Year MSD at a single fold), designating 10% of the data for testing in each fold. The remainder was divided into an 80%/20% training/validation split for epoch selection. Our results are presented as the mean and standard deviation across validation folds. We benchmarked NodeFlow against a suite of models, including four tree-based probabilistic models (NGBoost, RoNGBa, CatBoost, PGBM), a deep learning approach (Deep Ensemble), and a hybrid model (TreeFlow) for univariate tasks. For multivariate regression challenges, we adopted training/testing splits as per the referenced protocols, comparing NodeFlow against NGBoost variants and TreeFlow. The architecture specifics and hyperparameter tuning methodology for NodeFlow are detailed in the Appendix B.

4.2. Probabilistic Regression Framework

This segment evaluates NodeFlow's performance within a probabilistic framework, analyzing its negative log-likelihood (NLL) scores against benchmark datasets for both univariate and multivariate regression tasks previously outlined.

In Table 1, we present the evaluation results for the univariate regression task, where NodeFlow exhibited competitive performance across a range of datasets, frequently achieving the best or second-best NLL scores. Notably, NodeFlow excelled on the Year MSD dataset and secures commendable second-best results on the Wine, Protein, Power, and Kin8nm datasets. Our analysis extended to a detailed comparison of NodeFlow against various methodological approaches, including deep learning-based methods, tree-based ensemble methods, and the hybrid method TreeFlow. Against the Deep Ensemble, NodeFlow consistently demonstrated superior or at least equivalent performance, with particularly noteworthy achievements on the Energy, Power, Protein, Wine, and Yacht datasets. This is especially significant for the Protein and Wine datasets, which are characterized by their underlying multimodal target distributions—a scenario where NodeFlow's capabilities of flexible distribution modeling were especially advantageous (refer to [12] for details). When compared to tree-based methods such as CatBoost, NGBoost, RoNGBa, and PGBM, NodeFlow maintained a competitive edge, often outperforming or matching the best results, underscoring its robust ability to model complex data relationships within tabular datasets. In direct comparison with TreeFlow, NodeFlow and TreeFlow exhibited closely matched performance, with each method surpassing the other under different circumstances. This comparative analysis not only highlights NodeFlow's versatile efficacy across a broad spectrum of univariate regression challenges but also its capacity to address the intricacies of tabular data modeling through its advanced, adaptive learning framework.

Table 1. Benchmark for *univariate probabilistic* regression problem with tabular data using negative log-likelihood (NLL) as the metric. The best results are marked by **bold text**, and the second best results are underlined.

Dataset	Deep. Ens.	CatBoost	NGBoost	RoNGBa	PGBM	TreeFlow	NodeFlow
Concrete	3.06 ± 0.18	3.06 ± 0.13	3.04 ± 0.17	2.94 ± 0.18	$\mathbf{2.75 \pm 0.21}$	3.02 ± 0.15	3.15 ± 0.21
Energy	1.38 ± 0.22	1.24 ± 1.28	0.60 ± 0.45	$\mathbf{0.37 \pm 0.28}$	1.74 ± 0.04	0.85 ± 0.35	0.90 ± 0.25
Kin8nm	$\mathbf{-1.20 \pm 0.02}$	-0.63 ± 0.02	-0.49 ± 0.02	-0.60 ± 0.03	-0.54 ± 0.04	-1.03 ± 0.06	-1.10 ± 0.05
Naval	$\mathbf{-5.63 \pm 0.05}$	-5.39 ± 0.04	-5.34 ± 0.04	-5.49 ± 0.04	-3.44 ± 0.04	-5.54 ± 0.16	-5.45 ± 0.08
Power	2.79 ± 0.04	2.72 ± 0.12	2.79 ± 0.11	2.65 ± 0.08	$\mathbf{2.60 \pm 0.02}$	2.65 ± 0.06	2.62 ± 0.05
Protein	2.83 ± 0.02	2.73 ± 0.07	2.81 ± 0.03	2.76 ± 0.03	2.79 ± 0.01	$\mathbf{2.02 \pm 0.02}$	2.04 ± 0.04
Wine	0.94 ± 0.12	0.93 ± 0.08	0.91 ± 0.06	0.91 ± 0.08	0.97 ± 0.20	$\mathbf{-0.56 \pm 0.62}$	-0.21 ± 0.28
Yacht	1.18 ± 0.21	0.41 ± 0.39	0.20 ± 0.26	1.03 ± 0.44	$\mathbf{0.05 \pm 0.28}$	0.72 ± 0.40	0.79 ± 0.55
Year MSD	$3.35 \pm$ NA	$3.43 \pm$ NA	$3.43 \pm$ NA	$3.46 \pm$ NA	$3.61 \pm$ NA	$3.27 \pm$ NA	$\mathbf{3.09 \pm}$ NA

In Table 2, we detail NodeFlow's performance across multivariate probabilistic regression tasks, where it consistently outperformed competing approaches in five of the six datasets examined. Compared with TreeFlow, NodeFlow's superiority was particularly evident in datasets with multiple target dimensions, such as scm20d (16 target dimensions) and Energy (17 target dimensions). For two-dimensional target datasets like Parkinsons and US Flight, NodeFlow continued to outperform, albeit with a narrower margin. The distinction became more nuanced with one-dimensional targets, as presented in prior analyses, where NodeFlow and TreeFlow showed competitive yet comparable results. This differentiation underscores the strength of NodeFlow's end-to-end learning model, which excels in complex, high-dimensional settings by providing finely tuned representations. Such comprehensive learning is absent in TreeFlow, limiting its effectiveness in comparison. This evidence reinforces the indispensable value of end-to-end learning in achieving optimal performance, particularly in addressing the intricate demands of multivariate regression problems.

Table 2. Benchmark for *multivariate probabilistic* regression problem with tabular data using negative log-likelihood (NLL) as the metric. The best results are marked by **bold text**, and the second best results are underlined.

Dataset	Ind. NGBoost	NGBoost	TreeFlow	NodeFlow
Parkinsons	6.86	5.85	5.26	**5.06**
Scm20d	94.40	94.81	93.41	**91.98**
Wind	−0.65	−0.67	−2.57	**−3.20**
Energy	166.90	175.80	180.00	**163.86**
USflight	9.56	8.57	7.49	**7.38**
Ocean.	7.74	**7.73**	7.84	7.81

4.3. Point-Prediction Regression Setup

This section assesses the effectiveness of our method in a point-prediction context by comparing its Root-Mean-Square Error (RMSE) scores on the univariate regression datasets. To calculate the RMSE results for the TreeFlow and NodeFlow methods, we used the RMSE@K metric introduced in [12], where $K = 2$. This metric is suitable for uni- and multivariate regression problems with multiple-point predictions. We present the results in Table 3. Our method achieved the best results on two datasets and ranked second on two others. For the remaining datasets, it remained competitive with benchmark methods. Notably, these results are commendable, considering our approach is designed for probabilistic setups. Providing point estimates, particularly from multimodal distributions, presents unique challenges compared to simply taking the mean of parametric distributions like Gaussian. This context underscores the strength of our method's performance across various datasets.

Table 3. Benchmark for *univariate point prediction* regression problem with tabular data using Root-Mean-Square Error (RMSE). Note that for TreeFlow and NodeFlow, we used the RMSE@2 metric, which is more relevant. The best results are marked by **bold text**, and the second best results are underlined.

Dataset	Deep. Ens.	CatBoost	NGBoost	RoNGBa	PGBM	TreeFlow (@2)	NodeFlow(@2)
Concrete	6.03 ± 0.58	5.21 ± 0.53	5.06 ± 0.61	4.71 ± 0.61	$\mathbf{3.97 \pm 0.76}$	5.41 ± 0.71	5.51 ± 0.66
Energy	2.09 ± 0.29	0.57 ± 0.06	0.46 ± 0.06	$\mathbf{0.35 \pm 0.07}$	$\mathbf{0.35 \pm 0.06}$	0.65 ± 0.12	0.70 ± 0.40
Kin8nm	0.09 ± 0.00	0.14 ± 0.00	0.16 ± 0.00	0.14 ± 0.00	0.13 ± 0.01	0.10 ± 0.01	$\mathbf{0.08 \pm 0.00}$
Naval	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$
Power	4.11 ± 0.17	3.55 ± 0.27	3.70 ± 0.22	3.47 ± 0.19	$\mathbf{3.35 \pm 0.15}$	3.79 ± 0.25	3.94 ± 0.16
Protein	4.71 ± 0.06	3.92 ± 0.08	4.33 ± 0.03	4.21 ± 0.06	3.98 ± 0.06	$\mathbf{3.01 \pm 0.06}$	4.32 ± 0.03
Wine	0.64 ± 0.04	0.63 ± 0.04	0.62 ± 0.04	0.62 ± 0.05	0.60 ± 0.05	$\mathbf{0.41 \pm 0.09}$	0.44 ± 0.03
Yacht	1.58 ± 0.48	0.82 ± 0.40	$\mathbf{0.50 \pm 0.20}$	0.90 ± 0.35	0.63 ± 0.21	0.75 ± 0.26	1.18 ± 0.47
Year MSD	$8.89 \pm$ NA	$8.99 \pm$ NA	$8.94 \pm$ NA	$9.14 \pm$ NA	$9.09 \pm$ NA	$\mathbf{8.64 \pm}$ NA	$8.84 \pm$ NA

4.4. Summary

In summary, our evaluation of NodeFlow across both probabilistic and point-prediction scenarios demonstrates its efficacy. While NodeFlow's performance on tasks with one-dimensional targets aligns with existing benchmarks, it distinctly excels in handling problems with two or more target dimensions. The results unequivocally indicate that the greater the dimensionality of the target variable, the more pronounced NodeFlow's superiority becomes. This superior performance is attributed to NodeFlow's flexible probabilistic modeling and comprehensive end-to-end learning approach, ensuring highly tailored representations for complex problems. Consequently, NodeFlow stands out as a superior method for probabilistic regression tasks involving high-dimensional targets, affirming its suitability for addressing advanced modeling challenges.

5. Ablation Studies

In the pursuit of a comprehensive understanding of NodeFlow method, a series of ablation studies were undertaken to scrutinize the impacts of critical design choices therein. Specifically, this investigation focused on two integral constituents: the feature representation component, in NodeFlow attained by the usage of NODEs, and the probabilistic modeling segment, which was realized through the utilization of CNFs. We evaluated our methods using both probabilistic and point-prediction frameworks. Additionally, we conducted a qualitative analysis of the learned representations and estimated probability density functions. Moreover, the results of the computational time comparison are included in Section 6.

5.1. Feature Representation Component

In our ablation study, we assessed the critical role of the Neural Oblivious Decision Ensemble (NODE) component in enhancing feature extraction within our proposed framework, NodeFlow. To this end, we conducted both quantitative and qualitative analyses, employing two benchmarking variants for comparison: one with the NODE component removed, relying solely on min-max scaling (termed as *CNF*), and another replacing the NODE with a shallow Multilayer Perceptron (MLP), labeled as *CNF + MLP*.

Quantitative results, detailed in Table 4, evaluate the performance across probabilistic and point-prediction metrics: negative log-likelihood (NLL), Continuous Ranked Probability Score (CRPS), and Root-Mean-Square Error at 2 (RMSE@2), presented as mean values alongside their standard deviations. The experimental setup was kept consistent with the main experiments.

Our findings reveal that NodeFlow, with the NODE component integrated, consistently delivered the lowest NLL values across a majority of datasets, highlighting its exceptional data modeling and prediction accuracy capabilities. Additionally, NodeFlow surpassed comparative approaches in CRPS, indicating its enhanced precision in proba-

bilistic forecasting. Furthermore, NodeFlow achieved the most favorable RMSE scores, underlining the NODE component's pivotal role in achieving precise point predictions.

Table 4. Ablation study of the *feature representation* component in terms of negative log-likelihood (NLL), Continuous Ranked Probability Score (CRPS), and Root-Mean-Square Error at 2 (RMSE@2) metrics.

Dataset	NLL			CRPS			RMSE		
	CNF	CNF + MLP	NodeFlow	CNF	CNF + MLP	NodeFlow	CNF	CNF + MLP	NodeFlow
Concrete	3.24 ± 0.28	$\mathbf{3.15 \pm 0.13}$	$\mathbf{3.15 \pm 0.21}$	3.80 ± 1.33	3.39 ± 0.34	$\mathbf{2.80 \pm 0.34}$	7.16 ± 2.22	6.43 ± 0.54	$\mathbf{5.51 \pm 0.66}$
Energy	2.90 ± 0.45	2.43 ± 0.31	$\mathbf{0.90 \pm 0.25}$	2.73 ± 1.45	1.73 ± 0.77	$\mathbf{0.35 \pm 0.14}$	4.90 ± 2.41	3.26 ± 1.26	$\mathbf{0.70 \pm 0.40}$
Kin8nm	-0.66 ± 0.12	-0.86 ± 0.07	$\mathbf{-1.10 \pm 0.05}$	0.07 ± 0.01	0.06 ± 0.00	$\mathbf{0.04 \pm 0.00}$	0.14 ± 0.02	0.11 ± 0.01	$\mathbf{0.08 \pm 0.00}$
Naval	-3.42 ± 0.34	-3.55 ± 0.21	$\mathbf{-5.45 \pm 0.08}$	0.01 ± 0.00	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	0.01 ± 0.00	0.01 ± 0.00	$\mathbf{0.00 \pm 0.00}$
Power	2.92 ± 0.24	2.90 ± 0.26	$\mathbf{2.62 \pm 0.05}$	2.59 ± 1.00	2.61 ± 1.15	$\mathbf{1.95 \pm 0.06}$	4.69 ± 1.71	4.77 ± 1.94	$\mathbf{3.94 \pm 0.16}$
Protein	2.57 ± 0.03	2.56 ± 0.02	$\mathbf{2.04 \pm 0.04}$	2.69 ± 0.04	2.67 ± 0.03	$\mathbf{1.75 \pm 0.03}$	5.88 ± 0.11	5.81 ± 0.10	$\mathbf{4.32 \pm 0.03}$
Wine	0.07 ± 0.62	0.34 ± 0.63	$\mathbf{-0.21 \pm 0.28}$	0.36 ± 0.04	0.37 ± 0.04	$\mathbf{0.34 \pm 0.02}$	0.54 ± 0.14	0.61 ± 0.14	$\mathbf{0.44 \pm 0.09}$
Yacht	1.92 ± 1.67	1.35 ± 1.82	$\mathbf{0.79 \pm 0.55}$	2.45 ± 3.06	1.26 ± 2.35	$\mathbf{0.50 \pm 0.19}$	5.06 ± 5.42	2.71 ± 4.33	$\mathbf{1.18 \pm 0.47}$

In our qualitative analysis, we visualized feature representations derived from the models, utilizing dimensionality reduction via the UMAP algorithm [24] and color-coding each point according to its target variable. Figure 3 illustrates these representations for the Energy dataset. The leftmost visualization corresponds to the CNF model, which, lacking additional processing layers, essentially reflects the rescaled raw dataset within the $(-1, 1)$ range. The middle image depicts the representation from the CNF + MLP model, while the rightmost image shows the outcome of employing a NODE within the NodeFlow method. Comparatively, the NodeFlow method's representation, facilitated by NODE processing, showcases a significantly enhanced separation and disentanglement of observations, with distinct clusters forming around similar target values. This level of disentanglement, absent in the CNF models' representations, likely plays a crucial role in NodeFlow's superior performance across quantitative metrics.

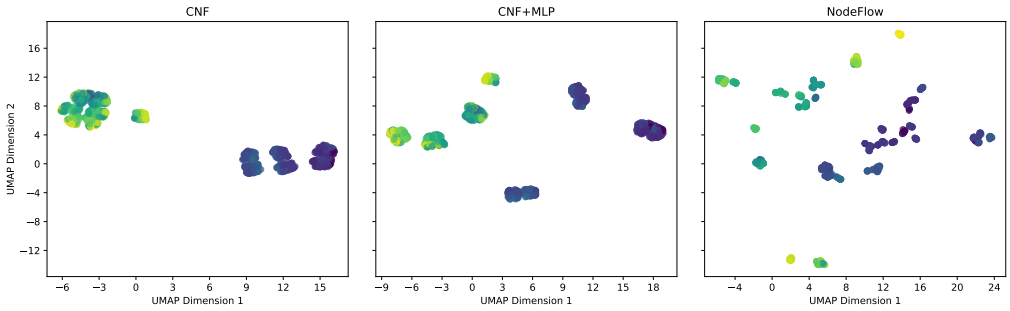


Figure 3. Feature representations for the Energy dataset via UMAP for the ablation study. Left: CNF model, showing rescaled data within $(-1, 1)$. Center: CNF + MLP model, indicating improved structuring. Right: NodeFlow with NODE, illustrating the superior hierarchical organization. Points are color-coded by the target variable.

Collectively, these outcomes validate the NODE component's indispensable contribution to NodeFlow's architecture, ensuring competitive or superior performance in NLL, CRPS, and RMSE metrics and disentangled and more clearly separated representations compared to the alternatives examined.

5.2. Probabilistic Modeling Component

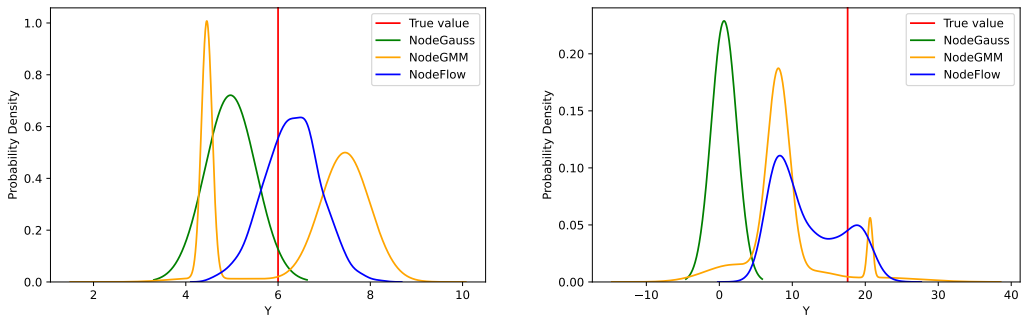
In this ablation study, we evaluated the effectiveness and fit of the probabilistic modeling component within our framework. Specifically, we substituted the CNF component with standard probabilistic distributions, labeling these variants as *NodeGauss* (using a Gaussian distribution) and *NodeGMM* (employing a mixture of Gaussians). This experimental design mirrors the setup of our previous ablation studies.

The findings, detailed in Table 5, indicate that NodeFlow consistently surpassed both NodeGauss and NodeGMM in the negative log-likelihood (NLL) across the majority of the datasets, with NodeGMM outperforming NodeFlow only in a single dataset instance. In terms of the Continuous Ranked Probability Score (CRPS), NodeFlow attained the lowest scores universally, indicating a more accurate calibration of predictive uncertainty relative to the alternatives. Point-prediction results further underscored NodeFlow's superiority as the most effective approach. Notably, these outcomes underscored the benefit of integrating a versatile probabilistic modeling component, as evidenced by the enhanced performance across all evaluated metrics.

Table 5. Ablation study of the *probabilistic modeling* component in terms of negative log-likelihood (NLL), Continuous Ranked Probability Score (CRPS), and Root-Mean-Square Error at 2 (RMSE@2) metrics.

Dataset	NLL			CRPS			RMSE		
	NodeGauss	NodeGMM	NodeFlow	NodeGauss	NodeGMM	NodeFlow	NodeGauss	NodeGMM	NodeFlow
Concrete	3.13 ± 0.39	$\mathbf{3.03 \pm 0.18}$	3.15 ± 0.21	8.54 ± 0.49	9.04 ± 0.49	$\mathbf{2.80 \pm 0.34}$	15.52 ± 0.86	16.08 ± 0.86	$\mathbf{5.51 \pm 0.66}$
Energy	1.84 ± 0.23	1.70 ± 0.21	$\mathbf{0.90 \pm 0.25}$	5.16 ± 0.27	5.59 ± 0.27	$\mathbf{0.35 \pm 0.14}$	9.53 ± 0.41	9.94 ± 0.41	$\mathbf{0.70 \pm 0.40}$
Kin8nm	-0.90 ± 0.07	-0.97 ± 0.06	$\mathbf{-1.10 \pm 0.05}$	0.14 ± 0.00	0.15 ± 0.00	$\mathbf{0.04 \pm 0.00}$	0.18 ± 0.01	0.22 ± 0.01	$\mathbf{0.08 \pm 0.00}$
Naval	-4.91 ± 0.29	-4.95 ± 0.15	$\mathbf{-5.45 \pm 0.08}$	0.01 ± 0.00	0.01 ± 0.00	$\mathbf{0.00 \pm 0.00}$	0.01 ± 0.00	0.01 ± 0.00	$\mathbf{0.00 \pm 0.00}$
Power	2.84 ± 0.05	2.76 ± 0.04	$\mathbf{2.62 \pm 0.05}$	8.88 ± 0.12	9.59 ± 0.12	$\mathbf{1.95 \pm 0.06}$	16.10 ± 0.22	16.88 ± 0.23	$\mathbf{3.94 \pm 0.16}$
Protein	2.84 ± 0.07	2.36 ± 0.12	$\mathbf{2.04 \pm 0.04}$	3.39 ± 0.02	3.39 ± 0.03	$\mathbf{1.75 \pm 0.03}$	6.03 ± 0.06	7.40 ± 0.36	$\mathbf{4.32 \pm 0.03}$
Wine	0.97 ± 0.08	0.51 ± 0.37	$\mathbf{-0.21 \pm 0.28}$	0.45 ± 0.03	0.45 ± 0.03	$\mathbf{0.34 \pm 0.02}$	0.82 ± 0.05	0.59 ± 0.16	$\mathbf{0.44 \pm 0.09}$
Yacht	2.26 ± 0.72	1.84 ± 0.63	$\mathbf{0.79 \pm 0.55}$	6.67 ± 1.52	6.62 ± 1.58	$\mathbf{0.50 \pm 0.19}$	14.19 ± 3.02	14.26 ± 2.95	$\mathbf{1.18 \pm 0.47}$

Figure 4 illustrates the probability density functions estimated by NodeFlow, NodeGauss, and NodeGMM for selected samples from the Wine Quality and Protein datasets. These datasets were chosen due to their complex distributions and the significant differences in results among the models. In the Wine Quality example, NodeFlow produced a distribution concentrated between values six and seven, lacking the distinct peak characteristic of Gaussian distributions. The Protein dataset example showcased NodeFlow's ability to model a bimodal distribution with significant probability mass between peaks and a heavy right tail. Notably, both NodeGauss and NodeGMM struggled to fully capture the complexity of these sample distributions. This observation underscored the necessity for more sophisticated distributional modeling, as provided by our Conditional Normalizing Flow (CNF) component in NodeFlow.



(**a**) Wine Quality (**b**) Protein

Figure 4. Comparison of probability density functions estimated by NodeFlow, NodeGauss, and NodeGMM for selected samples from the Wine Quality and Protein datasets.

Overall, NodeFlow's uniform advantage across diverse metrics and datasets together with supporting visualizations robustly validates the integral role of the CNF component in its architecture, underscoring its indispensability for achieving optimal model performance.

6. Computational Time Comparison

In this analysis, we evaluated the training duration of NodeFlow relative to benchmark models from ablation studies, including CNF, CNF + MLP from the feature representation study, and NodeGauss and NodeGMM from the probabilistic modeling investigation. Our objective was to elucidate the computational demands of training each model across

various datasets, as detailed in Table 6. The table delineates the mean training times and their standard deviations, offering insights into both average performance and variability.

Table 6. Comparative analysis of training duration for NodeFlow and ablation study approaches.

Dataset	CNF	CNF + MLP	NodeGauss	NodeGMM	NodeFlow
Concrete	335.23 ± 64.91 s	431.65 ± 232.73 s	43.82 ± 15.28 s	25.20 ± 9.74 s	482.69 ± 127.31 s
Energy	70.63 ± 6.34 s	80.83 ± 7.33 s	23.25 ± 7.35 s	15.48 ± 6.36 s	687.24 ± 99.62 s
Kin8nm	137.19 ± 9.76 s	169.22 ± 40.49 s	45.72 ± 13.31 s	55.14 ± 16.32 s	308.89 ± 61.57 s
Naval	213.13 ± 61.62 s	228.93 ± 20.99 s	56.22 ± 20.75 s	47.74 ± 27.42 s	2413.23 ± 649.67 s
Power	141.333 ± 12.30 s	180.81 ± 17.90 s	40.19 ± 15.56 s	43.93 ± 15.51 s	1360.29 ± 192.94 s
Protein	373.255 ± 40.39 s	417.45 ± 52.54 s	217.13 ± 22.18 s	224.45 ± 63.75 s	3018.98 ± 616.95 s
Wine	352.964 ± 69.65 s	353.93 ± 67.75 s	26.82 ± 10.80 s	11.92 ± 6.41 s	614.85 ± 136.68 s
Yacht	203.561 ± 117.80 s	259.64 ± 135.60 s	19.50 ± 10.33 s	13.31 ± 4.60 s	567.44 ± 216.81 s

In the feature representation study, the marginal difference in training times among NodeFlow, CNF, and CNF + MLP suggests that the NODE component's integration is cost-effective, enhancing the model output without a corresponding surge in training duration. Conversely, the probabilistic modeling study indicates a more pronounced disparity in training times, particularly between NodeFlow and the NodeGauss and NodeGMM variants, with NodeFlow achieving superior results with a proportional increase in computational time.

Overall, NodeFlow presents itself as a robust solution for probabilistic regression tasks on tabular data, adeptly balancing efficiency in training time with excellence in performance. This equilibrium makes NodeFlow a compelling option for both academic research and practical implementation, highlighting its potential as a preferred method in the domain.

7. Conclusions

In this study, we introduced NodeFlow, a novel framework for probabilistic regression on tabular data, leveraging Neural Oblivious Decision Ensembles (NODEs) and Conditional Continuous Normalizing Flows (CNFs). Our evaluations confirmed NodeFlow's exceptional capability in managing high-dimensional multivariate probabilistic regression tasks, effectively aligning with benchmarks for tasks with one-dimensional targets. Ablation studies elucidated the critical roles of the NODE and CNF components in NodeFlow's architecture, enhancing feature processing and complex distribution modeling, respectively. Moreover, NodeFlow emerges as a robust solution for advanced modeling and uncertainty quantification in regression tasks, adeptly balancing performance with computational efficiency. It not only establishes a significant presence in the domain of probabilistic regression but also lays a foundation for future advancements in machine learning interpretability and robustness. The differentiability of NodeFlow's architecture is particularly conducive to further research in interpretability techniques, including counterfactual explanations, feature attribution, and adversarial example generation, promising substantial contributions to the field's evolution.

Author Contributions: Conceptualization, P.W. and M.Z.; methodology, P.W.; software, P.W. and O.F.; validation, P.W. and O.F.; formal analysis, P.W. and O.F.; investigation, P.W. and O.F.; resources, P.W.; data curation, P.W.; writing—original draft preparation, P.W.; writing—review and editing, M.Z.; visualization, P.W.; supervision, M.Z.; project administration, P.W.; funding acquisition, M.Z. All authors have read and agreed to the published version of the manuscript.

Funding: The work conducted by Patryk Wielopolski, Oleksii Furman, and Maciej Zieba was supported by the National Centre of Science (Poland) grant no. 2021/43/B/ST6/02853. Moreover, we gratefully acknowledge Polish high-performance computing infrastructure PLGrid (HPC Center: ACK Cyfronet AGH) for providing computer facilities and support within computational grant no. PLG/2023/016636.

Institutional Review Board Statement: Not applicable.

Data Availability Statement: The data presented in this study are openly available in https://github.com/pfilo8/NodeFlow (accessed date: 28 May 2024).

Conflicts of Interest: Author Maciej Zięba was employed by the company Tooploox Ltd. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Appendix A. Datasets

In this section, we delve into the details of the datasets used in our study to validate the capabilities of NodeFlow empirically. These datasets are the standard in assessing method effectiveness and were chosen to evaluate NodeFlow's performance across various domains and to demonstrate its versatility in addressing complex probabilistic regression tasks. Table A1 furnishes comprehensive details on the datasets employed, encompassing the number of data points (N), the quantity of cross-validation (CV) splits or test dataset observations, along with the feature dimensionality (D) and target dimensionality (P).

What is important and different from the reference methods is that the datasets utilized in our study were scaled to the range $(-1, 1)$, encompassing both the features and target variables. This crucial preprocessing step was undertaken with a specific purpose in mind—to enhance the stability of the learning process within the neural network framework. By scaling both the features and targets to this common range, we aimed to mitigate potential issues related to the magnitude of data values, which can impact the convergence and performance of neural networks during training.

It is worth noting that the tree-based methods with which we compared NodeFlow did not require the extensive scaling of both features and targets as they inherently possess a scale-invariance property. This characteristic stems from the way decision trees partition the feature space, making them less sensitive to variations in feature and target scales and thereby obviating the need for such preprocessing.

Table A1. An overview of the datasets employed in our study to assess the performance of NodeFlow. The table includes information on the number of data points (N), the number of cross-validation (CV) splits or observations in the test dataset, feature dimensionality (D), and target dimensionality (P).

Dataset	N	CV Splits/N_{test}	D	P
Concrete	1030	20 CV	8	1
Energy	768	20 CV	8	1
Kin8nm	8192	20 CV	8	1
Naval	11,934	20 CV	16	1
Power	9568	20 CV	4	1
Protein	45,730	5 CV	9	1
Wine	1588	20 CV	11	1
Yacht	308	20 CV	6	1
Year MSD	515,345	1 CV	90	1
Parkinsons	4112	1763	16	2
scm20d	7173	1793	61	16
WindTurbine	4000	1000	8	6
Energy	57,598	14,400	32	17
usFlight	500,000	200,000	8	2
Oceanographic	373,227	41,470	9	2

Appendix B. Implementation Details

The research methodology adhered to the standard practices characteristic of machine learning projects. All models under consideration were implemented using Python 3.8, leveraging the deep learning library PyTorch. The training employed the usage of the PyTorch Lightning framework. We used the following infrastructure for the experiments:

Intel(R) Xeon(R) Silver 4108 32-Core CPU, 4 NVIDIA GeForce GTX 1080 Ti GPUs, and 126 GB RAM.

In our research paper, we employed a Hyperband Pruner [25] as the hyperparameter search method to optimize our machine learning models. Hyperband Pruner is a highly efficient technique that focuses on identifying promising hyperparameter configurations while discarding less promising ones. To explore the hyperparameter space effectively, we uniformly sampled parameters within the specified ranges, as detailed in Table A2. Each dataset underwent a comprehensive search process, with each fold requiring a maximum duration of three hours. This approach allowed us to tune our models efficiently and select the best-performing hyperparameters, ultimately enhancing the predictive capabilities of our machine learning algorithms.

Based on the results of the hyperparameter search, we conducted a comprehensive analysis to evaluate the significance of hyperparameters in the tuning process. To assess this, we employed the fANOVA Hyperparameter Importance Evaluation algorithm [26], which involves fitting a random forest regression model to predict the objective values of successfully completed trials based on their parameter configurations. The outcomes of this analysis are illustrated in Figure A1.

As depicted in the figure, three particular hyperparameters were identified as crucial in our hyperparameter tuning process. These critical hyperparameters are the number of layers and the depth of the trees within the NODE (Neural Oblivious Decision Ensemble) component and the dimensionality of the hidden layers within the CNF (Conditional Continuous Normalizing Flow) component. These specific hyperparameters played a pivotal role in influencing the model's performance and its ability to generalize effectively. Interestingly, the hyperparameter related to the output dimension of the NODE's tree did

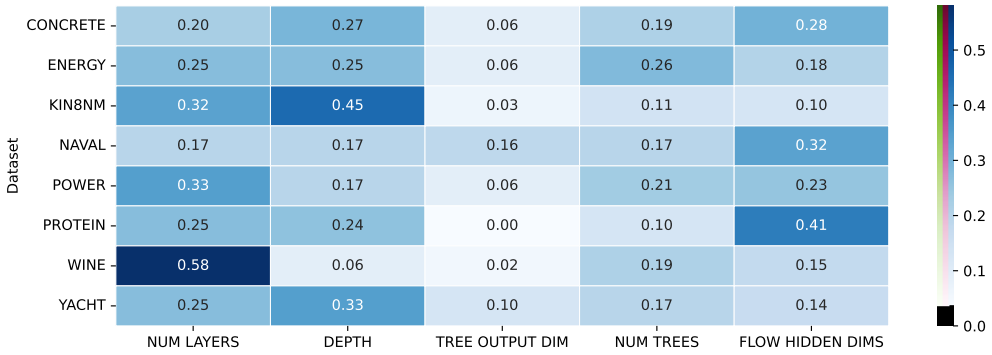


Figure A1. Hyperparameter importance analysis in the NodeFlow tuning process. Importance scores for each dataset and searched hyperparameter were calculated using the fANOVA Hyperparameter Importance Evaluation algorithm, with the highest scores underlining their pivotal role in the optimization process.

Table A2. Comprehensive overview of the hyperparameters employed in our research for optimizing the NodeFlow method. The hyperparameter ranges and settings for various datasets are detailed, allowing for a clear understanding of the tuning process.

Dataset	Num Layers	Depth	Tree Output Dim	Num Trees	Flow Hidden Dims	N Epochs	# of Iterations
CONCRETE	1–8	1–7	1–3	100–600	[4,4], [8,8], [16,16], [32,32]	400	400
ENERGY	1–8	1–6	1–3	100–600	[4,4], [8,8], [16,16], [32,32]	400	300
KIN8NM	1–8	1–6	1–3	100–600	[4,4], [8,8], [16,16], [32,32]	100	100
NAVAL	1–8	1–6	1–3	100–600	[4,4], [8,8], [16,16], [32,32]	300	100
POWER	1–8	1–6	1–3	100–600	[4,4], [8,8], [16,16], [32,32]	200	100
PROTEIN	1–8	1–6	1–3	100–600	[4,4], [8,8], [16,16], [32,32]	100	100
WINE	1–8	1–6	1–3	100–600	[4,4], [8,8], [16,16], [32,32]	400	500
YACHT	1–8	1–6	1–3	100–500	[4,4], [8,8], [16,16], [32,32]	400	400
YEAR MSD	6	2, 4	1	100, 300	[4,4], [8,8], [16,16], [32,32]	10	16

References

1. Borisov, V.; Leemann, T.; Seßler, K.; Haug, J.; Pawelczyk, M.; Kasneci, G. Deep Neural Networks and Tabular Data: A Survey. *arXiv* **2021**, arXiv:2110.01889.
2. Grinsztajn, L.; Oyallon, E.; Varoquaux, G. Why do tree-based models still outperform deep learning on typical tabular data? In Proceedings of the Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, 28 November–9 December 2022.
3. Chen, T.; Guestrin, C. XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 785–794. [CrossRef]
4. Prokhorenkova, L.O.; Gusev, G.; Vorobev, A.; Dorogush, A.V.; Gulin, A. CatBoost: Unbiased boosting with categorical features. In Proceedings of the Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, Montréal, QC, Canada, 3–8 December 2018; pp. 6639–6649.
5. Ke, G.; Meng, Q.; Finley, T.; Wang, T.; Chen, W.; Ma, W.; Ye, Q.; Liu, T. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In Proceedings of the Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, Long Beach, CA, USA, 4–9 December 2017; pp. 3146–3154.
6. Popov, S.; Morozov, S.; Babenko, A. Neural Oblivious Decision Ensembles for Deep Learning on Tabular Data. In Proceedings of the 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, 26–30 April 2020.
7. Abutbul, A.; Elidan, G.; Katzir, L.; El-Yaniv, R. DNF-Net: A Neural Architecture for Tabular Data. *arXiv* **2020**, arXiv:2006.06465 .
8. Gorishniy, Y.; Rubachev, I.; Khrulkov, V.; Babenko, A. Revisiting Deep Learning Models for Tabular Data. In Proceedings of the Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, Virtual, 6–14 December 2021; Ranzato, M., Beygelzimer, A., Dauphin, Y.N., Liang, P., Vaughan, J.W., Eds.; pp. 18932–18943.
9. Duan, T.; Anand, A.; Ding, D.Y.; Thai, K.K.; Basu, S.; Ng, A.Y.; Schuler, A. NGBoost: Natural Gradient Boosting for Probabilistic Prediction. In Proceedings of the 37th International Conference on Machine Learning, ICML 2020, PMLR, Virtual Event, 13–18 July 2020; Volume 119, pp. 2690–2700.
10. Sprangers, O.; Schelter, S.; de Rijke, M. Probabilistic Gradient Boosting Machines for Large-Scale Probabilistic Regression. In Proceedings of the KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, 14–18 August 2021; pp. 1510–1520. [CrossRef]
11. Malinin, A.; Prokhorenkova, L.; Ustimenko, A. Uncertainty in Gradient Boosting via Ensembles. In Proceedings of the 9th International Conference on Learning Representations, ICLR 2021, Virtual Event, 3–7 May 2021.
12. Wielopolski, P.; Zięba, M. TreeFlow: Going Beyond Tree-Based Parametric Probabilistic Regression. In *ECAI 2023*; Frontiers in Artificial Intelligence and Applications; IOS Press: Amsterdam, The Netherlands , 2023; Volume 372, pp. 2631–2638. [CrossRef]
13. Ren, L.; Sun, G.; Wu, J. RoNGBa: A Robustly Optimized Natural Gradient Boosting Training Approach with Leaf Number Clipping. *arXiv* **2019**, arXiv:1912.02338.
14. Arik, S.Ö.; Pfister, T. TabNet: Attentive Interpretable Tabular Learning. In Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, 2–9 February 2021; AAAI Press: Washington, DC, USA, 2021; pp. 6679–6687.
15. Somepalli, G.; Goldblum, M.; Schwarzschild, A.; Bruss, C.B.; Goldstein, T. SAINT: Improved Neural Networks for Tabular Data via Row Attention and Contrastive Pre-Training. *arXiv* **2021**, arXiv:2106.01342.
16. Lakshminarayanan, B.; Pritzel, A.; Blundell, C. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. In Proceedings of the Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, Long Beach, CA, USA, 4–9 December 2017; pp. 6402–6413.
17. Gal, Y.; Ghahramani, Z. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *JMLR Workshop and Conference Proceedings, Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York, NY, USA, 19–24 June 2016*; Balcan, M., Weinberger, K.Q., Eds.; Microtome Publishing: Brookline, MA, USA, 2016; Volume 48, pp. 1050–1059.
18. Hernández-Lobato, J.M.; Adams, R.P. Probabilistic Backpropagation for Scalable Learning of Bayesian Neural Networks. In *JMLR Workshop and Conference Proceedings, Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6–11 July 2015*; Bach, F.R., Blei, D.M., Eds.; Microtome Publishing: Brookline, MA, USA, 2016; Volume 37, pp. 1861–1869.
19. Peters, B.; Niculae, V.; Martins, A.F.T. Sparse Sequence-to-Sequence Models. In Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, 28 July–2 August 2019; Volume 1: Long Papers; Korhonen, A., Traum, D.R., Màrquez, L., Eds.; Association for Computational Linguistics: Stroudsburg, PA, USA, 2019; pp. 1504–1519. [CrossRef]
20. Yang, G.; Huang, X.; Hao, Z.; Liu, M.; Belongie, S.J.; Hariharan, B. PointFlow: 3D Point Cloud Generation With Continuous Normalizing Flows. In Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Republic of Korea, 27 October–2 November 2019; pp. 4540–4549. [CrossRef]
21. Sendera, M.; Tabor, J.; Nowak, A.; Bedychaj, A.; Patacchiola, M.; Trzcinski, T.; Spurek, P.; Zieba, M. Non-Gaussian Gaussian Processes for Few-Shot Regression. In Proceedings of the Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, Virtual, 6–14 December 2021; pp. 10285–10298.

22. Grathwohl, W.; Chen, R.T.Q.; Bettencourt, J.; Sutskever, I.; Duvenaud, D. FFJORD: Free-Form Continuous Dynamics for Scalable Reversible Generative Models. In Proceedings of the 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, 6–9 May 2019.
23. Carlisle, M. Racist Data Destruction? 2019. Available online: https://medium.com/@docintangible/racist-data-destruction-113e3eff54a8 (accessed on 7 October 2023).
24. McInnes, L.; Healy, J. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *arXiv* **2018**, arXiv:1802.03426.
25. Li, L.; Jamieson, K.G.; DeSalvo, G.; Rostamizadeh, A.; Talwalkar, A. Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization. *J. Mach. Learn. Res.* **2017**, *18*, 185:1–185:52.
26. Hutter, F.; Hoos, H.H.; Leyton-Brown, K. An Efficient Approach for Assessing Hyperparameter Importance. In Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21–26 June 2014; JMLR Workshop and Conference Proceedings; Volume 32, pp. 754–762.

Modeling Uncertainty in Personalized Emotion Prediction with Normalizing Flows

Piotr Miłkowski[†1], Konrad Karanowski[†1], Patryk Wielopolski[1], Jan Kocoń[1], Przemysław Kazienko[1], Maciej Zięba[1,2]
[1] *Department of Artificial Intelligence, Wrocław University of Science and Technology, Poland*
[2] *Tooploox, ul. Tęczowa 7, 53-601 Wrocław, Poland*
{piotr.milkowski,konrad.karanowski,patryk.wielopolski,jan.kocon,kazienko,maciej.zieba}@pwr.edu.pl

***Abstract*—Designing predictive models for subjective problems in natural language processing (NLP) remains challenging. This is mainly due to its non-deterministic nature and different perceptions of the content by different humans. It may be solved by Personalized Natural Language Processing (PNLP), where the model exploits additional information about the reader to make more accurate predictions. However, current approaches require complete information about the recipients to be straight embedded. Besides, the recent methods focus on deterministic inference or simple frequency-based estimations of the probabilities. In this work, we overcome this limitation by proposing a novel approach to capture the uncertainty of the forecast using conditional Normalizing Flows. This allows us to model complex multimodal distributions and to compare various models using negative log-likelihood (NLL). In addition, the new solution allows for various interpretations of possible reader perception thanks to the available sampling function. We validated our method on three challenging, subjective NLP tasks, including emotion recognition and hate speech. The comparative analysis of generalized and personalized approaches revealed that our personalized solutions significantly outperform the baseline and provide more precise uncertainty estimates. The impact on the text interpretability and uncertainty studies are presented as well. The information brought by the developed methods makes it possible to build hybrid models whose effectiveness surpasses classic solutions. In addition, an analysis and visualization of the probabilities of the given decisions for texts with high entropy of annotations and annotators with mixed views were carried out.**

***Index Terms*—artificial neural networks, natural language processing, human profile modelling, probabilistic technique**

I. Introduction

Human affective states, including emotions, strongly depend on the individual, the stimulant eliciting them, and the associated context [1]. Therefore, the reasoning of a person's perception based on machine learning bears a significant degree of uncertainty. It refers to the reaction to any content, including text reading. We can say that disagreements in human textual inferences are inherent [2]. Most solutions to subjective problems in natural language processing (NLP), like recognition of emotions, hate speech, sarcasm, sense of humor, sentiment, and many others, rely on generalized perspectives. They consider only text and its single generalized interpretation. Then, the commonly used solution is to simplify multiple distinct views, i.e., annotations provided by many annotators using majority voting or other methods to achieve a sole perception. Overall, we can identify two sources of uncertainty: (1) humans, who are unsure and imprecise in their annotations (this is a hidden factor), and (2) a community of annotators. The latter refers to discrepancies between people in understanding the problem, and perception of a given text [3]–[5]. The standard measures for inter-rater agreement are Krippendorff's alpha [6] or Fleiss' kappa [7]. However, they provide only a single value characterizing the set of all annotations for all texts. Yet another (3) source of uncertainty: the trained model itself. It means that the model is not capable of precisely learning about concepts (what is *joy* or *hate speech*?) and relations from the available learning samples. This leads to errors and proximate reasoning. Simultaneously, emotions can be considered multidimensional objects, which requires multi-task learning [8] and further complicates the problem of uncertainty modeling. Most of the proposed approaches for subjective modeling in the NLP domain focus on deterministic predictions. In this work, we propose to enrich the family of emotional methods by introducing Emotional Normalizing Flow – an entirely probabilistic framework that utilizes conditional Normalizing Flows to model uncertainty. We postulate to represent the considered tasks as multivariate regression problems and represent the distribution of the outputs with conditional flows. This approach allows us to model complex multimodal distributions of multidimensional outputs. The experiments and validation were carried out on emotion detection (ten tasks) and hate speech (two tasks). We examine various choices of flow models and compare their performance with the mixture of Gaussians, showing the superiority of Emotional Normalizing Flow compared to the selected baseline. Moreover, we show that incorporating personalization into our model leads to better distribution adjustment measured with negative log-likelihood (NLL) value.

To summarize, the contributions of this work are as follows:

- We introduce a novel approach for probabilistic modeling in subjective NLP-based problems;
- We examine the impact of personalization on the quality of the model and show that in most of the considered experimental cases, additional information about the reader leads to better probability adjustment;
- We show that our approach outperforms the standard baseline that utilizes a mixture of Gaussians;

[†]These authors contributed equally to this work.

- We propose a hybrid approach utilizing Normalizing Flows and personalization that outperform previous models.

II. Related work

Initial work on emotion recognition in the text was based mainly on frequency analysis of words defined in lexicons of emotions [9], [10]. These lexicons contained words with assigned categories of basic emotions, e.g., joy, anger, sadness [11], [12]. Emotions occurring most frequently at the lexical level were then assigned to the entire text. With the development of text classification methods based on machine learning, datasets containing texts manually annotated with emotions began to emerge [13]–[20]. Due to annotators' subjective perception of emotions, and thus low inter-annotator agreement, it was common to assign emotion labels to text based on majority voting [13], [18]. Based on such prepared data, text classification models were trained. Initially, such models as SVM [21], BiLSTM, and GRU [22] were used. Currently, transformer-based models such as BERT perform best in the task of emotion recognition [18], [23]. The aforementioned approaches require data for which the inter-annotator agreement is high. However, there are some data sets such as Wiki-Detox [24], Sentimenti [16] or Measuring Hate Speech [25], which contain an annotator identifier linked to their affective annotation. They also include multiple annotations for a given text from multiple annotators. For such data, new personalized approaches have recently been developed, in which the context of the annotator is taken into account in the model learning process [26]–[36]. This makes it possible, for example, to answer the question of what emotions a particular text evokes for a particular user. Recent method proposals also focus on neuro-symbolic approaches to explain decisions made [37], usage of large-scale pre-trained language model (PLM) for prompt-based classification tasks such as sentiment analysis and emotion detection [38], using recently popular large language models (LLMs) [39], or methods of complex persona attribute extraction [40]. However, the methods mentioned above do not model the uncertainty associated with the community's subjective perception of emotions and the degree of indecision of the annotators themselves.

In this paper, to model uncertainty described in the Introduction we adapt the concept of Normalizing Flows. The best-known Normalizing Flow models such as NICE [41], RealNVP [42], MAF [43], and CNF [44] were originally used for density estimation and image generation tasks. These models were further extended and used as components for more sophisticated tasks or even for other domains of applications. In Computer Vision, there were proposed models such as RegFlow [45] for probabilistic future location prediction, Flow Plugin Network [46], PluGeN [47], and StyleFlow [48] models for conditional image generation. For the tabular data, recently, TreeFlow [49] was proposed that utilizes a combination of tree-based models with conditional Normalizing Flows to estimate uncertainty for uni- and multivariate regression problems. In terms of Natural Language Processing and Normalizing Flows, only Discrete Flow [50] was proposed to model character-level datasets using Normalizing Flows dedicated to the discrete data. To the best of our knowledge, no probabilistic approach has been proposed to model distributions of uncertainty in personalized natural language processing, and our Emotional Normalizing Flow is the first probabilistic model proposed for multi-task prediction of personalized emotions.

III. Background

a) Generalized and Personalized Approach to Subjective NLP Problems.: In the classic approach to the task of text classification or regression, we assume a training set of the form $\mathcal{D} = \{(\mathbf{t}_i, \mathbf{y}_i)\}_{i=1}^{N}$, where $\mathbf{t}_i \in \mathcal{T}$ is the i-th text document and $\mathbf{y}_i$ is its annotation. However, many NLP tasks, such as recognizing emotions in a text or detecting hate speech, can be subjective because each person perceives these phenomena. This leads to a situation when we can have more than one annotation $\mathbf{y}$ for the same text $\mathbf{t}$, as different people may annotate the same texts differently. Therefore, a training set is in the form of $\mathcal{D} = \{(\mathbf{t}_i, \mathbf{p}_i, \mathbf{y}_i)\}_{i=1}^{N}$, where $\mathbf{y}_i$ is the annotation given by person $\mathbf{p}_i \in \mathcal{P}$ for text $\mathbf{t}_i \in \mathcal{T}$.

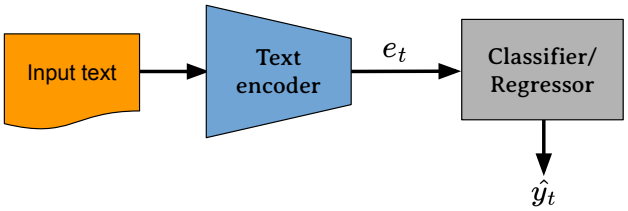
One approach to subjective tasks in NLP is the so-called *generalized*. It assumes that the model predicts the result based solely on the text and returns the same prediction for every user. Generalized models usually consist of two parts: **text encoder** (language model), which creates **text representation** e_t and **classifier** or **regressor** (usually fully-connected layer) that gives prediction $\hat{y}$. However, recent studies [28], [51], [52] show that this approach should not be considered correct, as adding information about the annotator significantly improves model quality and yields better results. The approach that combines information about the text and the human is so-called *personalized*. Compared to the generalized, personalized model adds another component called **profile extractor**, that creates **human representation** e_p. The comparison of generalized and personalized approaches is shown in Fig 1.

There are few existing architectures [28], [51] utilizing this fact. Still, all of them are deterministic, meaning none model uncertainty as a direct optimization of negative log-likelihood.

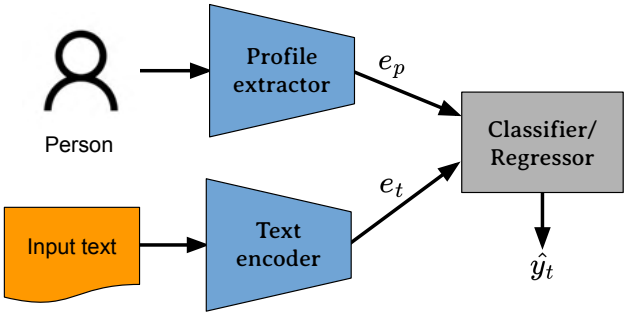
b) Normalizing Flows.: Normalizing Flows [53] are a class of generative models that enables estimation of the uncertainty of prediction thanks to the access to log probability function and thus enable direct optimization of negative log-likelihood (NLL). The goal of the model is to transform base distribution $p_U(\mathbf{u})$ (usually Gaussians with independent components) to the complex distribution of the data $p_Y(\mathbf{y})$ using a series of K invertible functions that can be written as $\mathbf{u} = \mathbf{f}_K \circ \cdots \circ \mathbf{f}_1(\mathbf{y})$. For that purpose, Normalizing Flows utilize the change-of-variable formula and then the NLL $\mathbf{y}$ is given by

$$\log p_Y(\mathbf{y}) = \log p_U(\mathbf{u}) - \sum_{k=1}^{K} \log \left| \det \frac{\partial \mathbf{f}_n}{\partial \mathbf{z}_{k-1}} \right|. \qquad (1)$$

To specify the exact Normalizing Flow model, we need to define transformations $\mathbf{f}_1, \ldots, \mathbf{f}_K$. Here, multiple models were

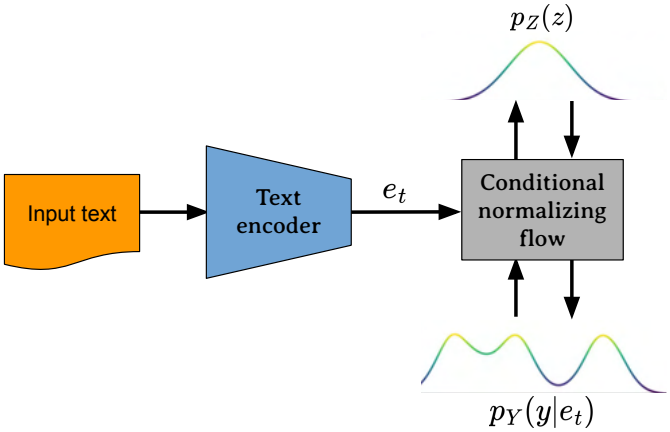


(a) Generalized deterministic model.

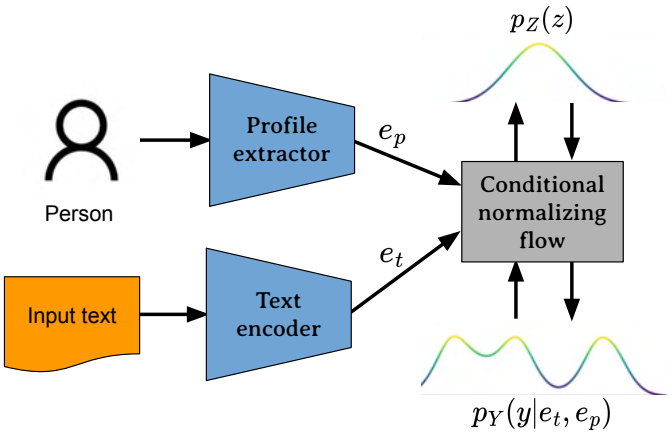


(b) Personalized deterministic model.

Fig. 1: Comparison of (a) generalized and (b) personalized deterministic models. (a) consists of two parts: a text encoder (language model) that creates text embedding e_t and a classifier or regressor (mostly fully-connected layer) that provides prediction $\hat{y}$. This approach is not considered suitable for subjective NLP tasks, like emotion recognition, because it does not respect the individual perception of the text. Model (b) fixes this problem by adding a profile extractor in the form of user representation e_p. It allows human individual characteristics to be included in the inference process. Both models are deterministic, giving us limited, spolight information about subjective tasks.



(a) Generalized flow-based probabilistic model.



(b) Personalized flow-based probabilistic model.

Fig. 2: Comparison of (a) generalized and (b) personalized flow-based probabilistic models. Model (a), as in the case of generalized deterministic, uses only information about the text. However, unlike it, it models the conditional probability distribution $p_Y(\mathbf{y}|\mathbf{e}_t)$ using Normalizing Flow. Model (b) extends the concept of the personalized deterministic model in a similar way to (a) but it exploits representations of both the text e_t and user e_p to model conditional probability distribution $p_Y(\mathbf{y}|\mathbf{e}_p, \mathbf{e}_t)$ for the subjective output predictions representing emotions.

proposed such as NICE [41], RealNVP [42], MAF [43] or Continuous Normalizing Flows [44].

IV. Our approach

In this section, we introduce Emotional Normalizing Flow - the probabilistic model for subjective uncertainty modeling in the NLP domain. The general schema of the proposed approach is provided in Fig. 2. The model is composed of *Profile extractor* that is responsible for creating the representation of the person, $\mathbf{e}_p$, and *Text encoder* that creates embedding $\mathbf{e}_t$ directly from the input text. Both components can be represented by various models (trainable and fixed), and we elaborate on this further in this section.

The extracted vectors $\mathbf{e}_p$ and $\mathbf{e}_t$ are further delivered to the conditional flow represented by the complex transformation function $\mathbf{f}(\cdot)$. The role of the function is to transform multivariate regression outputs $\mathbf{y}$ to $\mathbf{z}$ that represents the variable in the base space, assuming given vectors, $\mathbf{e}_p$ and $\mathbf{e}_t$. Formally, we have $\mathbf{z} = \mathbf{f}(\mathbf{y}, \mathbf{e}_p, \mathbf{e}_t)$, where $\mathbf{f}$ is invertible with respect to $\mathbf{y}$, $\mathbf{y} = \mathbf{f}^{-1}(\mathbf{z}, \mathbf{e}_p, \mathbf{e}_t)$. Moreover, the complex transformation $\mathbf{f}$ can be decomposed into a sequence of simple functions, $\mathbf{z} = \mathbf{f}_K \circ \cdots \circ \mathbf{f}_1(\mathbf{y}, \mathbf{e}_p, \mathbf{e}_t)$, where the K is number discrete transformations. With such assumptions, the probability distribution for $\mathbf{y}$ that represents the distribution over the regression outputs can be calculated using the formula:

$$p_Y(\mathbf{y}|\mathbf{e}_p, \mathbf{e}_t) = p_Z(\mathbf{z}) \cdot \prod_{k=1}^{K} \left| \det \frac{\partial \mathbf{f}_k}{\partial \mathbf{z}_{k-1}} \right|, \qquad (2)$$

where $\mathbf{z}_0, \ldots, \mathbf{z}_K$ are intermediate steps after discrete transformations, assuming $\mathbf{z}_0 := \mathbf{y}$, and $\mathbf{z}_K := \mathbf{z}$. $p_Z(\mathbf{z})$ is the assumed base distribution for $\mathbf{z}$ with the known density

function, usually represented by Gaussian. Consequently, we have direct access to the density function for that conditional distribution. Therefore we can calculate the likelihood function for a set of input-output pairs to evaluate the quality of the model. We can sample an infinite number of output values assuming given inputs and interpret the results.

The proposed model can quickly adapt to the problems without personalization, simply skipping $\mathbf{e}_p$ conditioning in the flow. Our approach is independent of the conditional Normalizing Flow type, and we experimentally compare the performances of the most popular models. We follow the methodology of incorporating conditional components described in [46].

a) Profile extractor.: Vector $\mathbf{e}_p$ contains information about the user specific to the personalization architecture used. This can include information such as the deviation of responses from the majority voice, metadata about the user, user identifier [28], the correlation of the text's context with historical evaluations, or other features unique to the recipient of the text. It also can be randomly initialized and tuned during the learning process by backpropagation [51].

b) Text encoder.: In the case of $\mathbf{e}_t$ vector, text representation is implemented using Transfomer language models. An attentional weight is assigned for a given text input, divided into individual tokens. The assigned values are then used to calculate the weighted sum of the resulting vectors [54]. It is possible to fine-tune the language model using the loss function of the final model.

c) Training the model.: To trained Emotional Normalizing Flow we use the dataset $\mathcal{D} = \{(t_n, p_n, \mathbf{y}_n)\}_{n=1}^{N}$, composed of t_n textual input, p_n features of the person, and corresponding subjective annotation y_n given by the person p_n for text t_n. We train our model directly by optimizing the negative log-likelihood function:

$$\mathcal{L} = -\sum_{n=1}^{N} \log p_Y(\mathbf{y}_n | \mathbf{e}_{n,p}, \mathbf{e}_{n,t}), \tag{3}$$

where $\mathbf{e}_{n,p}$ is a vector, that represents profile of the person p_n, and $\mathbf{e}_{n,t}$ is an embedding of the text t_n. The model can be trained in a two-stage mode or end-to-end paradigm depending on the form of *Profile extractor* and *Text encoder*. In the first case, the embeddings $\mathbf{e}_p$ and $\mathbf{e}_t$ are extracted in the first stage, and parameters of the flow are trained while optimizing $\mathcal{L}$. Alternatively, suppose the *Profile extractor* or *Text encoder* are represented by differentiable architectures. In that case, the entire system can be optimized end-to-end, directly minimizing the negative log-likelihood function.

V. Experiments

In this subsection, we evaluate our approach on a set of challenging datasets, investigating the impact of adding contextual information about the person in the model. Moreover, we compare flow-based probabilistic models to a simple Gaussian Mixture Model. Then, we compare our solution to the deterministic models using sampling from flow and discretization. Finally, we mix deterministic and probabilistic approaches to create a hybrid model.

A. Datasets

a) Wikipedia-Detox.: The Wikipedia Detox project has created a crowd-sourced dataset that contains one million annotations covering 100,000 discussions of page edits on Wikipedia [24]. These were often filled with toxic statements, verbal aggression, and even personal attacks. Each comment was annotated by about ten annotators provided by the Crowdflower service.

The collection containing toxic statements consists of 160,000 texts. It includes a binary determination of toxicity (where: 0 = non-toxic, 1 = toxic), as well as a rating from -2 to 2 (where: 2 = very healthy, 0 = neutral, and -2 = very toxic).

Sets for personal attacks and verbal aggression consist of 100,000 of the same comments. In addition to the binary marks for aggression (0 = neutral or friendly comment, 1 = aggressive or attacking), aggression is put on a scale analogous to toxicity from -2 to 2 (where: 2 = very friendly, 0 = neutral, and -2 = very aggressive). Personal attacks are divided into types: quoting, recipient, third party, or another type of attack. In addition to texts shared between these collections, the same applies to annotators. Thus, we can use knowledge from one collection to benefit from it in another or a collective approach. Those willing to participate in the study also completed questionnaires so that we have demographic information about them available.

b) Emotion Simple.: This collection consists of 100 texts marked on 10 scales by 5,365 annotators [55]. Texts are opinions posted on websites. This gives 53.65 annotations per text and 1.69 markings from a single user. The texts were rated for eight basic emotions (sadness, anticipation, joy, fear, surprise, disgust, trust, and anger) and emotional arousal on a scale from 0 to 4 for each dimension. In addition, the tenth aspect rated is the valence expressed on a scale from -3 to 3 (where -3 = negative, 0 = neutral, and 3 = positive). In the set of individuals with two marks, those with three or more annotations also appear.

c) Emotion Meanings.: In [56], a huge collection containing 6,000 assessed word collocations was prepared and published. It contains dimensions and scales analogous to the Emotion Simple collection – the basic emotions from Robert Plutchik's Wheel of Emotions [57].

The scale of the collection makes it one of the most interesting and, simultaneously, the most difficult for personalizing emotion detection. It has 303,143 annotations from 16,101 people who participated in the study. Each collocation has been evaluated 50.67 times, and a single annotator has an average of 18.83 annotations.

The difficulty in working with these data is also because these are not full-fledged textual statements containing context but just two words. An example item from the collection: "colorful beads". Annotator data include information such as

gender, age, education, size of residence, relationship status, income, or political views.

B. Setups

The dataset was divided into training, validation, and testing splits. Users and texts were not mixed between sets to bring the evaluation as close to the real-world scenario. Each experiment consisted of 10-fold cross-validation, and obtained results were averaged. Statistical significance tests were performed: t-test with Bonferroni correction to address the problem of multiple comparisons. In the tables within the rows, comparisons were made between models without and with personalization. Bold indicates the best result, and underline indicates the absence of statistically significant differences for each dataset. Within the "Type" column, the best probabilistic model type or no significant difference between the two was similarly marked for each dataset separately.

a) Baselines.: We have three reference points. To check the impact of personalization, we compared personalized models with a baseline that uses only textual information (TXT-Baseline); it is a generalized approach. To investigate the impact of normalizing flows, we compared them with a Gaussian Mixture Model to have a reference point in the form of another, less complex probabilistic method. Finally, we compared our method with deterministic approaches.

b) Models for conditional normalizing flows.: In our experimental evaluation, we consider Emotional Normalizing Flow with various types of conditional normalizing flows. For single-dimensional datasets: *Wikipedia Detox: Toxicity*, *Wikipedia Detox: Aggression* and *Wikipedia Detox: Attack*, we used MAF (**maf**) and CNF (**cnf**). For multi-dimensional *Emotions Meanings* and *Emotions Simple*, we used two extra flows: RealNVP (**real_nvp**) and NICE (**nice**). We compared the results against the baseline that uses mixtures of Gaussians to model the probability (**gmm**).

c) Models for personalization.: We investigate three approaches to respect the personalization context: OneHot, HuBi-Formula, and HuBi-Medium [51]. They are confronted with TXT-Baseline (generalized, non-personalized) that does not contain any information about the annotator. We exploit LaBSE [58] as a language model in every experiment.

C. Experimental scenarios

a) Experiment 1 - Comparison of generalized and personalized solutions in the probabilistic approach.: The first approach verifies the performance of Emotional Normalizing Flow with fixed hyperparameters on multiple data sets and tasks: *Wikipedia Detox: Toxicity*, *Wikipedia Detox: Aggression*, *Wikipedia Detox: Attack*, *Emotion Meanings* and *Emotion Simple*. We also verified the ability of the proposed Emotional Normalizing Flow to transfer knowledge between thematically similar multidimensional text labels. For this purpose, the *Wikipedia Detox: Aggression* and *Wikipedia Detox: Attack* datasets were joined, as they contain annotations for the same texts performed by the same annotators. As a result, we obtained a dataset with multi-dimensional labels. This experiment aimed to examine the effect of personalization models on the prediction of probability distributions, thus verifying whether the additional information provided to the model reduces its uncertainty and comparing Normalizing Flows to Gaussian Mixture Model.

b) Experiment 2 - Investigating the effect of hyperparameters selected for personalization and Emotional Normalizing Flow methods on the most difficult dataset.: The second approach was to verify the maximum possible reduction of model uncertainty by tuning the model hyperparameters to a given set and checking which normalizing-flow model obtained the best results. Due to limited resources, we decided to perform this experiment using only *Emotion Meanings* dataset. The parameters that we tuned were: the number of hidden features, number of layers, number of blocks per layer, dropout probability, batch normalization within layers, batch normalization between layers, learning rate, the size of hidden layers used to prepare user embeddings, and the size of the output of these embeddings.

c) Experiment 3 - Comparison of probabilistic and deterministic approaches.: To compare with classical methods [8], which are deterministic, it was necessary to prepare conversions of the Emotional Normalizing Flow output to the form of exact values. Included in the body of the paper is the application of two best normalizing flows (RealNVP and CNF) for multidimensional datasets (*Aggression & Attack* [classification task] and *Emotion Simple* [regression task]).

For the first type of task, each text or text-user pair was sampled using an iterative method. In the preparation step, we increased the number of samples in the test part of the dataset so that the value from 0 to 1 with a step of 0.1 for the class was tested as a possible context. Iterations were done twice for values of 0 and 1 in the opposite class. Next, an exponential was applied to the 44 probabilities of the resulting sample (22 per class for each text). Within the values for the opposite sampling, (e.g., [0.5, 0] and [0.5, 1]) of a given dimension were summed, and then for each stopper (0.0, 0.1, ..., 1.0) divided by the sum of all values for the dimension. If the probability mass prevailed on the side from 0.0 to 0.5, it was considered that the class was not assigned and vice versa for the other part of the axis.

It was impossible for a 10-dimensional set for the regression task to sample each possible dimension in all values separately because of the number of possible combinations. Each item from the test subset was replicated 100 times containing random real values from 0 to 1 in each class. Majority voting was then conducted to determine the most likely response for the scale of each dimension. In the collection, each dimension had a value analogous to the slider setting during annotation. For this reason, the task was treated as an ordinary regression, and the resulting values were rounded to the nearest possible position. This assumption was used for both values from the deterministic and probabilistic approaches.

d) Experiment 4 - Hybrid approach (utilizing knowledge from the text and uncertainty modeling).: The combined approach, hereafter referred to as hybrid, was done in two

steps. In the first, the learned Emotional Normalizing Flow models were sampled in the same way as in Experiment 3, but for all the texts in the collection. Then, the network input was extended to the deterministic model with an additional feature. A vector containing the resulting probabilities for each text was entered along with its embeddings and, in the case of approaches with personalization, the user profile. This vector contained all the values from the sampling, and no additional mathematical operations were performed on it.

D. Results

a) Results of Experiment 1.: The first experiment proved that adding personalization reduces the uncertainty of probabilistic models, Tab. I. For *Wikipedia Detox* datasets (*Aggression*, *Attack* and *Toxicity*), all personalized models received significantly lower negative log-likelihood values compared to the non-personalized TXT-Baseline. For all three tasks, the best architecture was HuBi-Medium combined with CNF. For *Aggression & Attack* dataset, personalization improved most cases' results. The best results were obtained by OneHot combined with RealNVP and HuBi-Formula combined with CNF. In the case of *Emotion Simple* and *Emotion Meaning*, personalization also reduced model uncertainty in most of the cases. For both datasets, the best results were obtained by the HuBi-Medium model combined with RealNVP. It is worth noting that compared to Gaussian Mixture Model, Normalizing Flows always obtain lower negative log-likelihood values. It suggests that target variables, i.e., emotions, have complex distributions, and using a simple probabilistic approach is not enough.

b) Results of Experiment 2.: In the second experiment, we carried out hyperparameter tuning on the most challenging dataset: *Emotion Meanings*, Tab. II. All possible combinations of hyperparameters were considered when performing the grid search. The results seem to confirm earlier speculations about MAF's better ability to deal with multidimensional problems compared to other approaches. Moreover, none of the variants indicated the benefit of using text alone as input.

c) Results of Experiment 3.: In the third experiment, we compared results obtained by deterministic models and Emotional Normalizing Flow, Tab. III. It was carried out on two datasets: combined *Wikipedia Detox: Aggression & Attack* and *Emotions Simple*. We also decided to use only two Normalizing Flow Models that performed the best on both of these datasets: RealNVP and CNF.

In the case of *Aggression & Attack* dataset, the results obtained by deterministic models were better for every architecture, including the non-personalized one. In the case of *Emotion Simple* dataset, the results obtained by probabilistic models significantly outperformed deterministic models. The best model was a combination of HuBi-Medium and CNF. This result seems to confirm that the probabilistic approach is especially effective on complex multi-dimensional tasks.

d) Results of Experiment 4.: In the fourth experiment, we mixed the deterministic approach with the probabilistic, to create a hybrid model, Tab. IV. In both *Aggression & Attack* and *Emotion Simple* tasks, the results obtained by the hybrid approach outperformed previous methods by a large margin. For *Aggression & Attack* dataset, the best model turned out to be HuBi-Medium with CNF. For *Emotion Simple* dataset, HuBi-Medium with both RealNVP and CNF performed comparably well. The results of this experiment prove, that adding information about the model uncertainty makes big difference in the inference process, and helps to predict better for difficult and subjective tasks.

Dataset	Type	TXT-Baseline	OneHot	HuBi-Formula	HuBi-Medium
Toxicity	maf	0.0702	-0.0197	**-0.0947**	0.0053
	cnf	0.1231	-0.0707	-0.1202	**-0.1378**
	gmm	0.6422	0.6164	0.5829	**0.5072**
Aggression	maf	0.1705	0.1695	**0.0526**	0.0859
	cnf	0.1685	0.0978	0.0180	**-0.0431**
	gmm	0.8948	0.7783	0.7841	**0.7446**
Attack	maf	0.1669	0.1180	-0.0229	-0.0250
	cnf	0.1474	0.0811	-0.0427	**-0.0950**
	gmm	0.7512	0.7318	0.7105	**0.6911**
Aggression & Attack	maf	**-1.3788**	N/A	-0.3966	-0.3834
	nice	-0.8678	**-1.1281**	-1.0524	-1.0914
	real_nvp	-3.3482	**-3.6181**	-3.0235	-1.7208
	cnf	-3.5113	-2.7339	**-3.7002**	-2.1357
	gmm	3.1729	**2.4028**	2.5673	2.6858
Emotion Meanings	maf	0.5337	**-0.0135**	0.8525	0.1936
	nice	-1.9099	-1.8707	**-2.0283**	-1.4792
	real_nvp	-5.4775	-2.9377	-5.5189	**-5.6377**
	cnf	-3.7186	-1.9640	-3.4632	**-4.8458**
	gmm	5.9559	5.4858	4.8034	**4.4719**
Emotion Simple	maf	1.9130	2.6398	2.6393	1.9314
	nice	2.4254	**1.8163**	2.3502	1.9613
	real_nvp	2.5583	1.7845	2.3726	**1.4355**
	cnf	2.1220	**1.8197**	2.3347	1.9702
	gmm	4.2706	**3.7496**	4.2312	4.0910

TABLE I: Experiment 1 - negative log-likelihood values for all datasets, without hyperparameter tuning.

Dataset	Type	TXT-Baseline	OneHot	HuBi-Formula	HuBi-Medium
Emotion Meanings	**maf**	-11.5380	**-14.0785**	-10.6476	-12.1934
	nice	4.2167	**-2.0243**	-1.0978	-1.9208
	real_nvp	-2.3509	-4.5813	-5.2833	**-7.0285**
	cnf	-1.9623	-3.9381	**-5.2247**	-4.9381
	gmm	12.6564	9.6047	**7.8908**	8.4545

TABLE II: Experiment 2 - negative log-likelihood values for Emotion Meanings dataset, with hyperparameter tuning.

VI. Conclusions

In this paper, we proposed a novel Emotional Normalizing Flow approach to personalized NLP that opens up new perspectives on predicting reader behavior in a non-deterministic way. From the perspective of psychology and the variability of emotion sensation over time, the problem of emotion recognition is one of the most difficult and subjective tasks facing NLP. People do not perceive their emotions as zero-one, and most of the attempts so far classified their feelings in this way. The presented probabilistic approach based on normalizing flows provides more complex information about the uncertainty and diversity of possible emotional states. A comparative analysis of models for emotion recognition without and with personalization indicated that new methods are also effectively applicable in a non-deterministic setup. The generalized, non-personalized solution generates a completely different concentration of probability mass, directed toward a

Dataset	Method	TXT-Baseline	OneHot	HuBi-Formula	HuBi-Medium
Aggression & Attack [Macro F-1]	deterministic	0.5874	0.5954	0.7354	**0.7847**
	discrete(real_nvp)	0.4829	0.4682	0.5860	**0.6788**
	discrete(cnf)	0.5189	0.4738	0.6397	**0.7374**
Emotion Simple [R2]	deterministic	0.3936	0.5403	0.5574	**0.5822**
	discrete(real_nvp)	0.4472	0.6535	0.6582	**0.6835**
	discrete(cnf)	0.4428	0.6274	0.6685	**0.7005**

TABLE III: Experiment 3 - Comparison of classification and regression using the classical deterministic method and the result of sampling Emotional Normalizing Flow. Macro F-1 score for *Aggression & Attack* and R-squared for *Emotion Simple* datasets.

Dataset	Method	TXT-Baseline	OneHot	HuBi-Formula	HuBi-Medium
Aggression & Attack [Macro F-1]	deterministic	0.5874	0.5954	0.7354	**0.7847**
	hybrid(real_nvp)	0.8479	0.8254	0.8233	**0.8743**
	hybrid(cnf)	0.8693	0.9052	0.8400	**0.9553**
Emotion Simple [R2]	deterministic	0.3936	0.5403	0.5574	**0.5822**
	hybrid(real_nvp)	0.4722	0.7149	0.7237	**0.7376**
	hybrid(cnf)	0.4867	0.6899	0.69223	**0.7388**

TABLE IV: Experiment 4 - Comparison of the classical deterministic approach and hybrid models, which in addition use probabilistic knowledge. Macro F-1 score for *Aggression & Attack* and R-squared for *Emotion Simple* datasets.

quantitative approach. Personalization can shift the view of the problem in a contextual way by dedicating reasoning to a single user. Finally, we showed that adding information about model uncertainty significantly improves the ability to predict complex and subjective behaviors such as recognizing hate speech or emotions in a text. The hybrid model we created significantly outperformed the previous methods, becoming a new state-of-the-art on two very challenging tasks. Our future work will focus on applications of our approach to some other tasks such as active or reinforcement learning.

LIMITATIONS

One important issue related to the nature of normalizing flows is their ability to convert probabilities to disambiguate uncertain answers. At the moment, there are no reference datasets available that contain text and annotator information simultaneously with multiple markings of the same text by the same person. This is due to cost constraints in preparing such data. However, we have conducted experiments on datasets with different characteristics in which (1) one person marked several hundred texts [*Wikipedia Detox Datasets*] and (2) one text was evaluated dozens of times by different people [*Emotions Simple* and *Emotion Meanings* datasets]. In order to address the problem mentioned in the introduction, one text should have N annotations from the same person, e.g., a few days apart. If we gain access to or prepare such a dataset, we would be happy to conduct in-depth studies on it.

Due to the language of one of the datasets being different from English, a multilingual model was used to embed the text. This decision was made in order to allow for direct comparisons and cross-referencing. This would have added an unnecessary layer to the already relatively complex problem that was addressed. It is possible to experiment with other language models as well using the source codes provided[†].

ACKNOWLEDGMENTS

This work was financed by (1) the National Science Centre, Poland, project no. 2021/41/B/ST6/04471; (2) Contribution to the European Research Infrastructure 'CLARIN ERIC - European Research Infrastructure Consortium: Common Language Resources and Technology Infrastructure', 2022-23 (CLARIN Q); (3) the Polish Ministry of Education and Science, CLARIN-PL; (4) the European Regional Development Fund, as a part of the 2014-2020 Smart Growth Operational Programme, projects no. POIR.04.02.00-00C002/19, POIR.01.01.01-00-0923/20, POIR.01.01.01-00-0615/21, and POIR.01.01.01-00-0288/22; (5) the statutory funds of the Department of Artificial Intelligence, Wroclaw University of Science and Technology; (6) the Polish Ministry of Education and Science within the programme "International Projects Co-Funded"; (7) the European Union under the Horizon Europe, grant no. 101086321 (OMINO). However, the views and opinions expressed are those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Executive Agency. Neither the European Union nor European Research Executive Agency can be held responsible for them. The work conducted by Maciej Zieba and Patryk Wielopolski was supported by the National Centre of Science (Poland) Grant No. 2021/43/B/ST6/02853.

REFERENCES

[1] L. F. Barrett, *How emotions are made: The secret life of the brain*. Pan Macmillan, 2017.

[2] E. Pavlick and T. Kwiatkowski, "Inherent Disagreements in Human Textual Inferences," *Transactions of the Association for Computational Linguistics*, vol. 7, pp. 677–694, 11 2019.

[3] C. Beck, H. Booth, M. El-Assady, and M. Butt, "Representation problems in linguistic annotations: Ambiguity, variation, uncertainty, error and bias," in *Proceedings of the 14th Linguistic Annotation Workshop*, (Barcelona, Spain), pp. 60–73, Association for Computational Linguistics, Dec. 2020.

[4] A. M. Davani, M. Díaz, and V. Prabhakaran, "Dealing with disagreements: Looking beyond the majority vote in subjective annotations," *Transactions of the Association for Computational Linguistics*, vol. 10, pp. 92–110, 2022.

[5] E. Troiano, S. Padó, and R. Klinger, "Emotion ratings: How intensity, annotation confidence and agreements are entangled," in *Proceedings of the Eleventh Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, (Online), pp. 40–49, Association for Computational Linguistics, Apr. 2021.

[6] K. Krippendorff, "Computing krippendorff's alpha-reliability," *Annenberg School for Communication Departmental Papers: Philadelphia*, 2011.

[7] J. L. Fleiss, "Measuring nominal scale agreement among many raters.," *Psychological bulletin*, vol. 76, no. 5, p. 378, 1971.

[8] P. Miłkowski, S. Saganowski, M. Gruza, P. Kazienko, M. Piasecki, and J. Kocoń, "Multitask personalized recognition of emotions evoked by textual content," in *EmotionAware 2022: Sixth International Workshop on Emotion Awareness for Pervasive Computing Beyond Traditional Approaches at PerCom 2022*, (online), pp. 347–352, mar 2022.

[9] C. Strapparava and R. Mihalcea, "Learning to identify emotions in text," in *Proceedings of the 2008 ACM symposium on Applied computing*, pp. 1556–1560, 2008.

[†] https://github.com/CLARIN-PL/personalized-emotion-prediction-with-normalizing-flows

[10] F. S. Tabak and V. Evrim, "Comparison of emotion lexicons," in *2016 HONET-ICT*, pp. 154–158, IEEE, 2016.

[11] R. Plutchik, "A general psychoevolutionary theory of emotion," in *Theories of emotion*, pp. 3–33, Elsevier, 1980.

[12] P. Ekman, "An argument for basic emotions," *Cognition & emotion*, vol. 6, no. 3-4, pp. 169–200, 1992.

[13] L. A. M. Oberländer and R. Klinger, "An analysis of annotated corpora for emotion classification in text," in *Proceedings of the 27th International Conference on Computational Linguistics*, pp. 2104–2119, 2018.

[14] J. Kocoń, A. Janz, and M. Piasecki, "Context-sensitive sentiment propagation in wordnet," in *Proceedings of the 9th global wordnet conference*, pp. 329–334, 2018.

[15] J. Kocoń and A. Janz, "Propagation of emotions, arousal and polarity in wordnet using heterogeneous structured synset embeddings," in *Proceedings of the 10th Global Wordnet Conference*, pp. 336–341, 2019.

[16] J. Kocoń, A. Janz, P. Miłkowski, M. Riegel, M. Wierzba, A. Marchewka, A. Czoska, D. Grimling, B. Konat, K. Juszczyk, *et al.*, "Recognition of emotions, valence and arousal in large-scale multi-domain text reviews," in *9th Language & Technology Conference*, 2019.

[17] J. Kocoń, P. Miłkowski, M. Wierzba, B. Konat, K. Klessa, A. Janz, M. Riegel, K. Juszczyk, D. Grimling, A. Marchewka, *et al.*, "Multilingual and language-agnostic recognition of emotions, valence and arousal in large-scale multi-domain text reviews," in *Language and Technology Conference*, pp. 214–231, Springer, 2019.

[18] D. Demszky, D. Movshovitz-Attias, J. Ko, A. Cowen, G. Nemade, and S. Ravi, "Goemotions: A dataset of fine-grained emotions," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4040–4054, 2020.

[19] J. Kocoń, J. Radom, E. Kaczmarz-Wawryk, K. Wabnic, A. Zajączkowska, and M. Zaśko-Zielińska, "Aspectemo: multi-domain corpus of consumer reviews for aspect-based sentiment analysis," in *2021 International Conference on Data Mining Workshops (ICDMW)*, pp. 166–173, IEEE, 2021.

[20] A. Srivastava, A. Rastogi, A. Rao, A. A. M. Shoeb, A. Abid, A. Fisch, A. R. Brown, A. Santoro, A. Gupta, A. Garriga-Alonso, *et al.*, "Beyond the imitation game: Quantifying and extrapolating the capabilities of language models," *Transactions on Machine Learning Research*, 2023.

[21] S. Mohammad, "# emotional tweets," in ** SEM 2012: The First Joint Conference on Lexical and Computational Semantics–Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pp. 246–255, 2012.

[22] M. Abdul-Mageed and L. Ungar, "Emonet: Fine-grained emotion detection with gated recurrent neural networks," in *Proceedings of the 55th annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 718–728, 2017.

[23] C.-C. Hsu and L.-W. Ku, "Socialnlp 2018 emotionx challenge overview: Recognizing emotions in dialogues," in *Proceedings of the sixth international workshop on natural language processing for social media*, pp. 27–31, 2018.

[24] E. Wulczyn, N. Thain, and L. Dixon, "Ex machina: Personal attacks seen at scale," in *Proceedings of the 26th international conference on world wide web*, pp. 1391–1399, 2017.

[25] C. J. Kennedy, G. Bacon, A. Sahn, and C. von Vacano, "Constructing interval variables via faceted rasch measurement and multitask deep learning: a hate speech application," *arXiv preprint arXiv:2009.10277*, 2020.

[26] S. Akhtar, V. Basile, and V. Patti, "Modeling annotator perspective and polarized opinions to improve hate speech detection," in *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, vol. 8, pp. 151–154, 2020.

[27] J. Kocoń, A. Figas, M. Gruza, D. Puchalska, T. Kajdanowicz, and P. Kazienko, "Offensive, aggressive, and hate speech analysis: From data-centric to human-centered approach," *Information Processing & Management*, vol. 58, no. 5, p. 102643, 2021.

[28] F. Mireshghallah, V. Shrivastava, M. Shokouhi, T. Berg-Kirkpatrick, R. Sim, and D. Dimitriadis, "Useridentifier: implicit user representations for simple and effective personalized sentiment analysis," *arXiv preprint arXiv:2110.00135*, 2021.

[29] K. Kanclerz, A. Figas, M. Gruza, T. Kajdanowicz, J. Kocoń, D. Puchalska, and P. Kazienko, "Controversy and conformity: from generalized to personalized aggressiveness detection," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 5915–5926, 2021.

[30] P. Milkowski, M. Gruza, K. Kanclerz, P. Kazienko, D. Grimling, and J. Kocon, "Personal bias in prediction of emotions elicited by textual opinions," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL-IJCNLP 2021): Student Research Workshop*, (Online), pp. 248–259, Association for Computational Linguistics, Aug. 2021.

[31] A. Ngo, A. Candri, T. Ferdinan, J. Kocoń, and W. Korczynski, "Studemo: A non-aggregated review dataset for personalized emotion recognition," in *Proceedings of the 1st Workshop on Perspectivist Approaches to NLP@ LREC2022*, pp. 46–55, 2022.

[32] K. Kanclerz, M. Gruza, K. Karanowski, J. Bielaniewicz, P. Miłkowski, J. Kocoń, and P. Kazienko, "What if ground truth is subjective? personalized deep neural hate speech detection," in *Proceedings of the 1st Workshop on Perspectivist Approaches to NLP@ LREC2022*, pp. 37–45, 2022.

[33] J. Bielaniewicz, K. Kanclerz, P. Miłkowski, M. Gruza, K. Karanowski, P. Kazienko, and J. Kocoń, "Deep-sheep: Sense of humor extraction from embeddings in the personalized context," in *2022 IEEE International Conference on Data Mining Workshops (ICDMW)*, pp. 967–974, IEEE, 2022.

[34] T. Ferdinan and J. Kocoń, "Personalized models resistant to malicious attacks for human-centered trusted ai," in *The AAAI-23 Workshop on Artificial Intelligence Safety (SafeAI 2023)*, CEUR Workshop Proceedings, 2023.

[35] W. Mieleszczenko-Kowszewicz, K. Kanclerz, J. Bielaniewicz, M. Oleksy, M. Gruza, S. Woźniak, E. Dzięcioł, P. Kazienko, and J. Kocoń, "Capturing human perspectives in nlp: Questionnaires, annotations, and biases," in *The ECAI 2023 2nd Workshop on Perspectivist Approaches to NLP*, CEUR Workshop Proceedings, 2023.

[36] J. Kocoń, J. Baran, K. Kanclerz, M. Kajstura, and P. Kazienko, "Differential dataset cartography: Explainable artificial intelligence in comparative personalized sentiment analysis," in *International Conference on Computational Science*, pp. 148–162, Springer, 2023.

[37] E. Cambria, Q. Liu, S. Decherchi, F. Xing, and K. Kwok, "Senticnet 7: A commonsense-based neurosymbolic ai framework for explainable sentiment analysis," in *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pp. 3829–3839, 2022.

[38] R. Mao, Q. Liu, K. He, W. Li, and E. Cambria, "The biases of pre-trained language models: An empirical study on prompt-based sentiment analysis and emotion detection," *IEEE Transactions on Affective Computing*, 2022.

[39] M. M. Amin, R. Mao, E. Cambria, and B. W. Schuller, "A wide evaluation of chatgpt on affective computing tasks," *arXiv preprint arXiv:2308.13911*, 2023.

[40] L. Zhu, W. Li, R. Mao, V. Pandelea, and E. Cambria, "Paed: zero-shot persona attribute extraction in dialogues," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9771–9787, 2023.

[41] L. Dinh, D. Krueger, and Y. Bengio, "Nice: Non-linear independent components estimation," *arXiv*, 2014.

[42] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using Real NVP," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, OpenReview.net, 2017.

[43] G. Papamakarios, T. Pavlakou, and I. Murray, "Masked autoregressive flow for density estimation," 2018.

[44] W. Grathwohl, R. T. Chen, J. Bettencourt, I. Sutskever, and D. Duvenaud, "Ffjord: Free-form continuous dynamics for scalable reversible generative models," *arXiv preprint arXiv:1810.01367*, 2018.

[45] M. Zieba, M. Przewieźlikowski, M. Śmieja, J. Tabor, T. Trzcinski, and P. Spurek, "RegFlow: Probabilistic Flow-based Regression for Future Prediction," *CoRR*, vol. abs/2011.14620, 2020.

[46] P. Wielopolski, M. Koperski, and M. Zieba, "Flow plugin network for conditional generation," *arXiv preprint arXiv:2110.04081*, 2021.

[47] M. Wolczyk, M. Proszewska, L. Maziarka, M. Zieba, P. Wielopolski, R. Kurczab, and M. Smieja, "Plugen: Multi-label conditional generation from pre-trained models," in *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, pp. 8647–8656, AAAI Press, 2022.

[48] R. Abdal, P. Zhu, N. J. Mitra, and P. Wonka, "Styleflow: Attribute-conditioned exploration of stylegan-generated images using conditional continuous normalizing flows," *ACM Trans. Graph.*, vol. 40, no. 3, pp. 21:1–21:21, 2021.
[49] P. Wielopolski and M. Zieba, "Treeflow: Going beyond tree-based gaussian probabilistic regression," *CoRR*, vol. abs/2206.04140, 2022.
[50] D. Tran, K. Vafa, K. K. Agrawal, L. Dinh, and B. Poole, "Discrete flows: Invertible generative models of discrete data," in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada* (H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, eds.), pp. 14692–14701, 2019.
[51] J. Kocoń, M. Gruza, J. Bielaniewicz, D. Grimling, K. Kanclerz, P. Miłkowski, and P. Kazienko, "Learning personal human biases and representations for subjective tasks in natural language processing," in *2021 IEEE International Conference on Data Mining (ICDM)*, pp. 1168–1173, IEEE, 2021.
[52] P. Kazienko, J. Bielaniewicz, M. Gruza, K. Kanclerz, K. Karanowski, P. Miłkowski, and J. Kocoń, "Human-centred neural reasoning for subjective content processing: Hate speech, emotions, and humor," *Information Fusion*, 2023.
[53] D. J. Rezende and S. Mohamed, "Variational Inference with Normalizing Flows," in *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, vol. 37 of *JMLR Workshop and Conference Proceedings*, pp. 1530–1538, JMLR.org, 2015.
[54] J. Vig and Y. Belinkov, "Analyzing the structure of attention in a transformer language model," *arXiv preprint arXiv:1906.04284*, 2019.
[55] P. Miłkowski, M. Gruza, K. Kanclerz, P. Kazienko, D. Grimling, and J. Kocoń, "Personal bias in prediction of emotions elicited by textual opinions," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: Student Research Workshop*, pp. 248–259, 2021.
[56] M. Wierzba, M. Riegel, J. Kocoń, P. Miłkowski, A. Janz, K. Klessa, K. Juszczyk, B. Konat, D. Grimling, M. Piasecki, *et al.*, "Emotion norms for 6000 polish word meanings with a direct mapping to the polish wordnet," *Behavior Research Methods*, pp. 1–16, 2021.
[57] R. Plutchik, *The emotions*. University Press of America, 1991.
[58] F. Feng, Y. Yang, D. Cer, N. Arivazhagan, and W. Wang, "Language-agnostic BERT sentence embedding," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, (Dublin, Ireland), pp. 878–891, Association for Computational Linguistics, May 2022.

APPENDIX

A. Personalization architectures

Architectures for personalization have been modified to return a probability distribution instead of the probability of a class. Input features are passed to the normalizing flow as context. We are dealing with four architectures:

- Baseline (Fig. 3) - the input is just an embedding of text
- OneHot (Fig. 4) - the user represented as a one-hot vector is concatenated to the text embedding
- HuBi-Formula (Fig. 5) - the deviation of the user's response is taken as its representative feature
- HuBi-Medium (Fig. 6) - annotation-based learned user embedding combined with text embeddings provides the context

B. Implementation details

Experiments were performed using the code provided in the "Anonymous".

Grid search for hyperparameters of Normalizing Flows in experiment 2 was performed with the values specified in Tab. V and Tab. VI.

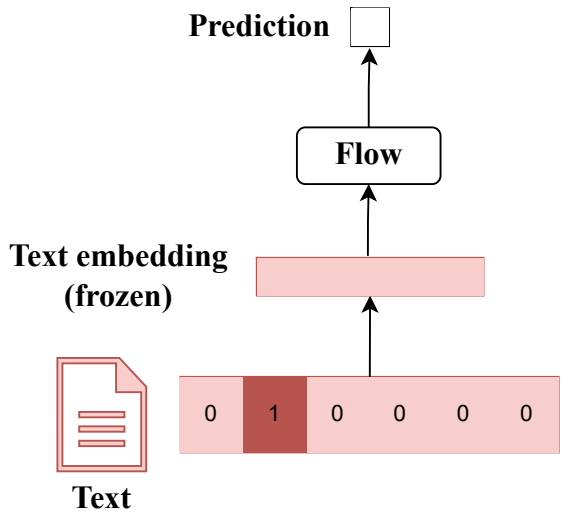


Fig. 3: TXT-Baseline architecture utilizing normalizing flows.

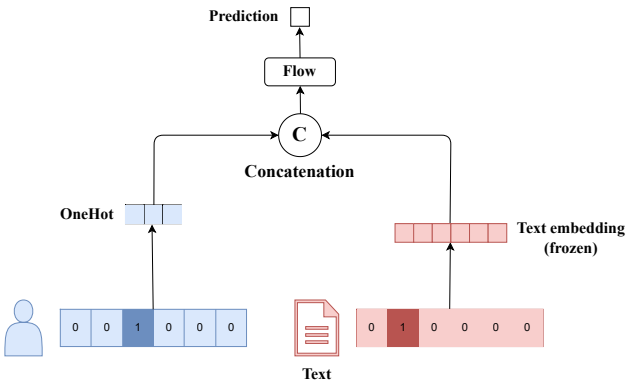


Fig. 4: OneHot architecture utilizing normalizing flows.

For all experiment purposes, we used a machine with AMD Ryzen Threadripper PRO 3955WX 16-Core Processor CPU, 2 x NVIDIA GeForce RTX 3090 GPUs, and 256 GB RAM.

C. Visualization of probabilities

For the combined set of Aggression & Attack, visualizations of the waveform of the probability function were prepared as a result of Experiment 3. described in the publication, are presented in Fig. 7 and Fig. 8.

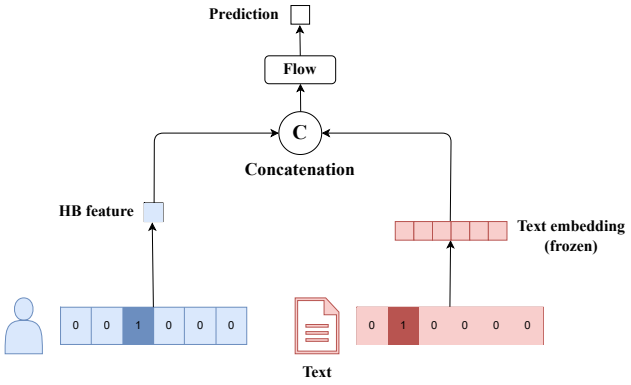


Fig. 5: HuBi-Formula architecture utilizing normalizing flows.

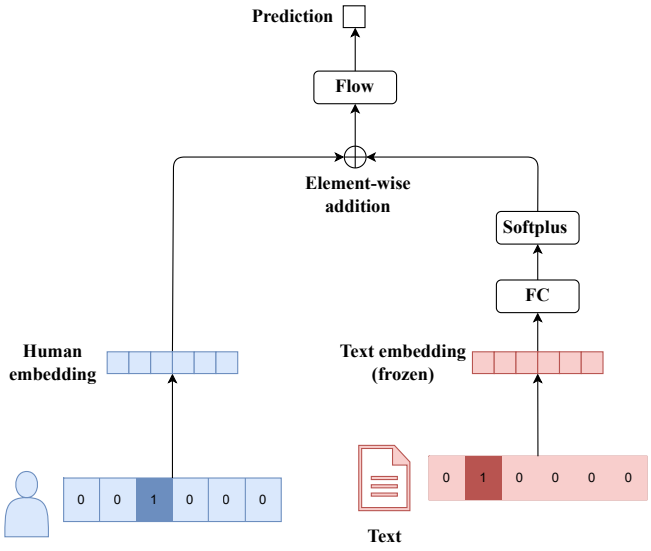


Fig. 6: HuBi-Medium architecture utilizing normalizing flows.

Hyperparameter	Values
hidden features	[2, 4, 6, 8]
num layers	[1, 2, 3, 4, 5]
num. blocks per layer	[1, 2, 3, 4]
dropout probability	[0.0, 0.1, 0.2, 0.4]
batch norm within layers	[True, False]
batch norm between layers	[True, False]

TABLE V: Hyperparameters for Normalizing Flows and their possible values during experiment 2. Note that for MAF, the *dropout probability* hyperparameter was not used at all.

Hyperparameter	Values for TXT-Baseline & OneHot & HuBi-Formula	Values for HuBi-Medium
embedding dim.	50	50
hidden dim.	50	[128, 256, 512, 786]
output dim.	-	[128, 256, 512, 768]
learning rate	[1e-5, 1e-4]	[1e-5, 1e-4]

TABLE VI: Hyperparameters for training and architectures, and their possible values during experiment 2.

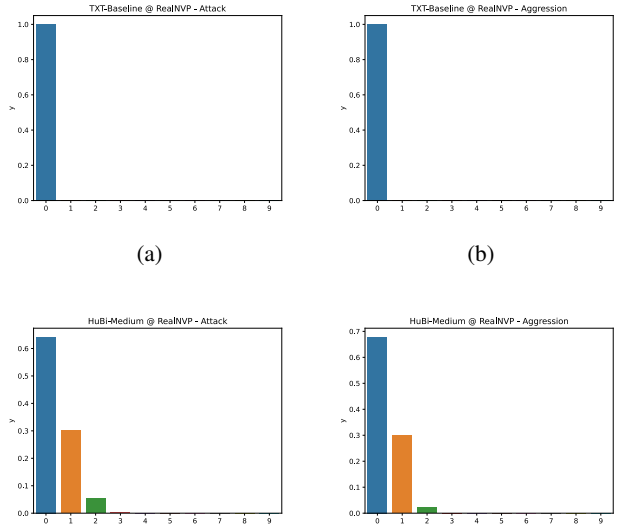


Fig. 7: Visualizations of the waveform of the probability function for RealNVP with different architectures and for Attack and Aggression.

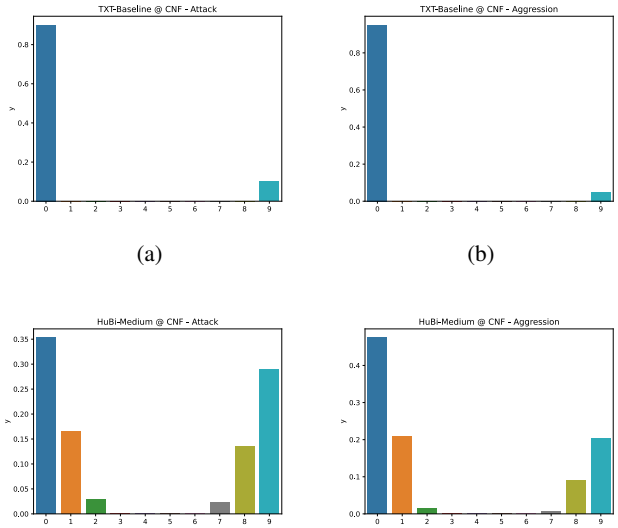


Fig. 8: Visualizations of the waveform of the probability function for CNF with different architectures and for Attack and Aggression.

Flow Plugin Network for Conditional Generation

Patryk Wielopolski[1][0000−0003−2579−8293], Michał Koperski[2][0000−0000−0000−0000], and Maciej Zięba[1,2][0000−0003−4217−7712]

[1] Faculty of Information and Communication Technology, Wroclaw University of Science and Technology, Wybrzeze Wyspianskiego 27, 50-370 Wroclaw, Poland
{patryk.wielopolski,maciej.zieba}@pwr.edu.pl
[2] Tooploox Ltd., ul. Teczowa 7, 53-601 Wrocław, Poland
michal.koperski@tooploox.com

Abstract. Generative models have gained many researchers' attention in the last years resulting in models such as StyleGAN for human face generation or PointFlow for 3D point cloud generation. However, by default, we cannot control its sampling process, i.e., we cannot generate a sample with a specific set of attributes. The current approach is model retraining with additional inputs and different architecture, which requires time and computational resources. We propose a novel approach that enables generating objects with a given set of attributes without retraining the base model. For this purpose, we utilize the normalizing flow models - Conditional Masked Autoregressive Flow, and Conditional Real NVP, as a Flow Plugin Network (FPN).

Keywords: Plugin networks · Normalizing Flows · Conditional Image Generation · Deep generative models

1 Introduction

In the last years, generative models have achieved superior performance in object generation with leading examples such as StyleGAN [8] for human face synthesis and PointFlow [22] for 3D point cloud generation and reconstruction area. These models perform exceptionally well in the case of the unconditional generation process. However, by default, we cannot control its generating process, i.e., we cannot out-of-the-box generate a sample with a specific set of attributes. To perform such a conditional generation, we must put additional effort into creating a new model with such functionality. In the case of images, specific solutions were proposed, e.g., conditional modification of the well-known unconditional generative models - Conditional Variational Autoencoders [20] and Conditional Generative Adversarial Networks [14]. These approaches provide a good result in conditional image generation; however, they require additional effort for training and model design. The ideal solution would consist of a robust transformation of the unconditional base model to a conditional one.

This work proposes a novel approach that enables a generation of objects with a given set of attributes from an already trained unconditional generative

model (a base model) without its retraining. We assume that the base model has an autoencoder architecture, i.e., an encoder and a decoder network that enables the transformation of an object to the latent space representation and inverts that transformation, respectively. In our method, we utilize the concept of the plugin network [10], whose task is to incorporate information known during the inference time to the pretrained model and extend that approach to generative models. As the Flow Plugin Network backbone model, we use the conditional normalizing flow models - Conditional Masked Autoregressive Flow and Conditional Real NVP. Finally, we perform several experiments on three datasets: MNIST, ShapeNet, and CelebA, including conditional object generation, attribute manipulation, and classification tasks. The code for the method and experiments is publicly available.[3]

Concluding, our contributions are as follows:

- we propose a method for conditional object generation from already trained models with an autoencoder architecture, both generative and non-generative;
- we show that the proposed method could be used for non-generative variants of the autoencoder models, thus making them generative;
- we show that it also could be successfully used for classification tasks or as a tool for attribute manipulation.

The rest of the paper is organized as follows. In the next section, we describe the theoretical background. Afterward, we describe in detail our proposed method. In the following section, we shortly describe related works. Then we present the results of the experiments, and in the last section, we summarize our work and propose directions for future research.

2 Background

2.1 Autoencoders

Autoencoder [4] is a neural network primarily designed to learn an informative representation of the data by learning to reconstruct its input. Formally, we want to learn the functions $\mathcal{E}_\phi : \mathbb{R}^n \to \mathbb{R}^q$ and $\mathcal{D}_\theta : \mathbb{R}^q \to \mathbb{R}^n$ where ϕ and θ are parameters of the encoder and decoder. The model is usually trained by minimizing the reconstruction loss: $L_{rec} = \sum_{n=1}^{N} ||\mathbf{x}_n - \mathcal{D}_\theta \circ \mathcal{E}_\phi(\mathbf{x}_n)||_2^2$, assuming training $\mathcal{X}_N = \{\mathbf{x}_n\}$ is given. Some regularization terms can enrich the simple architecture of the autoencoder to obtain the generative model. For Variational Autoencoder (VAE) [9] models, it is achieved by enforcing the latent representation to be normally distributed using Kullback-Leibler divergence. For the adversarial autoencoders, [13], the assumed prior on the embeddings is obtained via adversarial training. With such extensions, the model is not only capable of representing the data on a low-dimensional manifold, but it is also capable of generating samples from an assumed prior in latent space.

[3] https://github.com/pfilo8/Flow-Plugin-Network-for-conditional-generation

2.2 Normalizing Flows

Normalizing flows [18] represent the group of generative models that can be efficiently trained via direct likelihood estimation. They provide a general way of constructing flexible probability distributions over continuous random variables. Suppose we have a D-dimensional real vector $\mathbf{x}$, and we would like to define a joint distribution over $\mathbf{x}$. The main idea of flow-based modeling is to express $\mathbf{x}$ as a transformation T of a real vector $\mathbf{u}$ sampled from a base distribution $p_u(\mathbf{u})$ with a known density function:

$$\mathbf{x} = T(\mathbf{u}) \quad \text{where} \quad \mathbf{u} \sim p_u(\mathbf{u}) \tag{1}$$

It is common to chain multiple transformations $T_1, \ldots, T_K$ to obtain transformation $T = T_K \circ \cdots \circ T_1$, where each T_k transforms $\mathbf{z}_{k-1}$ into $\mathbf{z}_k$, assuming $\mathbf{z}_0 = \mathbf{u}$ and $\mathbf{z}_K = \mathbf{x}$. In practice we implement either T_k or T_k^{-1} using a neural network f_{ϕ_k} with parameters ϕ_k. Assuming given training data $\mathcal{X}_N = \{\mathbf{x}_n\}$ the parameters ϕ_k are estimated in the training procedure by minimizing the negative log-likelihood (NLL):

$$-\log p_{\mathbf{x}}(\mathbf{x}_n) = -\sum_{n=1}^{N}[\log p_{\mathbf{u}}(\mathbf{u}_n) - \sum_{k=1}^{K} \log|\det J_{T_k}(\mathbf{z}_{n,k-1})|], \tag{2}$$

where $p_u(\mathbf{u})$ is usually a Gaussian distribution with independent components. The most challenging aspect of training flow-based models is an optimization of $\log|\det J_{T_k}(\mathbf{z}_{k-1})|$, where the Jacobian $J_T(\mathbf{u})$ is the $D \times D$ matrix of all partial derivatives of T, that may be challenging to compute for high-dimensional data. That issue is usually solved by enforcing the Jacobian matrix to be triangular, for which the determinant is easy to calculate.

3 Flow Plugin Network

3.1 Overview

We consider the base pretrained model $\mathcal{M}$ with an autoencoder architecture, i.e., it has an encoder $\mathcal{E}$, a bottleneck with latent space $\mathcal{Z}$, and decoder $\mathcal{D}$, which was previously trained on dataset $\mathcal{X}$. We also assume we have a set of attributes $\mathcal{Y}$ for samples in the dataset $\mathcal{X}$. The main idea in the proposed method is to use a conditional normalizing flow model $\mathcal{F}$ to learn a bidirectional conditional mapping between a simple base distribution $\mathcal{N}$ (usually standard normal distribution) and a latent space representation which will be conditioned by some attributes (classes) from the set $\mathcal{Y}$. This approach will enable us to generate a random sample from the specific region of the latent space, which should correspond to object representation with a particular set of attributes. In the last step, using a decoder $\mathcal{D}$, we can generate an object with a requested set of attributes. The schema of the proposed method can be found in Figure 1.

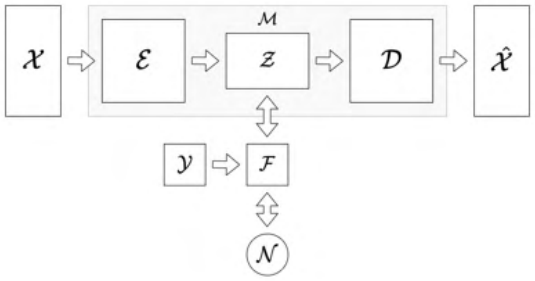


Fig. 1: High-level schema of the Flow Plugin Network method.

3.2 Training

We assume that we have a pretrained model $\mathcal{M}$, samples $\mathbf{x}$ from the dataset $\mathcal{X}$, and attributes $\mathbf{y}$ from the attribute set $\mathcal{Y}$. Our training objective is to learn a bidirectional mapping between a simple base distribution and a latent space representation. For that reason, we need to have a dataset consisting of pairs $(\mathbf{z}_i, \mathbf{y}_i)$ where $\mathbf{z}_i$ is a latent space representation of the vector $\mathbf{x}_i$. In the first step, we obtain such a dataset by encoding vectors $\mathbf{x}_i$ to vectors $\mathbf{z}_i$ using encoder $\mathcal{E}$. In the second step, we train conditional normalizing flow $\mathcal{F}$ using obtained pairs $(\mathbf{z}_i, \mathbf{y}_i)$, by minimizing the conditional negative log-likelihood function:

$$-\sum_{i=1}^{N} \log p(\mathbf{z}_i|\mathbf{y}_i) = -\sum_{i=1}^{N} \left(\log p_u(T^{-1}(\mathbf{z}_i|\mathbf{y}_i)) + \log|\det J_{T^{-1}}(\mathbf{z}_i|\mathbf{y}_i)|\right). \quad (3)$$

It is essential to highlight that weights of the base model $\mathcal{M}$ are frozen during training, and only parameters of transformation T are optimized.

As plugin networks, we use a conditional version of the normalizing flows. For MAF [17], the conditioning component is concatenated to the inputs to mean and log standard deviation functions:

$$\mu_i = f_{\mu_i}(\mathbf{x}_{1:i-1}, \mathbf{y}), \quad \alpha_i = f_{\alpha_i}(\mathbf{x}_{1:i-1}, \mathbf{y}) \quad (4)$$

For RealNVP [6] model, the same operation is applied to scaling and shifting functions for each of the coupling layers:

$$\mathbf{x}_{1:d} = \mathbf{u}_{1:d} \quad (5)$$

$$\mathbf{x}_{d+1:D} = \mathbf{u}_{d+1:D} \odot \exp\left(\mathbf{s}(\mathbf{u}_{1:d}, \mathbf{y})\right) + \mathbf{t}(\mathbf{u}_{1:d}, \mathbf{y}), \quad (6)$$

3.3 Conditional object generation

In this section, we will present how the proposed approach can be applied to conditional image generation. First, we generate a sample from the base distribution $\mathcal{N}$ and construct a vector that encodes the attributes $\mathbf{y}$ of the desired

object that will be generated. Then, the generated sample is passed via the flow conditioned on the vector of attributes embeddings to obtain a vector in the latent space $\mathcal{Z}$ of the base model $\mathcal{M}$. After that, we process that vector to the decoder and obtain the requested object.

3.4 Attribute manipulation

We could also use the proposed approach for object attribute manipulation, e.g., we may want to change the color of the person's hair on the image. First, to perform such an operation, we need to encode an input object to a latent space representation. In the next step, we need to pass the obtained sample through the normalizing flow $\mathcal{F}$ with attributes corresponding to the original image and obtain a sample in the base distribution domain. Subsequently, we need to change the selected attributes. Then, we go back to the latent space representation using the previously obtained sample from the base distribution and the new attributes. Finally, we decode the obtained representation and obtain the final result.

3.5 Classification

The proposed approach can be easily applied as a generative classification model that utilizes the autoencoder's feature representation $\mathcal{Z}$. Assuming that a set $\mathbf{Y}$ is a set of all possible classes, we can use the Bayes rule to calculate predictive distribution $P(\mathbf{y}|\mathbf{z})$, which is a probability of specific class $\mathbf{y}$ given the vector of the latent space representation $\mathbf{z}$: $P(\mathbf{y}|\mathbf{z}) = \frac{P(\mathbf{z}|\mathbf{y})P(\mathbf{y})}{P(\mathbf{z})}$, where $P(\mathbf{y})$ is an assumed class prior and $P(\mathbf{z}) = \sum_{\mathbf{y}\in\mathbf{Y}} P(\mathbf{z}|\mathbf{y})P(\mathbf{y})$. As a consequence, we can omit the denominator as it is a normalizing factor and perform classification according to the rule given by the formula $\hat{\mathbf{y}} = \text{argmax}_{\mathbf{y}\in\mathbf{Y}} P(\mathbf{z}|\mathbf{y})P(\mathbf{y})$.

4 Related works

The problem of conditional generation is solved using a variety of generative approaches. Some of them extend traditional VAE architectures to construct the conditional variants of this kind of model for tasks like visual segmentation [20], conditional image [21], or text generation [23]. Conditional variants of GANs (cGANs) [14] are also popular for class-specific image [16] or point cloud [15] generation. Conditioning mechanisms are also implemented in some variants of flow models, including Masked Autoregressive Flows (MAF) [17], RealNVP [6], and Continuous Normalizing Flows [7]. The conditional flows are successively applied to semi-supervised classification [3], super-resolution generation [12], noise modeling [2], structured sequence prediction [5], or 3D points generation [19]. The presented methods achieve outstanding results in solving a variety of tasks. However, they are trained in an end-to-end fashion, and any changes in the model require very expensive training procedures from scratch.

That issue is tackled by plugin models introduced in [10]. The idea behind plugins is to incorporate additional information, so-called partial evidence, into the trained base model without modifying the parameters. It can be achieved by adding small networks named plugins to the intermediate layers of the network. The goal of these modules is to incorporate additional signals, i.e., information about known labels, into the inference procedure and adjust the predicted output accordingly. Plugin models were successively applied to the classification and segmentation problems. The idea of incorporating an additional, conditional flow model into a pretrained generative model was introduced in StyleFlow model [1]. This approach was designed for StyleGAN [8], the state-of-the-art model for generating face images. MSP [11] utilizes manipulation of VAE latent space via matrix subspace projection to enforce desired features on images.

Compared to the reference approaches, we present a general framework for utilizing conditional flow for various tasks, including conditional image generation, attribute manipulation, and classification. Our model can be applied to any base autoencoder with bottleneck latent space without additional requirements.

5 Experiments

This section evaluates the proposed method for three tasks: conditional object generation, attribute manipulation, and classification. We perform experiments on two domains: images (MNIST, CelebA) and 3D point clouds (ShapeNet). Our goal in the experiments is to qualitatively and quantitatively validate our method.

5.1 Conditional object generation

Our goal in this section is qualitative validation of conditional image and conditional 3D point cloud generation capabilities of Flow Plugin Network (FPN). In this experiment, we utilize the already trained non-conditional autoencoder-like models and plug in our FPN to make the whole architecture conditionally generative.

Methodology The first part of the experiment we performed on the MNIST dataset. Due to the lack of the standard baseline VAE and AE models, we trained these models by ourselves. The model's bottleneck with latent space has a dimensionality equal to 40.

The second part of the experiments we performed on the ShapeNet dataset. It has more classes, which additionally are imbalanced, and it's an application to a different domain. We used PointFlow [22], a de-facto standard model for Point Clouds. The model has an autoencoder-like architecture and operates on a 128-dimensional latent space. We used an already trained non-generative version of the model for all 55 classes. For both datasets, we trained two models with different normalizing flow architectures for each dataset: Conditional Masked Autoregressive Flow (C-MAF) and Conditional Real NVP (C-RNVP).

(a) Autoencoder (b) Flow Plugin Network (FPN)

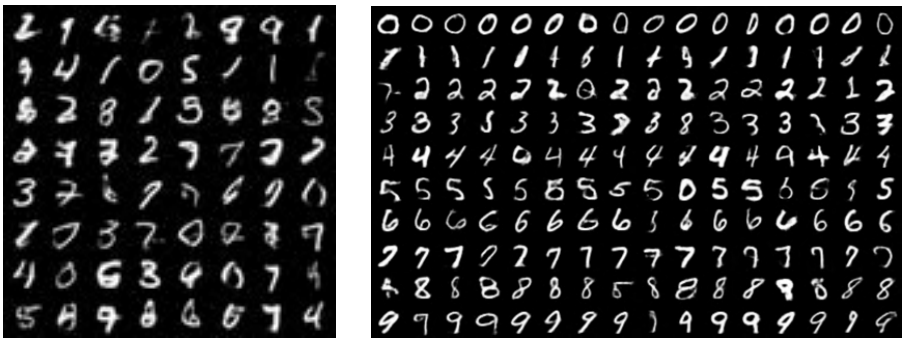
Fig. 2: Comparison of the latent spaces obtained using Autoencoder on MNIST dataset. We can observe that FPN could correctly fit and then sample the latent space distributions for all models. The results were visualized in the 2D space after UMAP transformation, and colors represent different classes.

Results We started experiments by evaluating conditional generative capabilities on latent spaces. The goal of this experiment was to evaluate if the Flow Plugin Network is capable of correctly learning the conditional mapping between base distributions and latent spaces of the models. A successful experiment will inform us that our model correctly generates embeddings, and possible errors on the final object could be caused by the decoder part of the autoencoder.

We generated latent space of the Autoencoder model for the MNIST dataset. It was generated by encoding images using the encoder part of the models. On the other hand, the latent space for Flow Plugin Networks with MAF backbone was generated by sampling 1000 data points conditioned on the specific class. The results are presented in Figure 2. We can observe that our method generates samples in the proper locations of the latent space, which confirms its ability to reconstruct the latent space properly.

In the next experiment, we go one step further, and we sample latent space vectors using the Flow Plugin Network model and decode them by the decoder from the base model. The experiment aims to evaluate the model in an end-to-end conditional generation process.

In this experiment, we utilize the already trained non-conditional autoencoder-like model and plug in our FPN to make the whole architecture conditionally generative. The results for the MNIST dataset are presented in Figure 3. We can observe samples generated from the VAE model on the left - the samples' classes are entirely random. On the right, our method generated samples with specific digit classes. What is essential here is that the weights of the already trained model are not changed during the FPN training phase. Thanks to that, we showed that we could extend the standard autoencoder-like models to the new task - conditional generation, without changes to any of the base model pa-



(a) Samples from Autoencoder. (b) Samples from Flow Plugin Network.

Fig. 3: Motivating example of the paper. Autoencoder correctly generates handwritten digits from the MNIST dataset (on the left), but numbers are completely random. Flow Plugin Network can utilize an already trained model to generate samples of the specific class (on the right).

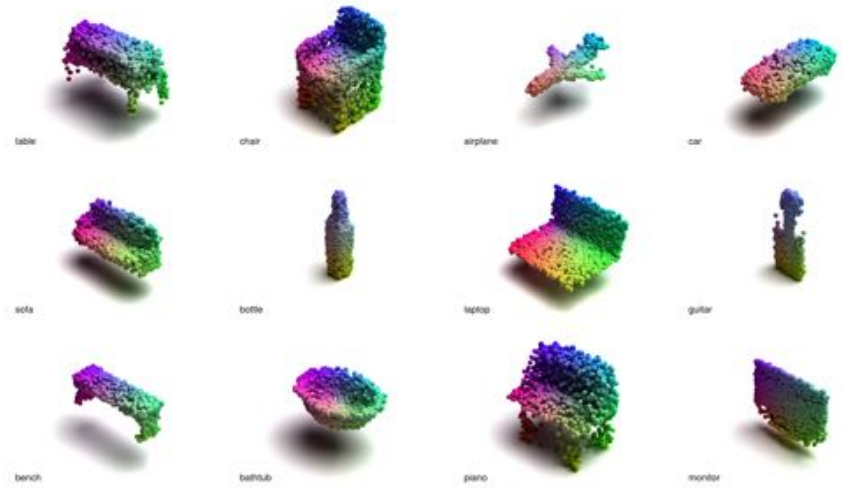


Fig. 4: Examples of 3D Point Clouds generated from PointFlow latent space using Flow Plugin Network model.

rameters. Moreover, we also observed that choosing the normalizing flow model used as a Flow Plugin Network did not impact the qualitative results.

We also performed the same experiment for the ShapeNet dataset using only the C-MAF backbone. The resulting objects are in Figure 4, and the model correctly generated the requested examples. The original PointFlow model was a pure autoencoder without generative capabilities. The authors created sepa-

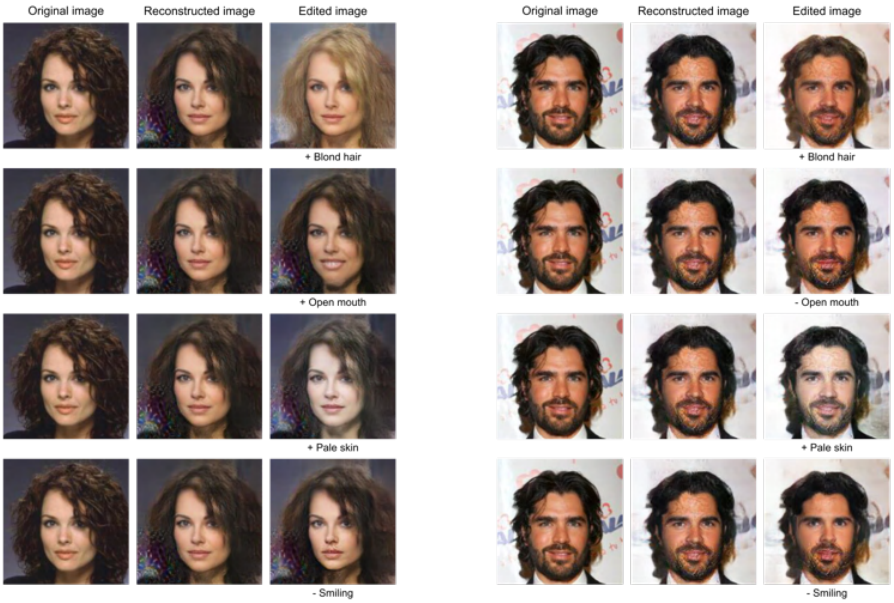


Fig. 5: Results of the feature manipulation experiment on the CelebA dataset. Legend: +/- corresponds to adding or removing a particular feature from the conditioning attributes.

rate models for different classes. Still, we used the general model trained on all classes to generate samples from all classes, effectively turning a non-generative autoencoder model into a generative one.

5.2 Attribute manipulation

Methodology In this experiment, we use the CelebA dataset, Variational Autoencoder with Latent Space Factorization via Matrix Subspace Projection [11] (MSP) as a base model and Conditional Masked Autoregressive Flow as a Flow Plugin Network model.

Results The results are presented in Figure 5. We have the original image on the left, the middle reconstruction from the base model, and the image with changed attributes on the right. We can notice that the base model introduces some artifacts in the reconstruction. The same artifacts are visible in the images generated with our method, as our method uses the base model decoder to generate the image. However, more importantly, the qualitative results show that our proposed method can control the attributes of the reconstructed image. Our model can do it without retraining or modifying the base model.

Table 1: MNIST and ShapeNet Classification experiment results. Our Flow Plugin Network architecture extends unsupervised autoencoders to the classification task. The proposed method achieves a similar level of accuracy as the supervised classification models.

Model	MNIST	ShapeNet
Logistic Regression	0.8833	0.8443
Linear SVM	0.9164	0.8503
RBF SVM	0.9603	**0.8643**
FPN (Conditional-MAF)	**0.9677**	0.8404
FPN (Conditional-RealNVP)	0.9616	0.8370

5.3 Classification

In this section, we show the results of the classification experiment in which we use the proposed Flow Plugin Network model as a classifier as described in Section 3.5. It is a fascinating property of our model, as we can extend either generative or non-generative models (trained in an unsupervised way) to classification models.

Methodology Similar to the Conditional Object Generation part of the experiments, we use MNIST and ShapeNet datasets. Additionally, we use the same base and Flow Plugin Network models described for Conditional Object Generation. This experiment compares our method with other classification baselines trained in a supervised way. Specifically, we compare our method to Logistic Regression, SVM with linear kernel, and SVM with RBF kernel. We performed a hyperparameter grid search for each baseline method using 5-fold cross-validation.

Results The accuracy assessment obtained by all models is presented in Table 1. We can observe that the Flow Plugin Network models were the best-performing ones on the MNIST dataset, and the ShapeNet dataset obtained comparable results to the baseline methods. Moreover, we can observe that both normalizing flows backbones - Conditional MAF and Conditional RealNVP performed comparably.

6 Summary

In this work, we have proposed and successfully tested a method for conditional object generation based on a model with an autoencoder architecture. In the case of non-generative autoencoders, this method makes them generative. Moreover, we used a trained Flow Plugin Network for classification and attribute manipulation tasks.

During experiments, we have shown that using the proposed method, we can conditionally generate images and 3D point clouds from generative models

such as Variational Autoencoder and non-generative ones such as Autoencoder or PointFlow. Moreover, we have performed a classification task on MNIST and ShapeNet datasets and compared the results with shallow machine learning models. We obtained the best results using our model on the former dataset and the latter comparable ones. Lastly, we have successfully manipulated images of human faces.

7 Acknowledgements

The work of M. Zieba was supported by the National Centre of Science (Poland) Grant No. 2020/37/B/ST6/03463.

References

1. Abdal, R., Zhu, P., Mitra, N.J., Wonka, P.: Styleflow: Attribute-conditioned exploration of stylegan-generated images using conditional continuous normalizing flows. ACM Trans. Graph. **40**(3), 21:1–21:21 (2021)
2. Abdelhamed, A., Brubaker, M., Brown, M.S.: Noise flow: Noise modeling with conditional normalizing flows. In: 2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019. pp. 3165–3173. IEEE (2019)
3. Atanov, A., Volokhova, A., Ashukha, A., Sosnovik, I., Vetrov, D.P.: Semi-conditional normalizing flows for semi-supervised learning. CoRR **abs/1905.00505** (2019)
4. Bank, D., Koenigstein, N., Giryes, R.: Autoencoders. CoRR **abs/2003.05991** (2020)
5. Bhattacharyya, A., Hanselmann, M., Fritz, M., Schiele, B., Straehle, C.: Conditional flow variational autoencoders for structured sequence prediction. CoRR **abs/1908.09008** (2019)
6. Dinh, L., Sohl-Dickstein, J., Bengio, S.: Density estimation using real NVP. In: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. OpenReview.net (2017)
7. Grathwohl, W., Chen, R.T.Q., Bettencourt, J., Sutskever, I., Duvenaud, D.: FFJORD: free-form continuous dynamics for scalable reversible generative models. In: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019. OpenReview.net (2019)
8. Karras, T., Laine, S., Aila, T.: A style-based generator architecture for generative adversarial networks. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019. pp. 4401–4410. Computer Vision Foundation / IEEE (2019)
9. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. In: 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings (2014)
10. Koperski, M., Konopczynski, T.K., Nowak, R., Semberecki, P., Trzcinski, T.: Plugin networks for inference under partial evidence. In: IEEE Winter Conference on Applications of Computer Vision, WACV 2020, Snowmass Village, CO, USA, March 1-5, 2020. pp. 2872–2880. IEEE (2020)

11. Li, X., Lin, C., Li, R., Wang, C., Guerin, F.: Latent space factorisation and manipulation via matrix subspace projection. In: Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event. Proceedings of Machine Learning Research, vol. 119, pp. 5916–5926. PMLR (2020)
12. Lugmayr, A., Danelljan, M., Gool, L.V., Timofte, R.: Srflow: Learning the super-resolution space with normalizing flow. In: Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part V. Lecture Notes in Computer Science, vol. 12350, pp. 715–732. Springer (2020)
13. Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I.J.: Adversarial autoencoders. CoRR **abs/1511.05644** (2015)
14. Mateos, M., González, A., Sevillano, X.: Guiding gans: How to control non-conditional pre-trained gans for conditional image generation. CoRR **abs/2101.00990** (2021)
15. Milz, S., Simon, M., Fischer, K., Pöpperl, M., Gross, H.: Points2pix: 3d point-cloud to image translation using conditional gans. In: Pattern Recognition - 41st DAGM German Conference, DAGM GCPR 2019, Dortmund, Germany, September 10-13, 2019, Proceedings. Lecture Notes in Computer Science, vol. 11824, pp. 387–400. Springer (2019)
16. Odena, A., Olah, C., Shlens, J.: Conditional image synthesis with auxiliary classifier gans. In: Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017. Proceedings of Machine Learning Research, vol. 70, pp. 2642–2651. PMLR (2017)
17. Papamakarios, G., Murray, I., Pavlakou, T.: Masked autoregressive flow for density estimation. In: Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA. pp. 2338–2347 (2017)
18. Papamakarios, G., Nalisnick, E.T., Rezende, D.J., Mohamed, S., Lakshminarayanan, B.: Normalizing flows for probabilistic modeling and inference. J. Mach. Learn. Res. **22**, 57:1–57:64 (2021)
19. Pumarola, A., Popov, S., Moreno-Noguer, F., Ferrari, V.: C-flow: Conditional generative flow models for images and 3d point clouds. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020. pp. 7946–7955. Computer Vision Foundation / IEEE (2020)
20. Sohn, K., Lee, H., Yan, X.: Learning structured output representation using deep conditional generative models. In: Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada. pp. 3483–3491 (2015)
21. Yan, X., Yang, J., Sohn, K., Lee, H.: Attribute2image: Conditional image generation from visual attributes. In: Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part IV. Lecture Notes in Computer Science, vol. 9908, pp. 776–791. Springer (2016)
22. Yang, G., Huang, X., Hao, Z., Liu, M., Belongie, S.J., Hariharan, B.: Pointflow: 3d point cloud generation with continuous normalizing flows. In: 2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019. pp. 4540–4549. IEEE (2019)
23. Zhao, T., Zhao, R., Eskénazi, M.: Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In: Barzilay, R., Kan, M. (eds.) Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers. pp. 654–664. Association for Computational Linguistics (2017)

The Thirty-Sixth AAAI Conference on Artificial Intelligence (AAAI-22)

PluGeN: Multi-Label Conditional Generation from Pre-trained Models

Maciej Wołczyk[1*], **Magdalena Proszewska**[1*], **Łukasz Maziarka**[1], **Maciej Zieba**[2,4], **Patryk Wielopolski**[2], **Rafał Kurczab**[3], **Marek Śmieja**[1†]

[1]Jagiellonian University
[2]Wroclaw University of Science and Technology
[3] Institute of Pharmacology PAS
[4] Tooploox

Abstract

Modern generative models achieve excellent quality in a variety of tasks including image or text generation and chemical molecule modeling. However, existing methods often lack the essential ability to generate examples with requested properties, such as the age of the person in the photo or the weight of the generated molecule. Incorporating such additional conditioning factors would require rebuilding the entire architecture and optimizing the parameters from scratch. Moreover, it is difficult to disentangle selected attributes so that to perform edits of only one attribute while leaving the others unchanged. To overcome these limitations we propose PluGeN (Plugin Generative Network), a simple yet effective generative technique that can be used as a plugin to pre-trained generative models. The idea behind our approach is to transform the entangled latent representation using a flow-based module into a multi-dimensional space where the values of each attribute are modeled as an independent one-dimensional distribution. In consequence, PluGeN can generate new samples with desired attributes as well as manipulate labeled attributes of existing examples. Due to the disentangling of the latent representation, we are even able to generate samples with rare or unseen combinations of attributes in the dataset, such as a young person with gray hair, men with make-up, or women with beards. We combined PluGeN with GAN and VAE models and applied it to conditional generation and manipulation of images and chemical molecule modeling. Experiments demonstrate that PluGeN preserves the quality of backbone models while adding the ability to control the values of labeled attributes. Implementation is available at https://github.com/gmum/plugen.

Introduction

Generative models such as GANs and variational autoencoders have achieved great results in recent years, especially in the domains of images (Brock, Donahue, and Simonyan 2018; Brown et al. 2020) and cheminformatics (Gómez-Bombarelli et al. 2018; Jin, Barzilay, and Jaakkola 2018). However, in many practical applications, we need to control the process of creating samples by enforcing particular features of generated objects. This would be required to regulate the biases present in the data, e.g. to assure that people of each ethnicity are properly represented in the generated set of face images. In numerous realistic problems, such as drug discovery, we want to find objects with desired properties, like molecules with a particular activity, non-toxicity, and solubility.

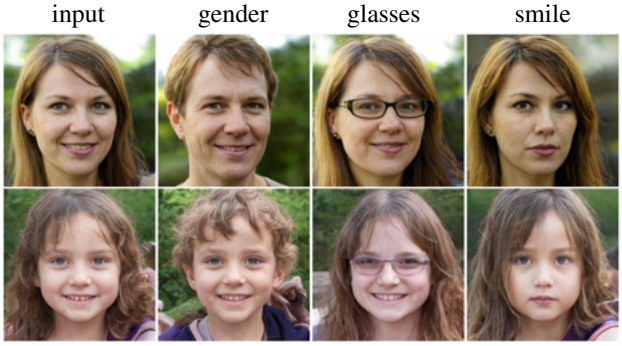
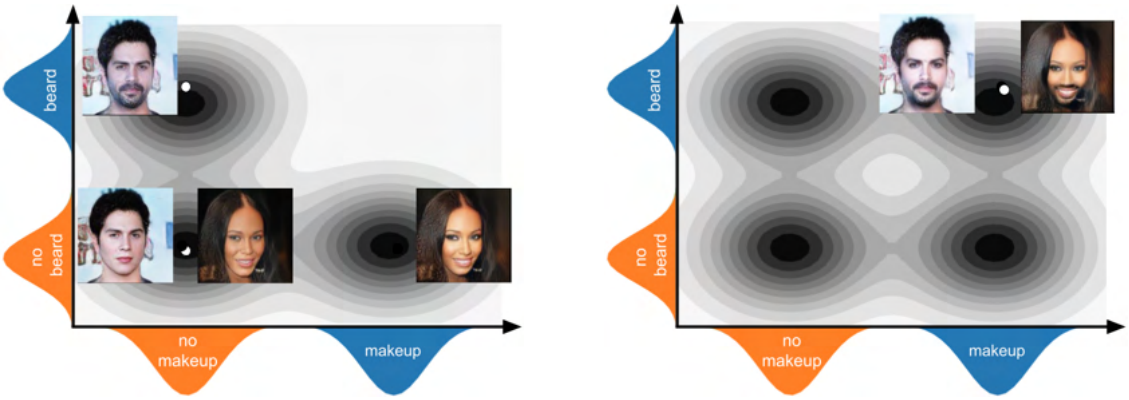


Figure 1: Attributes manipulation performed by PluGeN using the StyleGAN backbone.

Designing the conditional variants of generative models that operate on multiple labels is a challenging problem due to intricate relations among the attributes. Practically, it means that some combinations of attributes (e.g. a woman with a beard) might be unobserved or rarely observed in the training data. In essence, the model should be able to go beyond the distribution of seen data and generate examples with combinations of attributes not encountered previously. One might approach this problem by building a new conditional generative model from the ground up or design a solution tailored for a specific existing unsupervised generative model. However, this introduces an additional effort when one wants to adapt it to a newly invented approach.

To tackle this problem while leveraging the power of existing techniques, we propose PluGeN (Plugin Generative Network), a simple yet effective generative technique that can be used as a plugin to various pre-trained generative models such as VAEs or GANs, see Figure 1 for demonstration. Making use of PluGeN, we can manipulate the attributes of input examples as well as generate new samples with desired features. When training the proposed module, we do not change the parameters of the base model and thus

*Equal contribution
†Corresponding author: marek.smieja@uj.edu.pl



(a) Factorization of true data distribution

(b) Probability distribution covered by PluGeN.

Figure 2: PluGeN factorizes true data distribution into components (marginal distributions) related to labeled attributes, see (a), and allows for describing unexplored regions of data (uncommon combinations of labels) by sampling from independent components, see (b). In the case illustrated here, PluGeN constructs pictures of men with make-up or women with beards, although such examples rarely (or never) appear in the training set.

we retain its generative and reconstructive abilities, which places our work in the emerging family of non-invasive network adaptation methods (Wołczyk et al. 2021; Rebuffi, Bilen, and Vedaldi 2017; Koperski et al. 2020; Kaya, Hong, and Dumitras 2019; Zhou et al. 2020).

Our idea is to find a mapping between the entangled latent representation of the backbone model and a disentangled space, where each dimension corresponds to a single, interpretable attribute of the image. By factorizing the true data distribution into independent components, we can sample from each component independently, which results in creating samples with arbitrary combinations of attributes, see Figure 2. In contrast to many previous works, which are constrained to the attributes combinations visible in the training set, PluGeN gives us full control of the generation process, being able to create uncommon combinations of attributes, such as a woman with a beard or a man with heavy make-up. Generating samples with unseen combinations of attributes can be viewed as extending the distribution of generative models to unexplored although reasonable regions of data space, which distinguishes our approach from existing solutions.

Extensive experiments performed on the domain of images and a dataset of chemical compounds demonstrate that PluGeN is a reusable plugin that can be applied to various architectures including GANs and VAEs. In contrast to the baselines, PluGeN can generate new samples as well as manipulate the properties of existing examples, being capable of creating uncommon combinations of attributes.

Our contributions are as follow:

- We propose a universal and reusable plugin for multi-label generation and manipulation that can be attached to various generative models and applied it to diverse domains, such as chemical molecule modeling.
- We introduce a novel way of modeling conditional distributions using invertible normalizing flows based on the latent space factorization.
- We experimentally demonstrate that PluGeN can produce samples with uncommon combinations of attributes going beyond the distribution of training data.

Related work

Conditional VAE (cVAE) is one of the first methods which includes additional information about the labeled attributes in a generative model (Kingma et al. 2014). Although this approach has been widely used in various areas ranging from image generation (Sohn, Lee, and Yan 2015; Yan et al. 2016; Klys, Snell, and Zemel 2018) to molecular design (Kang and Cho 2018), the independence of the latent vector from the attribute data is not assured, which negatively influences the generation quality. Conditional GAN (cGAN) is an alternative approach that gives results of significantly better quality (Mirza and Osindero 2014; Perarnau et al. 2016; He et al. 2019), but the model is more difficult to train (Kodali et al. 2017). cGAN works very well for generating new images and conditioning factors may take various forms (images, sketches, labels) (Park et al. 2019; Jo and Park 2019; Choi et al. 2020), but manipulating existing examples is more problematic because GAN models lack the encoder network (Tov et al. 2021). Fader Networks (Lample et al. 2017) combine features of both cVAE and cGAN, as they use encoder-decoder architecture, together with the discriminator, which predicts the image attributes from its latent vector returned from the encoder. As discussed in (Li et al. 2020), the training of Fader Networks is even more difficult than standard GANs, and disentanglement of attributes is not preserved. MSP (Li et al. 2020) is a recent auto-encoder based architecture with an additional projection matrix, which is responsible for disentangling the latent space and separating the attribute information from other characteristic information. In contrast to PluGeN, MSP cannot be used with pre-trained GANs and performs poorly at generating new images (it was designed for manipulating existing examples). CAGlow (Liu et al. 2019) is an adaptation of Glow (Kingma and Dhariwal

2018) to conditional image generation based on modeling a joint probabilistic density of an image and its conditions. Since CAGlow does not reduce data dimension, applying it to more complex data might be problematic.

While the above approaches focus on building conditional generative models from scratch, recent works often focus on manipulating the latent codes of pre-trained models. StyleFlow (Abdal et al. 2021) operates on the latent space of StyleGAN (Karras, Laine, and Aila 2019) using a conditional continuous flow module. Although the quality of generated images is impressive, the model has not been applied to other generative models than StyleGAN and domains other than images. Moreover, StyleFlow needs an additional classifier to compute the conditioning factor (labels) for images at test time. Competitive approaches to StyleGAN appear in (Gao et al. 2021; Tewari et al. 2020; Härkönen et al. 2020; Nitzan et al. 2020). InterFaceGAN (Shen et al. 2020) postulates that various properties of the facial semantics can be manipulated via linear models applied to the latent space of GANs. Hijack-GAN (Wang, Yu, and Fritz 2021) goes beyond linear models and designs a proxy model to traverse the latent space of GANs.

In disentanglement learning, we assume that the data has been generated from a fixed number of independent factors of underlying variation. The goal is then to find a transformation that unravels these factors so that a change in one dimension of the latent space corresponds to a change in one factor of variation while being relatively invariant to changes in other factors (Bengio, Courville, and Vincent 2013; Kim and Mnih 2018; Higgins et al. 2017; Brakel and Bengio 2017; Kumar, Sattigeri, and Balakrishnan 2017; Chen et al. 2019; Spurek et al. 2020; Dinh, Krueger, and Bengio 2014; Sorrenson, Rother, and Köthe 2020; Chen et al. 2016). As theoretically shown in (Locatello et al. 2019), the unsupervised learning of disentangled representations is fundamentally impossible without inductive biases on both the models and the data. In this paper, we solve a slightly different problem than typical disentanglement, as we aim to deliver an efficient plug-in model to a large variety of existing models in order to manipulate attributes without training the entire system. Creating compact add-ons for large models saves training time and energy consumption.

Plugin Generative Network

We propose a plugin generative network (PluGeN), which can be attached to pre-trained generative models and allows for direct manipulation of labeled attributes, see Figure 3 for the basic scheme of PluGeN. Making use of PluGeN we preserve all properties of the base model, such as generation quality and reconstruction in the case of auto-encoders, while adding new functionalities. In particular, we can:

- modify selected attributes of existing examples,
- generate new samples with desired labels.

In contrast to typical conditional generative models, PluGeN is capable of creating examples with rare or even unseen combinations of attributes, e.g. man with makeup.

Probabilistic model. PluGeN works in a multi-label setting, where every example $\mathbf{x} \in \mathcal{X}$ is associated with a K-dimensional vector of binary labels[1] $\mathbf{y} = (y_1, \ldots, y_K) \in \{0,1\}^K$. We assume that there is a pre-trained generative model $\mathcal{G} : \mathcal{Z} \to \mathbb{R}^D$, where $\mathcal{Z} \subset \mathbb{R}^N$ is the latent space, which is usually heavily entangled. That is, although each latent code $\mathbf{z} \in \mathcal{Z}$ contains the information about the labels $\mathbf{y}$, there is no direct way to extract or modify it.

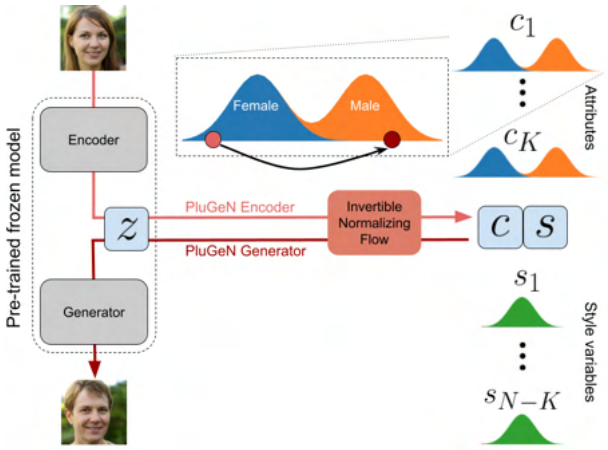


Figure 3: PluGeN maps the entangled latent space $\mathcal{Z}$ of pre-trained generative models using invertible normalizing flow into a separate space, where labeled attributes are modeled using independent 1-dimensional distributions. By manipulating label variables in this space, we fully control the generation process.

We want to map this entangled latent space $\mathcal{Z}$ into a separate latent space $\mathcal{D} \subset \mathbb{R}^N$ which encodes the values of each label y_k as a separate random variable C_k living in a single dimension of this space. Thus, by changing the value of C_k, going back to the entangled space $\mathcal{Z}$ and generating a sample, we can control the values of y_k. Since labeled attributes usually do not fully describe a given example, we consider additional $N-K$ random variables S_k, which are supposed to encode the information not included in the labels. We call $\mathbf{C} = (C_1, \ldots, C_K)$ the label variables (or attributes) and $\mathbf{S} = (S_1, \ldots, S_{N-K})$ the style variables.

Since we want to control the value of each attribute independently of any other factors, we assume the factorized form of the probability distribution of the random vector $(\mathbf{C}, \mathbf{S})$. More precisely, the conditional probability distribution of $(\mathbf{C}, \mathbf{S})$ given any condition $\mathbf{Y} = \mathbf{y}$ imposed on labeled attributes is of the form:

$$p_{\mathbf{C},\mathbf{S}|\mathbf{Y}=\mathbf{y}}(\mathbf{c},\mathbf{s}) = \prod_{i=1}^{K} p_{C_i|Y_i=y_i}(c_i) \cdot p_{\mathbf{S}}(\mathbf{s}), \qquad (1)$$

for all $(\mathbf{c}, \mathbf{s}) = (c_1, \ldots, c_K, s_1, \ldots, s_{N-K}) \in \mathbb{R}^N$. In other words, modifying $Y_i = y_i$ influences only the i-th factor C_i leaving other features unchanged.

Parametrization. To instantiate the above probabilistic model (1), we need to parametrize the conditional distribu-

[1]Our model can be extended to continuous values, which we describe in the supplementary materials due to page limit.

tion of C_i given $Y_i = y_i$ and the distribution of $\mathbf{S}$. Since we do not impose any constraints on style variables, we use standard Gaussian distribution for modeling density of $\mathbf{S}$:

$$p_{\mathbf{S}} = \mathcal{N}(0, I_{N-K}).$$

To provide the consistency with $p_{\mathbf{S}}$ and avoid potential problems with training our deep learning model using discrete distributions, we use the mixture of two Gaussians for modeling the presence of labels – each component corresponds to a potential value of the label (0 or 1). More precisely, the conditional distribution of C_i given $Y_i = y_i$ is parametrized by:

$$p_{C_i|Y_i=y_i} = \mathcal{N}(m_0, \sigma_0)^{(1-y_i)} \cdot \mathcal{N}(m_1, \sigma_1)^{y_i}, \quad (2)$$

where $m_0, m_1, \sigma_0, \sigma_1$ are the user-defined parameters. If $y_i = 0$, then the latent factor C_i takes values close to m_0; otherwise we get values around m_1 (depending on the value of σ_0 and σ_1). To provide good separation between components, we put $m_0 = -1, m_1 = 1$; the selection of σ_0, σ_1 will be discussed is the supplementary materials.

Thanks to this continuous parametrization, we can smoothly interpolate between different labels, which would not be so easy using e.g. Gumbel softmax parametrization (Jang, Gu, and Poole 2016). In consequence, we can gradually change the intensity of certain labels, like smile or beard, even though such information was not available in a training set (see Figure 4 in the experimental section).

Training the model To establish a two-way mapping between entangled space $\mathcal{Z}$ and the disentangled space $\mathcal{D}$, we use an invertible normalizing flow (INF), $\mathcal{F} : \mathbb{R}^N \to \mathcal{Z}$. Let us recall that INF is a neural network, where the inverse mapping is given explicitly and the Jacobian determinant can be easily calculated (Dinh, Krueger, and Bengio 2014). Due to the invertibility of INF, we can transform latent codes $\mathbf{z} \in \mathcal{Z}$ to the prior distribution of INF, modify selected attributes, and map the resulting vector back to $\mathcal{Z}$. Moreover, INFs can be trained using log-likelihood loss, which is very appealing in generative modeling.

Summarizing, given a latent representation $\mathbf{z} \in \mathcal{Z}$ of a sample $\mathbf{x}$ with label $\mathbf{y}$, the loss function of PluGeN equals:

$$\begin{aligned}
&-\log p_{\mathbf{Z}|\mathbf{Y}=\mathbf{y}}(\mathbf{z}) = \\
&\quad -\log\left(p_{\mathbf{C},\mathbf{S}|\mathbf{Y}=\mathbf{y}}(\mathbf{c},\mathbf{s}) \cdot \left|\det \frac{\partial \mathcal{F}^{-1}(\mathbf{z})}{\partial \mathbf{z}}\right|\right) = \\
&-\log\left(\prod_{i=1}^{K} p_{C_i|Y_i=y_i}(c_i) \cdot p_{\mathbf{S}}(\mathbf{s})\right) - \log\left|\det \frac{\partial \mathcal{F}^{-1}(\mathbf{z})}{\partial \mathbf{z}}\right| = \\
&-\sum_{i=1}^{K} \log p_{C_i|Y_i=y_i}(c_i) - \log p_{\mathbf{S}}(\mathbf{s}) - \log\left|\det \frac{\partial \mathcal{F}^{-1}(\mathbf{z})}{\partial \mathbf{z}}\right|,
\end{aligned} \quad (3)$$

where $(\mathbf{c}, \mathbf{s}) = \mathcal{F}^{-1}(\mathbf{z})$. In the training phase, we collect latent representations $\mathbf{z}$ of data points $\mathbf{x}$. Making use of labeled attributes $\mathbf{y}$ associated with every $\mathbf{x}$, we modify the weights of $\mathcal{F}$ so that to minimize the negative log-likelihood (3) using gradient descent. The weights of the base model $\mathcal{G}$ are kept frozen.

In contrast to many previous works (Abdal et al. 2021), PluGeN can be trained in a semi-supervised setting, where only partial information about labeled attributes is available (see supplementary materials for details).

Inference. We may use PluGeN to generate new samples with desired attributes as well as to manipulate attributes of input examples. In the first case, we generate a vector $(\mathbf{c}, \mathbf{s})$ from the conditional distribution $p_{\mathbf{C},\mathbf{S}|\mathbf{Y}=\mathbf{y}}$ with selected condition $\mathbf{y}$. To get the output sample, the vector $(\mathbf{c}, \mathbf{s})$ is transformed by the INF and the base generative network $\mathcal{G}$, which gives us the final output $\mathbf{x} = \mathcal{G}(\mathcal{F}(\mathbf{c}, \mathbf{s}))$.

In the second case, to manipulate the attributes of an existing example $\mathbf{x}$, we need to find its latent representation $\mathbf{z}$. If $\mathcal{G}$ is a decoder network of an autoencoder model, then $\mathbf{x}$ should be passed through the encoder network to obtain $\mathbf{z}$ (Li et al. 2020). If $\mathcal{G}$ is a GAN, then $\mathbf{z}$ can be found by minimizing the reconstruction error between $\mathbf{x}' = \mathcal{G}(\mathbf{z})$ and $\mathbf{x}$ using gradient descent for a frozen $\mathcal{G}$ (Abdal et al. 2021). In both cases, $\mathbf{z}$ is next processed by INF, which gives us its factorized representation $(\mathbf{c}, \mathbf{s}) = \mathcal{F}^{-1}(\mathbf{z})$. In this representation, we can modify any labeled variable c_i and map the resulting vector back through $\mathcal{F}$ and $\mathcal{G}$ as in the generative case.

Observe that PluGeN does not need to know what are the values of labeled attributes when it modifies attributes of existing examples. Given a latent representation $\mathbf{z}$, PluGeN maps it through $\mathcal{G}^{-1}$, which gives us the factorization into labeled and unlabeled attributes. In contrast, existing solutions based on conditional INF, e.g StyleFlow (Abdal et al. 2021), have to determine all labels before passing $\mathbf{z}$ through INF as they represent the conditioning factors. In consequence, these models involve additional classifiers for labeled attributes.

Experiments

To empirically evaluate the properties of PluGeN, we combine it with GAN and VAE architectures to manipulate attributes of image data. Moreover, we present a practical use-case of chemical molecule modeling using CharVAE. Due to the page limit, we included architecture details and additional results in the supplementary materials.

GAN backbone First, we consider the state-of-the-art StyleGAN architecture (Karras, Laine, and Aila 2019), which was trained on Flickr-Faces-HQ (FFHQ) containing 70 000 high-quality images of resolution 1024×1024. The Microsoft Face API was used to label 8 attributes in each image (gender, pitch, yaw, eyeglasses, age, facial hair, expression, and baldness).

PluGeN is instantiated using NICE flow model (Dinh, Krueger, and Bengio 2014) that operates on the latent vectors $\mathbf{w} \in \mathbb{R}^{512}$ sampled from the $\mathbf{W}$ space of the StyleGAN. As a baseline, we select StyleFlow (Abdal et al. 2021), which is currently one of the state-of-the-art models for controlling the generation process of StyleGAN. In contrast to PluGeN, StyleFlow uses the conditional continuous INF to operate on the latent codes of StyleGAN, where the conditioning factor corresponds to the labeled attributes. For eval-

uation, we modify one of 5 attributes[2] and verify the success of this operation using the prediction accuracy returned by Microsoft Face API. The quality of images is additionally assessed by calculating the standard Fréchet Inception Distance (FID) (Heusel et al. 2017).

Figure 1 (first page) and 4 present the effects of how PluGeN and StyleFlow manipulate images sampled by StyleGAN. It is evident that PluGeN can switch the labels to opposite values as well as gradually change their intensities. At the same time, the requested modifications do not influence the remaining attributes leaving them unchanged. One can observe that the results produced by StyleFlow are also acceptable, but the modification of the requested attribute implies the change of other attributes. For example, increasing the intensity of "baldness" changes the type of glasses, or turning the head into right makes the woman look straight.

The above qualitative evaluation is supported by the quantitative assessment presented in Table 1. As can be seen, StyleFlow obtains a better FID score, while PluGeN outperforms StyleFlow in terms of accuracy. Since FID compares the distribution of generated and real images, creating images with uncommon combinations of attributes that do not appear in a training set may be scored lower, which can explain the relation between accuracy and FID obtained by PluGeN and StyleFlow. In consequence, FID is not an adequate metric for measuring the quality of arbitrary image manipulations considered here, because it is too closely tied to the distribution of input images.

It is worth mentioning that PluGeN obtains these very good results using NICE model, which is the simplest type of INFs. In contrast, StyleFlow uses continuous INF, which is significantly more complex and requires using an ODE solver leading to unstable training. Moreover, to modify even a single attribute, StyleFlow needs to determine the values of all labels, since they represent the conditioning factors of INF. In consequence, every modification requires applying an auxiliary classifier to predict all image labels. The usage of PluGeN is significantly simpler, as subsequent coordinates in the latent space of INF correspond to the labeled attributes and they are automatically determined by PluGeN. Finally, our approach is less computationally expensive as we verified that, using the same hardware, PluGeN can be trained 3 times faster than StyleFlow and is around 100 times faster in inference.

Image manipulation on VAE backbone In the following experiment, we show that PluGeN can be combined with autoencoder models to effectively manipulate image attributes. We use CelebA database, where every image of the size 256×256 is annotated with 40 binary labels.

We compare PluGeN to MSP (Li et al. 2020), a strong baseline, which uses a specific loss for disentangling the latent space of VAE. Following the idea of StyleFlow, we also consider a conditional INF attached to the latent space of pre-trained VAE (referred to as cFlow), where conditioning factors correspond to the labeled attributes. The architecture of the base VAE and the evaluation protocol were taken from the original MSP paper. More precisely, for every input image, we manipulate the values of two attributes (we inspect 20 combinations in total). The success of the requested manipulation is verified using a multi-label ResNet-56 classifier trained on the original CelebA dataset.

Requested value	PluGeN	StyleFlow
female	**0.95**	**0.95**
male	**0.92**	0.87
no-glasses	**1.00**	0.99
glasses	**0.90**	0.70
not-bald	**1.00**	**1.00**
bald	0.53	**0.54**
no-facial-hair	**1.00**	**1.00**
facial-hair	**0.72**	0.65
no-smile	**0.99**	0.92
smile	0.96	**0.99**
Average Acc	**0.90**	0.86
Average FID	46.51	**32.59**

Table 1: Accuracy and FID scores of attributes modification using StyleGAN backbone.

The sample results presented in Figure 5 demonstrate that PluGeN attached to VAE produces high-quality images satisfying the constraints imposed on the labeled attributes. The quantitative comparison shown in Table 2 confirms that PluGeN is extremely efficient in creating uncommon combinations of attributes, while cFlow performs well only for the usual combinations. At the same time, the quality of images produced by PluGeN and MSP is better than in the case of cFlow. Although both PluGeN and MSP focus on disentangling the latent space of the base model, MSP has to be trained jointly with the base VAE model and it was designed only to autoencoder models. In contrast, PluGeN is a separate module, which can be attached to arbitrary pre-trained models. Due to the use of invertible neural networks, it preserves the reconstruction quality of the base model, while adding manipulation functionalities. In the following experiment, we show that PluGeN also performs well at generating entirely new images, which is not possible using MSP.

Image generation with VAE backbone In addition to manipulating the labeled attributes of existing images, PluGeN generates new examples with desired attributes. To verify this property, we use the same VAE architecture as before trained on CelebA dataset. The baselines include cFlow and two previously introduced methods for multi-label conditional generation[3]: cVAE (Yan et al. 2016) and Δ-GAN (Gan et al. 2017). We exclude MSP from the comparison because it cannot generate new images, but only manipulate the attributes of existing ones (see supplementary materials for a detailed explanation).

Figure 6 presents sample results of image generation with the specific conditions. In each row, we fix the style variables $\mathbf{s}$ and vary the label variables $\mathbf{c}$ in each column, generating

[2]The remaining 3 attributes (age, pitch, yaw) are continuous and it is more difficult to assess their modifications.

[3]For cVAE and Δ-GAN we use images of the size 64×64 following their implementations.

(a) PluGeN (b) StyleFlow

Figure 4: Gradual modification of attributes (age, baldness, and yaw, respectively) performed on the StyleGAN latent codes.

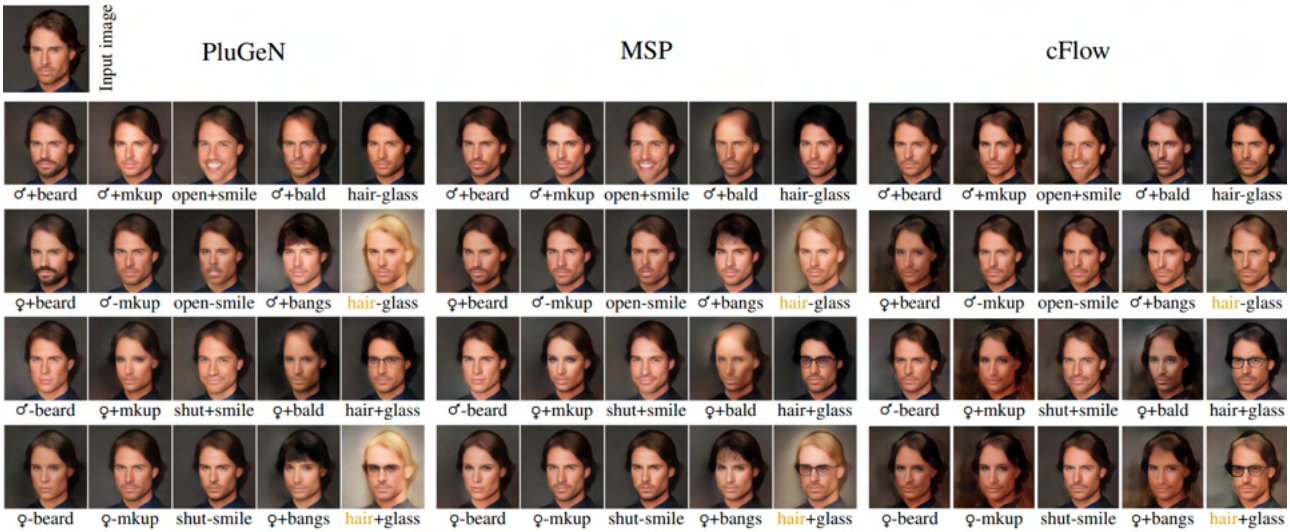


Figure 5: Examples of image attribute manipulation using VAE backbone.

the same person but with different characteristics such as hair color, eyeglasses, etc. Although cVAE manages to modify the attributes, the quality of obtained samples is poor, while Δ-GAN falls completely out of distribution. PluGeN and cFlow generate images of similar quality, but only PluGeN is able to correctly manipulate the labeled attributes. The lower quality of generated images is caused by the poor generation abilities of VAE backbone, which does not work well with high dimensional images (see supplementary materials for a discussion). For this reason, it is especially notable that PluGeN can improve the generation performance of the backbone model in contrast to MSP.

Disentangling the attributes The attributes in the CelebA dataset are strongly correlated and at times even contradictory, e.g. attributes 'bald' and 'blond hair' cannot both be present at the same time. In this challenging task, we aim to disentangle the attribute space as much as it is possible to allow for generating examples with arbitrary combinations of attributes. For this purpose, we sample the conditional variables c_i independently, effectively ignoring the underlying correlations of attributes, and use them to generate images. Since the attributes in the CelebA dataset are often imbalanced (e.g. only in 6.5% of examples the person wears glasses), we calculate F1 and AUC scores for each attribute.

The quantitative analysis of the generated images presented in Table 3 confirms that PluGeN outperforms the rest of the methods with respect to classification scores. The overall metrics are quite low for all approaches, which is due to the difficulty of disentanglement mentioned above, as well as the inaccuracy of the ResNet attribute classifier. Deep learning models often fail when the correlations in the training data are broken, e.g. the classifier might use the presence of a beard to predict gender, thus introducing noise in the evaluation (Beery, Horn, and Perona 2018).

Chemical molecules modeling Finally, we present a practical use-case, in which we apply PluGeN to generate chemical molecules with the requested properties. As a backbone model, we use CharVAE (Gómez-Bombarelli et al. 2018), which is a type of recurrent network used for processing SMILES (Weininger 1988), a textual representation of molecules. It was trained on ZINC 250k database (Sterling and Irwin 2015) of commercially available chemical com-

Requested value	PluGeN	MSP	cFlow
male x beard	0.80	0.79	**0.85**
female x beard	**0.59**	0.33	0.31
male x no-beard	0.88	**0.92**	0.91
female x no-beard	0.85	0.82	**0.95**
male x makeup	**0.44**	0.43	0.29
male x no-makeup	0.72	0.92	**0.96**
female x makeup	0.42	0.41	**0.58**
female x no-makeup	0.55	0.40	**0.85**
smile x open-mouth	0.97	**0.99**	0.79
no-smile x open-mouth	0.79	**0.82**	0.77
smile x calm-mouth	0.84	**0.91**	0.72
no-smile x calm-mouth	0.96	0.97	**0.99**
male x bald	0.26	**0.41**	0.34
male x bangs	0.58	**0.74**	0.45
female x bald	0.19	0.13	**0.39**
female x bangs	0.52	0.49	**0.60**
no-glasses x black-hair	0.92	**0.93**	0.74
no-glasses x golden-hair	**0.92**	0.91	0.81
glasses x black-hair	0.76	**0.90**	0.58
glasses x golden-hair	0.75	**0.85**	0.61
Average Acc	0.69	**0.70**	0.67
Average FID	**28.07**	30.67	39.68

Table 2: Accuracy and FID scores of image manipulation performed on the VAE backbone.

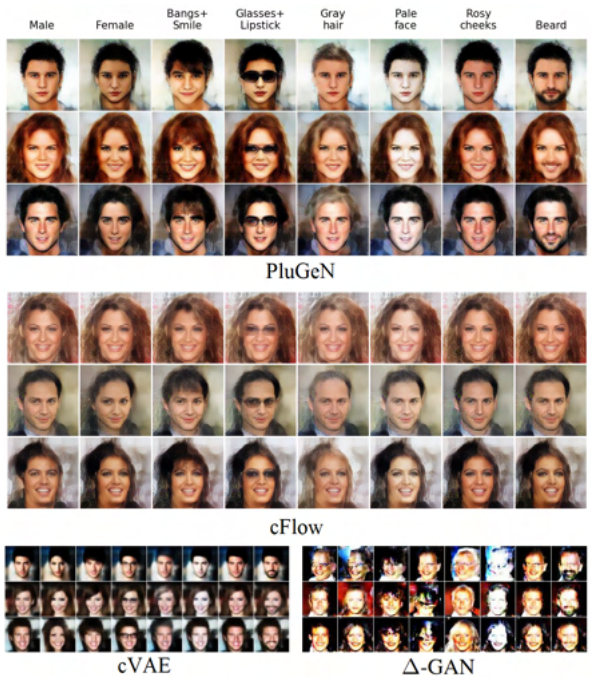


Figure 6: Examples of conditional generation using VAE backbone. Each row contains the same person (style variables) with modified attributes (label variables).

	PluGeN	cFlow	Δ-GAN	cVAE
F1	**0.44**	0.29	0.39	0.39
AUC	**0.76**	0.68	0.70	0.73

Table 3: Results of the independent conditional generation using VAE backbone.

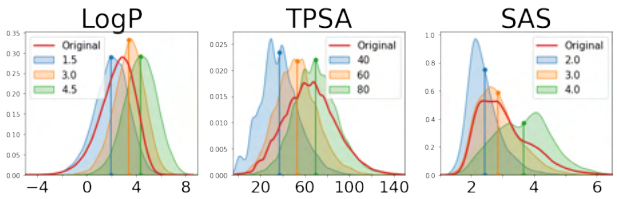


Figure 7: Distribution of attributes of generated molecules, together with distribution for the training dataset. Each color shows the value of a labeled attribute that was used for generation. PluGeN is capable of moving the density of generated molecules' attributes towards the desired value. The average of every distribution is marked with a vertical line.

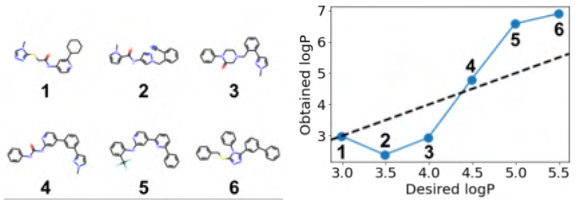
pounds. For every molecule, we model 3 physio-chemical continuous (not binary) labels: logP, SAS, TPSA, which values were calculated using RDKit package [4]. Additional explanations and more examples are given in the supplementary materials.

First, we imitate a practical task of de novo design (Olivecrona et al. 2017; Popova, Isayev, and Tropsha 2018), where we force the model to generate new compounds with desirable properties. For every attribute, we generate 25k molecules with 3 different values: for logP we set the label of generated molecules to: 1.5, 3.0, 4.5; for TPSA we set generated labels to: 40, 60, 80; for SAS we set them to: 2.0, 3.0, 4.0, which gives 9 scenarios in total. From density plots of labels of generated and original molecules presented in Figure 7, we can see that PluGeN changes the distribution of values of the attributes and moves it towards the desired value. A slight discrepancy between desired and generated values may follow from the fact that values of labeled attributes were sampled independently, which could make some combinations physically contradictory.

Next, we consider the setting of lead optimization (Jin et al. 2019; Maziarka et al. 2020), where selected compounds are improved to meet certain criteria. For this purpose, we encode a molecule into the latent representation of INF and force PluGeN to gradually increase the value of logP by 3 and decode the resulting molecules. The obtained molecules together with their logP are shown in Figure 8. As can be seen, PluGeN generates molecules that are structurally similar to the initial one, however with optimized desired attributes.

Obtained results show that PluGeN is able to model the physio-chemical molecular features, which is a non-trivial task that could speed up a long and expensive process of

[4] https://www.rdkit.org/



(a) Molecules decoded from path (b) LogP of presented molecules

Figure 8: Molecules obtained by the model during an optimization phase (left side), and their LogP (right side).

designing new drugs.

Conclusion

We proposed a novel approach for disentangling the latent space of pre-trained generative models, which works perfectly for generating new samples with desired conditions as well as for manipulating the attributes of existing examples. In contrast to previous works, we demonstrated that PluGeN performs well across diverse domains, including chemical molecule modeling, and can be combined with various architectures, such as GANs and VAEs backbones.

Acknowledgements

The research of M. Wołczyk was supported by the Foundation for Polish Science co-financed by the European Union under the European Regional Development Fund in the POIR.04.04.00-00-14DE/18-00 project carried out within the Team-Net programme. The research of M. Proszewska, Ł. Maziarka, M. Zieba and R. Kurczab was supported by the National Science Centre (Poland), grant no. 2018/31/B/ST6/00993, 2019/35/N/ST6/02125, 2020/37/B/ST6/03463 and the Polish National Centre for Research and Development (Grant LIDER/37/0137/L-9/17/NCBR/2018). The research of M. Śmieja was funded by the Priority Research Area DigiWorld under the program Excellence Initiative –– Research University at the Jagiellonian University in Kraków.

Ethical Impact

We did not identify ethical issues concerning our work, as we do not collect data, and we do not foresee malicious applications or societal harm. However, we believe that disentangling factors of variation can have a positive effect on reducing unjust correlations in the data. For example, even though CelebA dataset contains twice as many old men as old women, our method can generate an equal proportion of samples from those classes, thus avoiding amplifying bias in the data.

References

Abdal, R.; Zhu, P.; Mitra, N. J.; and Wonka, P. 2021. Styleflow: Attribute-conditioned exploration of stylegan-generated images using conditional continuous normalizing flows. *ACM Transactions on Graphics (TOG)*, 40(3): 1–21.

Beery, S.; Horn, G. V.; and Perona, P. 2018. Recognition in Terra Incognita. In *ECCV*.

Bengio, Y.; Courville, A.; and Vincent, P. 2013. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8): 1798–1828.

Brakel, P.; and Bengio, Y. 2017. Learning independent features with adversarial nets for non-linear ica. *arXiv preprint arXiv:1710.05050*.

Brock, A.; Donahue, J.; and Simonyan, K. 2018. Large Scale GAN Training for High Fidelity Natural Image Synthesis. In *International Conference on Learning Representations*.

Brown, T. B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; Agarwal, S.; Herbert-Voss, A.; Krueger, G.; Henighan, T.; Child, R.; Ramesh, A.; Ziegler, D. M.; Wu, J.; Winter, C.; Hesse, C.; Chen, M.; Sigler, E.; Litwin, M.; Gray, S.; Chess, B.; Clark, J.; Berner, C.; McCandlish, S.; Radford, A.; Sutskever, I.; and Amodei, D. 2020. Language Models are Few-Shot Learners. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Lin, H., eds., *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Chen, R. T.; Li, X.; Grosse, R.; and Duvenaud, D. 2019. Isolating Sources of Disentanglement in VAEs. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 2615–2625.

Chen, X.; Duan, Y.; Houthooft, R.; Schulman, J.; Sutskever, I.; and Abbeel, P. 2016. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. *arXiv preprint arXiv:1606.03657*.

Choi, Y.; Uh, Y.; Yoo, J.; and Ha, J.-W. 2020. Stargan v2: Diverse image synthesis for multiple domains. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8188–8197.

Dinh, L.; Krueger, D.; and Bengio, Y. 2014. Nice: Nonlinear independent components estimation. *arXiv preprint arXiv:1410.8516*.

Gan, Z.; Chen, L.; Wang, W.; Pu, Y.; Zhang, Y.; Liu, H.; Li, C.; and Carin, L. 2017. Triangle generative adversarial networks. *arXiv preprint arXiv:1709.06548*.

Gao, Y.; Wei, F.; Bao, J.; Gu, S.; Chen, D.; Wen, F.; and Lian, Z. 2021. High-Fidelity and Arbitrary Face Editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 16115–16124.

Gómez-Bombarelli, R.; Wei, J. N.; Duvenaud, D.; Hernández-Lobato, J. M.; Sánchez-Lengeling, B.; Sheberla, D.; Aguilera-Iparraguirre, J.; Hirzel, T. D.; Adams, R. P.; and Aspuru-Guzik, A. 2018. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2): 268–276.

Härkönen, E.; Hertzmann, A.; Lehtinen, J.; and Paris, S. 2020. Ganspace: Discovering interpretable gan controls. *arXiv preprint arXiv:2004.02546*.

He, Z.; Zuo, W.; Kan, M.; Shan, S.; and Chen, X. 2019. Attgan: Facial attribute editing by only changing what you want. *IEEE Transactions on Image Processing*, 28(11): 5464–5478.

Heusel, M.; Ramsauer, H.; Unterthiner, T.; Nessler, B.; and Hochreiter, S. 2017. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In Guyon, I.; von Luxburg, U.; Bengio, S.; Wallach, H. M.; Fergus, R.; Vishwanathan, S. V. N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, 6626–6637.

Higgins, I.; Matthey, L.; Pal, A.; Burgess, C.; Glorot, X.; Botvinick, M.; Mohamed, S.; and Lerchner, A. 2017. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*.

Jang, E.; Gu, S.; and Poole, B. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Jin, W.; Barzilay, R.; and Jaakkola, T. 2018. Junction tree variational autoencoder for molecular graph generation. In *International Conference on Machine Learning*, 2323–2332. PMLR.

Jin, W.; Yang, K.; Barzilay, R.; and Jaakkola, T. 2019. Learning multimodal graph-to-graph translation for molecular optimization. *International Conference on Learning Representations*.

Jo, Y.; and Park, J. 2019. Sc-fegan: Face editing generative adversarial network with user's sketch and color. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 1745–1753.

Kang, S.; and Cho, K. 2018. Conditional molecular design with deep generative models. *Journal of chemical information and modeling*, 59(1): 43–52.

Karras, T.; Laine, S.; and Aila, T. 2019. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4401–4410.

Kaya, Y.; Hong, S.; and Dumitras, T. 2019. Shallow-Deep Networks: Understanding and Mitigating Network Overthinking. In Chaudhuri, K.; and Salakhutdinov, R., eds., *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, 3301–3310. PMLR.

Kim, H.; and Mnih, A. 2018. Disentangling by factorising. In *International Conference on Machine Learning*, 2649–2658. PMLR.

Kingma, D. P.; and Dhariwal, P. 2018. Glow: Generative flow with invertible 1x1 convolutions. *arXiv preprint arXiv:1807.03039*.

Kingma, D. P.; Rezende, D. J.; Mohamed, S.; and Welling, M. 2014. Semi-supervised learning with deep generative models. *arXiv preprint arXiv:1406.5298*.

Klys, J.; Snell, J.; and Zemel, R. 2018. Learning latent subspaces in variational autoencoders. *arXiv preprint arXiv:1812.06190*.

Kodali, N.; Abernethy, J.; Hays, J.; and Kira, Z. 2017. On convergence and stability of gans. *arXiv preprint arXiv:1705.07215*.

Koperski, M.; Konopczynski, T.; Nowak, R.; Semberecki, P.; and Trzcinski, T. 2020. Plugin Networks for Inference under Partial Evidence. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2883–2891.

Kumar, A.; Sattigeri, P.; and Balakrishnan, A. 2017. Variational inference of disentangled latent concepts from unlabeled observations. *arXiv preprint arXiv:1711.00848*.

Lample, G.; Zeghidour, N.; Usunier, N.; Bordes, A.; Denoyer, L.; and Ranzato, M. 2017. Fader networks: Manipulating images by sliding attributes. *arXiv preprint arXiv:1706.00409*.

Li, X.; Lin, C.; Li, R.; Wang, C.; and Guerin, F. 2020. Latent space factorisation and manipulation via matrix subspace projection. In *International Conference on Machine Learning*, 5916–5926. PMLR.

Liu, R.; Liu, Y.; Gong, X.; Wang, X.; and Li, H. 2019. Conditional adversarial generative flow for controllable image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 7992–8001.

Locatello, F.; Bauer, S.; Lucic, M.; Raetsch, G.; Gelly, S.; Schölkopf, B.; and Bachem, O. 2019. Challenging common assumptions in the unsupervised learning of disentangled representations. In *international conference on machine learning*, 4114–4124. PMLR.

Maziarka, Ł.; Pocha, A.; Kaczmarczyk, J.; Rataj, K.; Danel, T.; and Warchoł, M. 2020. Mol-CycleGAN: a generative model for molecular optimization. *Journal of Cheminformatics*, 12(1): 1–18.

Mirza, M.; and Osindero, S. 2014. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*.

Nitzan, Y.; Bermano, A.; Li, Y.; and Cohen-Or, D. 2020. Disentangling in latent space by harnessing a pretrained generator. *arXiv preprint arXiv:2005.07728*, 2(3).

Olivecrona, M.; Blaschke, T.; Engkvist, O.; and Chen, H. 2017. Molecular de-novo design through deep reinforcement learning. *Journal of cheminformatics*, 9(1): 1–14.

Park, T.; Liu, M.-Y.; Wang, T.-C.; and Zhu, J.-Y. 2019. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2337–2346.

Perarnau, G.; Van De Weijer, J.; Raducanu, B.; and Álvarez, J. M. 2016. Invertible conditional gans for image editing. *arXiv preprint arXiv:1611.06355*.

Popova, M.; Isayev, O.; and Tropsha, A. 2018. Deep reinforcement learning for de novo drug design. *Science advances*, 4(7): eaap7885.

Rebuffi, S.; Bilen, H.; and Vedaldi, A. 2017. Learning multiple visual domains with residual adapters. In Guyon, I.; von

Luxburg, U.; Bengio, S.; Wallach, H. M.; Fergus, R.; Vishwanathan, S. V. N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, 506–516.

Shen, Y.; Yang, C.; Tang, X.; and Zhou, B. 2020. Interfacegan: Interpreting the disentangled face representation learned by gans. *IEEE transactions on pattern analysis and machine intelligence*.

Sohn, K.; Lee, H.; and Yan, X. 2015. Learning structured output representation using deep conditional generative models. *Advances in neural information processing systems*, 28: 3483–3491.

Sorrenson, P.; Rother, C.; and Köthe, U. 2020. Disentanglement by nonlinear ica with general incompressible-flow networks (gin). *arXiv preprint arXiv:2001.04872*.

Spurek, P.; Nowak, A.; Tabor, J.; Maziarka, Ł.; and Jastrzebski, S. 2020. Non-linear ICA based on Cramer-Wold metric. In *International Conference on Neural Information Processing*, 294–305. Springer.

Sterling, T.; and Irwin, J. J. 2015. ZINC 15–ligand discovery for everyone. *Journal of chemical information and modeling*, 55(11): 2324–2337.

Tewari, A.; Elgharib, M.; Bernard, F.; Seidel, H.-P.; Pérez, P.; Zollhöfer, M.; and Theobalt, C. 2020. Pie: Portrait image embedding for semantic control. *ACM Transactions on Graphics (TOG)*, 39(6): 1–14.

Tov, O.; Alaluf, Y.; Nitzan, Y.; Patashnik, O.; and Cohen-Or, D. 2021. Designing an encoder for stylegan image manipulation. *ACM Transactions on Graphics (TOG)*, 40(4): 1–14.

Wang, H.-P.; Yu, N.; and Fritz, M. 2021. Hijack-gan: Unintended-use of pretrained, black-box gans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 7872–7881.

Weininger, D. 1988. SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. *Journal of chemical information and computer sciences*, 28(1): 31–36.

Wołczyk, M.; Wójcik, B.; Bałazy, K.; Podolak, I.; Tabor, J.; Śmieja, M.; and Trzciński, T. 2021. Zero Time Waste: Recycling Predictions in Early Exit Neural Networks. *arXiv preprint arXiv:2106.05409*.

Yan, X.; Yang, J.; Sohn, K.; and Lee, H. 2016. Attribute2image: Conditional image generation from visual attributes. In *European Conference on Computer Vision*, 776–791. Springer.

Zhou, W.; Xu, C.; Ge, T.; McAuley, J. J.; Xu, K.; and Wei, F. 2020. BERT Loses Patience: Fast and Robust Inference with Early Exit. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Lin, H., eds., *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Multi-Label Conditional Generation from Pre-trained Models

Magdalena Proszewska, Maciej Wołczyk, Maciej Zieba, Patryk Wielopolski, Łukasz Maziarka, Marek Śmieja

Abstract—Although modern generative models achieve excellent quality in a variety of tasks, they often lack the essential ability to generate examples with requested properties, such as the age of the person in the photo or the weight of the generated molecule. To overcome these limitations we propose PluGeN (Plugin Generative Network), a simple yet effective generative technique that can be used as a plugin for pre-trained generative models. The idea behind our approach is to transform the entangled latent representation using a flow-based module into a multi-dimensional space where the values of each attribute are modeled as an independent one-dimensional distribution. In consequence, PluGeN can generate new samples with desired attributes as well as manipulate labeled attributes of existing examples. Due to the disentangling of the latent representation, we are even able to generate samples with rare or unseen combinations of attributes in the dataset, such as a young person with gray hair, men with make-up, or women with beards. In contrast to competitive approaches, PluGeN can be trained on partially labeled data. We combined PluGeN with GAN and VAE models and applied it to conditional generation and manipulation of images, chemical molecule modeling and 3D point clouds generation.

Index Terms—deep generative models, conditional generation, pre-trained models, invertible normalizing flows, GANs, VAEs.

1 Introduction

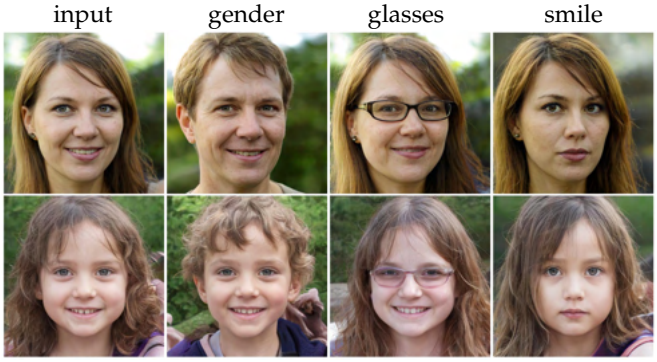


Fig. 1: Attributes manipulation performed by PluGeN using the StyleGAN backbone.

Generative models such as GANs and variational autoencoders have achieved great results in recent years, especially in the domains of images [1] and cheminformatics [2]. However, in many practical applications, we need to control the process of creating samples by enforcing particular features of generated objects. This would be required to regulate the biases present in the data, e.g. to assure that people of each ethnicity are properly represented in the generated set of face images. In numerous realistic problems, such as drug discovery, we want to find objects with desired properties, like molecules with a particular activity, non-toxicity, and solubility.

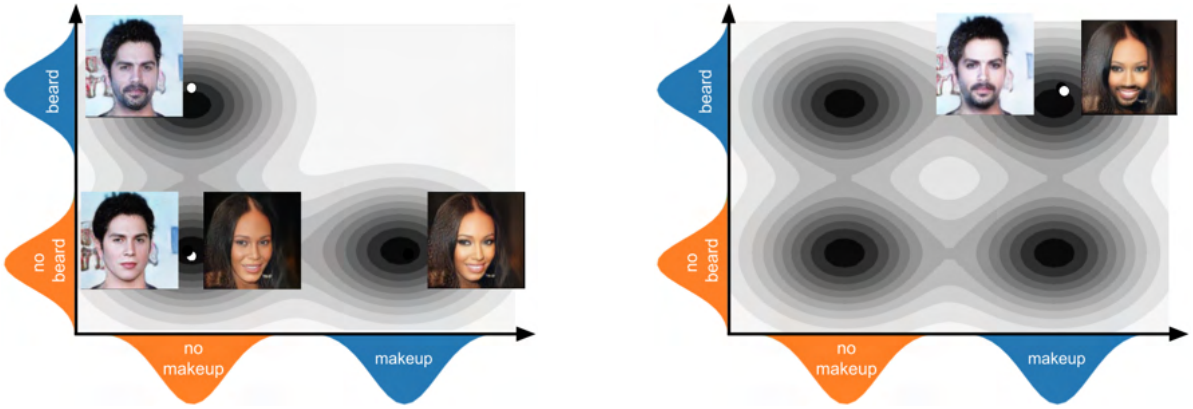
Designing the conditional variants of generative models that operate on multiple labels is a challenging problem due to intricate relations among the attributes. Practically, it means that some combinations of attributes (e.g. a woman with a beard) might be unobserved or rarely observed in the training data. In essence, the model should be able to go beyond the distribution of seen data and generate examples with combinations of attributes not encountered previously. One might approach this problem by building a new conditional generative model from the ground up or designing a solution tailored to a specific existing unsupervised generative model. However, this introduces an additional effort when one wants to adapt it to a newly invented approach.

To tackle this problem while leveraging the power of existing techniques, we propose PluGeN (Plugin Generative Network), a simple yet effective generative technique that can be used as a plugin to various pre-trained generative models such as VAEs or GANs, see Figure 1 for demonstration. Making use of PluGeN, we can manipulate the attributes of input examples as well as generate new samples with desired features. When training the proposed module, we do not change the parameters of the base model and thus we retain its generative and reconstructive abilities, which places our work in the emerging family of non-invasive network adaptation methods [3], [4]. In contrast to competitive approaches [5], [6], [7], PluGeN can be trained on partially labeled data, which is of great practical importance in real-life problems because of the substantial cost of labeling.

Our idea is to find a mapping between the entangled latent representation of the backbone model and a disentangled space, where each dimension corresponds to a

- *M. Proszewska, M. Wołczyk, Ł. Maziarka, M. Śmieja are with Faculty of Mathematics and Computer Science, Jagiellonian University. E-mail: marek.smieja@uj.edu.pl*
- *P. Wielopolski is with Wroclaw University of Science and Technology.*
- *M. Zieba is with Tooploox and Wroclaw University of Science and Technology.*

This is the extended version of a paper accepted to AAAI 2022 and IEEE Transactions on Pattern Analysis and Machine Intelligence March 14th, 2024.



(a) Factorization of true data distribution

(b) Probability distribution covered by PluGeN.

Fig. 2: PluGeN factorizes true data distribution into components (marginal distributions) related to labeled attributes, see (a), and allows for describing unexplored regions of data (uncommon combinations of labels) by sampling from independent components, see (b). In the case illustrated here, PluGeN constructs pictures of men with make-up or women with beards, although such examples rarely (or never) appear in the training set.

single, interpretable attribute of the image. By factorizing the true data distribution into independent components, we can sample from each component independently, which results in creating samples with arbitrary combinations of attributes, see Figure 2. In contrast to many previous works, which are constrained to the attributes combinations visible in the training set, PluGeN gives us full control of the generation process, being able to create uncommon combinations of attributes, such as a woman with a beard or a man with heavy make-up. Generating samples with unseen combinations of attributes can be viewed as extending the distribution of generative models to unexplored although reasonable regions of data space, which distinguishes our approach from the existing solutions.

Extensive experiments performed on the domain of images, chemical molecules and 3D point clouds demonstrate that PluGeN is a reusable plugin that can be applied to various architectures including GANs and VAEs. In contrast to the baselines, PluGeN can generate new samples as well as manipulate the properties of existing examples, being capable of creating uncommon combinations of attributes. PluGeN preserves its impressive performance in a semi-supervised setting even if less than 15% of labels are available in training on the StyleGAN backbone. In addition to the experimental study, we present an in-depth discussion comparing the proposed mechanism of modeling conditional distribution with the baseline conditioning approach used e.g. in StyleFlow [5] or FPN (flow plug-in network) [6].

This paper extends our conference publication [8] in the following aspects:

- We introduce a semi-supervised learning approach for training PluGeN on partially labeled data and examine its performance on a dataset of face images (Section 3.4).
- We discuss our conditioning mechanism in relation to the baseline one applied in StyleFlow (Section 3.6). Both approaches are illustrated on a toy example (Figure 4) and evaluated across various large-scale datasets (Section 5).
- We apply PluGeN in the domain of 3D point clouds, which further confirms that our model can be used in diverse applications (Section 6.2).
- Additionally, we clarify the parameterization of PluGeN in the case of continuous target attributes (last paragraph in Section 3.2).

2 Related Work

Conditional VAE (cVAE) is one of the first methods that include additional information about the labeled attributes in a generative model [9]. Although this approach has been widely used in various areas ranging from image generation [10] to molecular design [11], the independence of the latent vector from the attribute data is not assured, which negatively influences the generation quality. Conditional GAN (cGAN) is an alternative approach that gives results of significantly better quality [12], but the model is more difficult to train [13]. cGAN works very well for generating new images and conditioning factors may take various forms (images, sketches, labels) [14], but manipulating existing examples is more problematic because GAN models lack the encoder network [15]. Fader Networks [16] combine features of both cVAE and cGAN, as they use encoder-decoder architecture, together with the discriminator, which predicts the image attributes from its latent vector returned from the encoder. As discussed in [7], the training of Fader Networks is even more difficult than standard GANs, and disentanglement of attributes is not preserved. MSP [7] is a recent auto-encoder based architecture with an additional projection matrix, which is responsible for disentangling the latent space and separating the attribute information from other characteristic information. In contrast to PluGeN, MSP cannot be used with pre-trained GANs and performs poorly at generating new images (it was designed for manipulating existing examples). CAGlow [17] is an adaptation of Glow [18] to conditional image generation based on modeling a joint probabilistic density of an image and its conditions. Since CAGlow does not reduce data dimension, applying it to more complex data might be problematic.

While the above approaches focus on building conditional generative models from scratch, recent works often focus on manipulating the latent codes of pre-trained models. StyleFlow [5] operates on the latent space of StyleGAN [19] using a conditional continuous flow module. Although the quality of generated images is impressive, the model has not been applied to other generative models than StyleGAN and domains other than images. Moreover, StyleFlow needs an additional classifier to compute the conditioning factor (labels) for images at test time. Competitive approaches to StyleGAN appear in [20], [21], [22]. InterFaceGAN [23], [24] postulates that various properties of the facial semantics can be manipulated via linear models applied to the latent space of GANs. Hijack-GAN [25] goes beyond linear models and designs a proxy model to traverse the latent space of GANs.

To reduce the need for labeled data, unsupervised latent semantic factorizations in GANs were proposed. In GANSpace [26], the key latent directions are identified based on the Principal Component Analysis applied in the latent space of pre-trained GAN. Following this direction, the authors of SeFa [27] decomposed the pre-trained weights of the generator. Although these techniques are very attractive from a practical point of view, the user has to put extra effort into understanding the meaning of the output directions. Since there is no direct correspondence between the labeled attributes and the features returned by these methods, they cannot be easily compared with our approach.

In disentanglement learning, we assume that the data has been generated from a fixed number of independent factors of underlying variation. The goal is then to find a transformation that unravels these factors so that a change in one dimension of the latent space corresponds to a change in one factor of variation while being relatively invariant to changes in other factors [28], [29], [30]. As theoretically shown in [31], the unsupervised learning of disentangled representations is fundamentally impossible without inductive biases on both the models and the data. In this paper, we solve a slightly different problem than typical disentanglement, as we aim to deliver an efficient plug-in model to a large variety of existing models in order to manipulate attributes without training the entire system. Creating compact add-ons for large models saves training time and energy consumption.

3 Plugin Generative Network

We propose a plugin generative network (PluGeN), which can be attached to pre-trained generative models and allows for direct manipulation of labeled attributes, see Figure 3 for the basic scheme of PluGeN. Making use of PluGeN we preserve all properties of the base model, such as generation quality and reconstruction in the case of auto-encoders, while adding new functionalities. In particular, we can:

- modify selected attributes of existing examples,
- generate new samples with desired labels.

In contrast to typical conditional generative models, PluGeN is capable of creating examples with rare or even unseen combinations of attributes, e.g. man with a makeup. Moreover, it can be trained on partially labeled datasets.

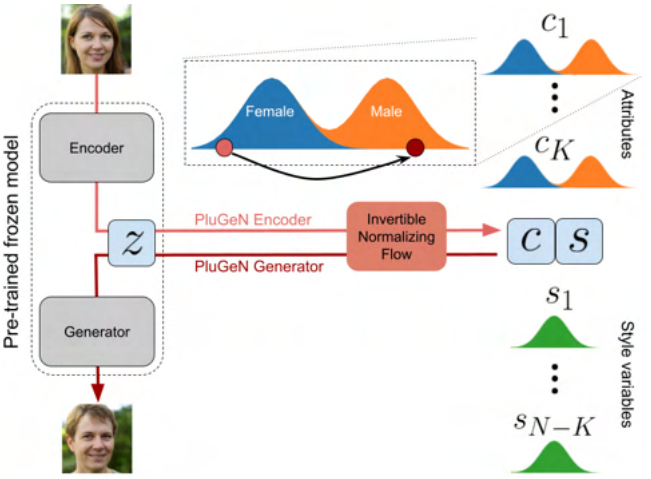


Fig. 3: PluGeN maps the entangled latent space $\mathcal{Z}$ of pre-trained generative models using invertible normalizing flow into a separate space, where labeled attributes are modeled using independent 1-dimensional distributions. By manipulating label variables in this space, we fully control the generation process.

3.1 Probabilistic model

PluGeN works in a multi-label setting, where every example $\mathbf{x} \in \mathcal{X}$ is associated with a K-dimensional vector of attributes $\mathbf{y} = (y_1, \ldots, y_K) \in \mathbb{R}^K$. Each attribute y_k can be a continuous, discrete, or binary variable. We assume that there is a pre-trained generative model $\mathcal{G} : \mathcal{Z} \to \mathbb{R}^D$, where $\mathcal{Z} \subset \mathbb{R}^N$ is the latent space, which is usually heavily entangled. That is, although each latent code $\mathbf{z} \in \mathcal{Z}$ contains the information about the labels $\mathbf{y}$, there is no direct way to extract or modify it.

We want to map this entangled latent space $\mathcal{Z}$ into a separate latent space $\mathcal{D} \subset \mathbb{R}^N$ which encodes the values of each label y_k as a separate random variable C_k living in a single dimension of this space. Thus, by changing the value of C_k, going back to the entangled space $\mathcal{Z}$ and generating a sample, we can control the values of y_k. Since labeled attributes usually do not fully describe a given example, we consider additional $N - K$ random variables S_k, which are supposed to encode the information not included in the labels. We call $\mathbf{C} = (C_1, \ldots, C_K)$ the label variables (or attributes) and $\mathbf{S} = (S_1, \ldots, S_{N-K})$ the style variables.

Since we want to control the value of each attribute independently of any other factors, we assume the factorized form of the probability distribution of the random vector $(\mathbf{C}, \mathbf{S})$. More precisely, the conditional probability distribution of $(\mathbf{C}, \mathbf{S})$ given any condition $\mathbf{Y} = \mathbf{y}$ imposed on labeled attributes is of the form:

$$p_{\mathbf{C},\mathbf{S}|\mathbf{Y}=\mathbf{y}}(\mathbf{c}, \mathbf{s}) = \prod_{i=1}^{K} p_{C_i|Y_i=y_i}(c_i) \cdot p_{\mathbf{S}}(\mathbf{s}), \tag{1}$$

for all $(\mathbf{c}, \mathbf{s}) = (c_1, \ldots, c_K, s_1, \ldots, s_{N-K}) \in \mathbb{R}^N$. In other words, modifying $Y_i = y_i$ influences only the i-th factor C_i leaving other features unchanged. We assume that label variables represent the first K dimensions in the target latent space $\mathcal{D}$ while the style variables are the subsequent dimensions.

3.2 Parameterization

To instantiate the above probabilistic model (1), we need to parametrize the conditional distribution of C_i given $Y_i = y_i$ and the distribution of $\mathbf{S}$. Since we do not impose any constraints on style variables, we use standard Gaussian distribution for modeling density of $\mathbf{S}$:

$$p_{\mathbf{S}} = \mathcal{N}(0, I_{N-K}).$$

Binary attributes. To provide the consistency with $p_{\mathbf{S}}$ and avoid potential problems with training our deep learning model using discrete distributions, we use the mixture of two Gaussians for modeling the presence of labels – each component corresponds to a potential value of the label (0 or 1). More precisely, the conditional distribution of C_i given $Y_i = y_i$ is parametrized by:

$$p_{C_i|Y_i=y_i} = \mathcal{N}(m_0, \sigma_0)^{(1-y_i)} \cdot \mathcal{N}(m_1, \sigma_1)^{y_i}, \tag{2}$$

where $m_0, m_1, \sigma_0, \sigma_1$ are the user-defined parameters. If $y_i = 0$, then the latent factor C_i takes values close to m_0; otherwise we get values around m_1 (depending on the value of σ_0 and σ_1). To provide good separation between components, we put $m_0 = -1, m_1 = 1$; the selection of σ_0, σ_1 will be discussed is the supplementary materials.

Thanks to this continuous parametrization, we can smoothly interpolate between different labels, which would not be so easy using e.g. Gumbel softmax parametrization [32]. In consequence, we can gradually change the intensity of certain labels, like smile or beard, even though such information was not available in a training set (see Figure 5 in the experimental section).

Continuous attributes. Without loss of generality, we assume that $y_i \in [-1, 1]$. Analogically to the case of binary labels, we assume that the conditional distribution of label variable C_i given $Y_i = y_i$ is parametrized by

$$p_{C_i|Y_i=y_i} = \mathcal{N}(y_i, \sigma),$$

where $\sigma > 0$ is the user-defined parameter controlling smoothness.

For high values of σ there is a huge overlap between Gaussian components. This results in small penalties in terms of negative log-likelihood for incorrect assignments. From a practical perspective, we start the training process with high values of σ, which provides reasonable initialization of PluGeN. Next, we gradually decrease σ to match the correct assignments.

3.3 Training the model

To establish a two-way mapping between entangled space $\mathcal{Z}$ and the disentangled space $\mathcal{D}$, we use an invertible normalizing flow (INF), $\mathcal{F} : \mathbb{R}^N \to \mathcal{Z}$. Let us recall that INF is a neural network, where the inverse mapping is given explicitly and the Jacobian determinant can be easily calculated [30]. Due to the invertibility of INF, we can transform latent codes $\mathbf{z} \in \mathcal{Z}$ to the prior distribution of INF, modify selected attributes, and map the resulting vector back to $\mathcal{Z}$. Moreover, INFs can be trained using log-likelihood loss, which is very appealing in generative modeling.

Summarizing, given a latent representation $\mathbf{z} \in \mathcal{Z}$ of a sample $\mathbf{x}$ with label $\mathbf{y}$, the loss function of PluGeN equals:

$$\begin{aligned}
&-\log p_{\mathbf{Z}|\mathbf{Y}=\mathbf{y}}(\mathbf{z}) = \\
&\qquad -\log\left(p_{\mathbf{C},\mathbf{S}|\mathbf{Y}=\mathbf{y}}(\mathbf{c},\mathbf{s}) \cdot \left|\det \frac{\partial \mathcal{F}^{-1}(\mathbf{z})}{\partial \mathbf{z}}\right|\right) = \\
&-\log\left(\prod_{i=1}^{K} p_{C_i|Y_i=y_i}(c_i) \cdot p_{\mathbf{S}}(\mathbf{s})\right) - \log\left|\det \frac{\partial \mathcal{F}^{-1}(\mathbf{z})}{\partial \mathbf{z}}\right| = \\
&-\sum_{i=1}^{K} \log p_{C_i|Y_i=y_i}(c_i) - \log p_{\mathbf{S}}(\mathbf{s}) - \log\left|\det \frac{\partial \mathcal{F}^{-1}(\mathbf{z})}{\partial \mathbf{z}}\right|,
\end{aligned} \tag{3}$$

where $(\mathbf{c}, \mathbf{s}) = \mathcal{F}^{-1}(\mathbf{z})$. In the training phase, we collect latent representations $\mathbf{z}$ of data points $\mathbf{x}$. Making use of labeled attributes $\mathbf{y}$ associated with every $\mathbf{x}$, we modify the weights of $\mathcal{F}$ so that to minimize the negative log-likelihood (3) using gradient descent. The weights of the base model $\mathcal{G}$ are kept frozen.

3.4 Semi-supervised training

In contrast to many previous works [5], [6], [7], PluGeN can be trained in a semi-supervised setting, where only partial information about labeled attributes is available. The proposed approach is based on the fact that the labeled attributes are modeled as target variables. In other words, to apply the INF mapping $\mathcal{F}^{-1}$ on the latent vector $\mathbf{z}$ we do not need to know the values of labels. They are predicted by the model itself by finding a factorized representation into attributes and style variables for the raw input $\mathbf{z}$ taken from the latent space of the backbone generative model.

To train PluGeN, we need to evaluate the negative log-likelihood of the latent vector $\mathbf{z}$ representing a training example $\mathbf{x}$. Suppose that we only know the values of the subset of attributes $\mathbf{Y}_I = \{Y_i\}_{i \in I}$, where $I \subset \{1, \ldots, K\}$ denotes selected indices. The negative log-likelihood of $\mathbf{z}$ with known attributes $\mathbf{Y}_I = \mathbf{y}_I$ equals:

$$\begin{aligned}
-\log p_{\mathbf{Z}|\mathbf{Y}_I=\mathbf{y}_I}(\mathbf{z}) = &-\sum_{i \in I} \log p_{C_i|Y_i=y_i}(c_i) - \sum_{i \notin I} \log p_{C_i}(c_i) \\
&- \log p_{\mathbf{S}}(\mathbf{s}) - \log\left|\det \frac{\partial \mathcal{F}^{-1}(\mathbf{z})}{\partial \mathbf{z}}\right|.
\end{aligned} \tag{4}$$

The known values of attributes affect only the variables $\{C_i\}_{i \in I}$ because we use the factorized form of the base distribution (1). For the remaining variables $\{C_i\}_{i \notin I}$, we use the total probability calculated from the conditional base distribution:

$$p_{C_i}(c_i) = \sum_{y_i \in Y_i} p_{Y_i}(y_i) p_{C_i|Y_i=y_i}(c_i). \tag{5}$$

Due to the assumption of the independence between latent variables, we compute the total probability in each dimension individually, which is numerically efficient.

In the following parts, we discuss two strategies of estimating the total probability, Eq. (5).

KDE approach. In the most straightforward strategy, we use all training examples with the known i-th attribute for

evaluating p_{C_i}. More precisely, given training set $\mathcal{X}$, the total probability can be estimated by:

$$p_{C_i} = \frac{1}{|\mathcal{X}|} \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{X}} p_{C_i|Y_i=y_i},$$

where $\mathbf{y} = (y_1, \ldots, y_K)$ denotes the label vector of $\mathbf{x}$.

In the case of binary attributes, this distribution equals:

$$p_{C_i} = p_0\mathcal{N}(m_0, \sigma_0) + p_1\mathcal{N}(m_1, \sigma_1),$$

where p_0, p_1 are the fractions of negatively and positively labeled examples in a training set $\mathcal{X}$. For continuous case, it is given by:

$$p_{C_i} = \frac{1}{|\mathcal{X}|} \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{X}} \mathcal{N}(y_i, \sigma).$$

Observe that these densities coincide with 1-dimensional kernel density estimators (KDEs). Although KDE does not work well in high-dimensional spaces, it is a reliable estimate of the probability density function in the 1-dimensional situation considered here.

Uniform distribution. In the previous strategy, we attach a higher probability to more likely values of attributes. In consequence, we encourage the model to map the unlabeled examples to more common values of a given attribute. While this is logically correct, this situation represents an unbalanced case and can lead to ignoring less likely events. Moreover, evaluating KDE can be slow for continuous attributes.

To cope with these problems, we propose a second approach, which relies on using uniform distribution instead of KDE. More precisely, we assume that the likelihood of any value in p_{C_i} is the same. In this case, the formula (4) reduces to:

$$-\log p_{\mathbf{Z}|\mathbf{Y}_I=\mathbf{y}_I}(\mathbf{z}) = -\sum_{i\in I} \log p_{C_i|Y_i=y_i}(c_i) - \text{const} - \log p_{\mathbf{S}}(\mathbf{s}) - \log\left|\det\frac{\partial\mathcal{F}^{-1}(\mathbf{z})}{\partial\mathbf{z}}\right|.$$

The constant term can be ignored in the optimization. In consequence, we evaluate the likelihood only for the known values of the attribute, which further speeds up the training.

3.5 Inference

We may use PluGeN to generate new samples with desired attributes as well as to manipulate attributes of input examples. In the first case, we generate a vector $(\mathbf{c}, \mathbf{s})$ from the conditional distribution $p_{\mathbf{C},\mathbf{S}|\mathbf{Y}=\mathbf{y}}$ with selected condition $\mathbf{y}$. To get the output sample, the vector $(\mathbf{c}, \mathbf{s})$ is transformed by the INF and the base generative network $\mathcal{G}$, which gives us the final output $\mathbf{x} = \mathcal{G}(\mathcal{F}(\mathbf{c}, \mathbf{s}))$.

In the second case, to manipulate the attributes of an existing example $\mathbf{x}$, we need to find its latent representation $\mathbf{z}$. If $\mathcal{G}$ is a decoder network of an autoencoder model, then $\mathbf{x}$ should be passed through the encoder network to obtain $\mathbf{z}$ [7]. If $\mathcal{G}$ is a GAN, then $\mathbf{z}$ can be found by minimizing the reconstruction error between $\mathbf{x}' = \mathcal{G}(\mathbf{z})$ and $\mathbf{x}$ using gradient descent for a frozen $\mathcal{G}$ [5]. In both cases, $\mathbf{z}$ is next processed by INF, which gives us its factorized representation $(\mathbf{c}, \mathbf{s}) = \mathcal{F}^{-1}(\mathbf{z})$. In this representation, we can modify any labeled variable c_i and map the resulting vector back through $\mathcal{F}$ and $\mathcal{G}$ as in the generative case.

3.6 Relation to the baseline conditioning

In this section, we discuss the proposed conditioning mechanism (instantiated by PluGeN) in relation to the baseline conditioning used in StyleFlow.

Baseline conditional INF. Many recent works, including StyleFlow [5] and FPN [6], use conditional INF to directly model conditional distribution. In this case, we do not factorize the base distribution into label and style variables $(\mathbf{C}, \mathbf{S})$, but use a standard Gaussian $\mathcal{N}(0, I_N)$ as a base distribution $p_{\mathbf{W}}$. The condition given by the attribute vector $\mathbf{y}$ is introduced by parameterizing the flow transformation $\mathcal{F}$ with $\mathbf{y}$. The form of incorporating conditional information depends strictly on the type of the INF, see [6] for details. The log-likelihood of a sample represented by a latent vector $\mathbf{z} \in \mathcal{Z}$ with the attributes $\mathbf{y}$ is given by:

$$\begin{aligned} -\log p_{\mathbf{Z}|\mathbf{Y}=\mathbf{y}}(\mathbf{z}) = \\ -\log\left(p_{\mathbf{W}}(\mathcal{F}^{-1}(\mathbf{z}|\mathbf{y}))\cdot\left|\det\frac{\partial\mathcal{F}^{-1}(\mathbf{z}|\mathbf{y})}{\partial\mathbf{z}}\right|\right) = \\ -\log\left(p_{\mathbf{W}}(\mathbf{w})\right) - \log\left|\det\frac{\partial\mathcal{F}^{-1}(\mathbf{z}|\mathbf{y})}{\partial\mathbf{z}}\right| \end{aligned} \quad (6)$$

where $\mathbf{w} = \mathcal{F}^{-1}(\mathbf{z}|\mathbf{y})$.

Observe that the selection of the conditioning factor $\mathbf{y}$ induces an individual form of INF transformation. In consequence, we do not obtain the formula for the likelihood of the latent vector $\mathbf{z}$ but only the conditional likelihood of $\mathbf{z}$ given $\mathbf{y}$. To calculate the (unconditional) likelihood of $\mathbf{z}$, we have to integrate over all conditioning factors:

$$p_{\mathbf{Z}}(\mathbf{z}) = \sum_{\mathbf{y}\in\mathbf{Y}} p_{\mathbf{Y}}(\mathbf{y}) p_{\mathbf{Z}|\mathbf{Y}=\mathbf{y}}(\mathbf{z}).$$

Evaluating this likelihood is difficult if the number of possible values of $\mathbf{Y}$ is large or the attribute vector is continuous. We need to perform as many forward passes as the cardinality of the value set of $\mathbf{Y}$. Consequently, if one wants to use conditional INF as a classifier for predicting attributes $\mathbf{y}$ from $\mathbf{z}$, then it is necessary to find:

$$\arg\max_{\mathbf{y}\in\mathbf{Y}} p_{\mathbf{Y}}(\mathbf{y}) p_{\mathbf{Z}|\mathbf{Y}=\mathbf{y}}(\mathbf{z}).$$

As before, this formula is expensive and cannot be used for attributes with large cardinality. Moreover, as we show in Figure 4, decision boundaries generated by such a classifier are highly irregular.

There is another practical disadvantage of modeling conditional distribution using conditional INF. Suppose that we want to modify the attributes of a given input example $\mathbf{x}$ represented by the latent vector $\mathbf{z}$. If $\mathbf{y}$ is the attribute vector of $\mathbf{x}$, then we pass $\mathbf{z}$ using conditional INF and obtain $\mathbf{w} = \mathcal{F}^{-1}(\mathbf{z}|\mathbf{y})$. Next, we apply the inverse transformation but with the requested condition $\mathbf{y}'$. This gives us the new latent representation $\mathbf{z}' = \mathcal{F}(\mathbf{w}|\mathbf{y}')$, which can be decoded using generator network $\mathbf{x}' = \mathcal{G}(\mathbf{z}')$. This procedure is feasible if we know the true attribute vector $\mathbf{y}$ of the input example $\mathbf{x}$ we want to modify. If this information is not provided, we need to employ an auxiliary classifier to predict the attributes. This introduces a bias to the manipulation procedure related to the classifier errors. Additionally, this makes the whole procedure more complex as we need to train the classifier and collect its predictions for each

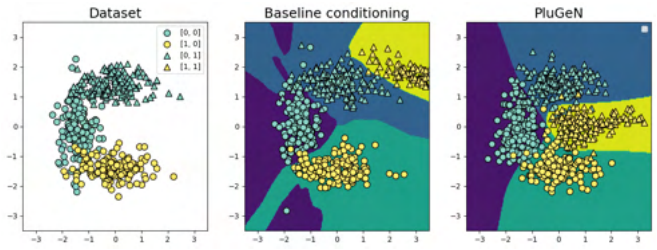


Fig. 4: Comparison of handling data with the baseline conditional INF vs. PluGeN. On the left, we show the dataset with one of the groups (corresponding to attributes vector $[1, 1]$) missing. In the middle and on the right we show samples obtained from, respectively, baseline conditional INF and PluGeN. The background color represents the probability of the most likely class induced by the conditional distributions of the models.

example we want to modify. This reasoning also prevents us from applying the semi-supervised training as we need to know labels before mapping the latent representations using conditional INF. As we show in supplementary materials, conditional INF performs poorly if we use incorrect labels for its parameterization.

PluGeN conditioning Making use of PluGeN, we model the joint distribution of attribute and style variables. Since we use a factorized form of the base distribution, we can easily calculate and evaluate the conditional distribution, see Eq. (3). One could argue that modeling the joint distribution is a more difficult task, but conducted experiments confirm that our model gives comparable or even better results than the conditional INF.

Moreover, the attributes modification is more viable in practice as we do not need to know the true attributes of manipulated examples, see Section 3.5. Given the latent representation of the input example, PluGeN directly predicts its attributes together with the style variables. In consequence, we can modify any of the attributes by setting the value of the corresponding coordinate to the requested value. In addition, modeling attributes as target variables instead of conditioning factors enables us to train the model using partially labeled data.

Since we use a single (unconditional) INF and directly split the latent space into regions corresponding to class labels, PluGeN might be better suited to understand the geometry of data. In order to investigate this question, we consider a simple example of two-dimensional data with two binary attributes shown in Figure 4 (left), where the group with attributes vector $[1, 1]$ (yellow triangles) is not present in the data. We train the baseline conditional INF and PluGeN on the data and check where they generate missing class. Although samples representing the combinations present in the dataset look similar for both models, the position of the unseen attribute class is different, see Figure 4 (middle and right). While the baseline model generates these points somewhat randomly, the geometrical formulation of PluGeN serves as an inductive bias to position the remaining class between the existing ones. Additionally, for the conditional INF, the regions representing the most likely class regions are non-regular and less predictable as we go away from the data. In the case of PluGeN, decision boundaries are smooth over the whole data space, which allows for better extrapolation. This example justifies our thesis that PluGeN is capable of generating samples with unseen combinations of attributes (compare this example with Figure 2).

4 Experiments on fully labeled image data

To empirically evaluate the properties of PluGeN, we combine it with two GAN models and one VAE architecture to manipulate attributes of image data.

4.1 StyleGAN backbone

First, we consider the state-of-the-art StyleGAN architecture [19], which was trained on Flickr-Faces-HQ (FFHQ) containing 70 000 high-quality images of resolution 1024×1024. The Microsoft Face API was used to label 8 attributes in each image (gender, pitch, yaw, eyeglasses, age, facial hair, expression, and baldness).

PluGeN is instantiated using NICE flow model [30] that operates on the latent vectors $\mathbf{w} \in \mathbb{R}^{512}$ sampled from the $\mathbf{W}$ space of the StyleGAN. As a baseline, we select StyleFlow [5], which is currently one of the state-of-the-art models for controlling the generation process of StyleGAN. In contrast to PluGeN, StyleFlow uses the conditional continuous INF to operate on the latent codes of StyleGAN, where the conditioning factor corresponds to the labeled attributes.

Figures 1 (first page) and 5 present the effects of how PluGeN and StyleFlow manipulate images sampled by StyleGAN. It is evident that PluGeN can switch the labels to opposite values as well as gradually change their intensities. At the same time, the requested modifications do not influence the remaining attributes leaving them unchanged. One can observe that the results produced by StyleFlow are also acceptable, but the modification of the requested attribute implies the change of other attributes. For example, increasing the intensity of "baldness" changes the type of glasses, or turning the head into right makes the woman look straight.

For qualitative evaluation, we modify one of 5 attributes[1] and inspect the resulting image. First, we measure the accuracy of changing a certain attribute using Microsoft Face API. Second, we verify how much of the image content was modified when manipulating the requested attributes. To this end, we compare the altered picture with the original image (before modification). To compare the difference between images, we calculate the mean square error (MSE) between embeddings of the original and modified images taken from a pre-trained network. We employ two networks applicable to processing face images: ArcFace[2] **[?]** (AF) and FaceRecognition[3] (FR). A model with a lower MSE preserves more features (including identity) from the original image. The quality of images is additionally assessed by calculating the standard Fréchet Inception Distance (FID) [33].

1. The remaining 3 attributes (age, pitch, yaw) are continuous and it is more difficult to assess their modifications.
2. https://github.com/deepinsight/insightface
3. https://github.com/ageitgey/face_recognition

(a) PluGeN (b) StyleFlow

Fig. 5: Gradual modification of attributes (age, baldness, and yaw, respectively) performed on the StyleGAN latent codes.

As can be seen from Table 1, PluGeN performs the requested modifications more accurately and is less invasive in modifying the content of the original image than StyleFlow. High accuracy of modification and low discrepancy between input and output images suggest that PluGeN better disentangles the latent space of the backbone model and precisely alters the desired attribute. One can observe that StyleFlow obtains a better FID score than PluGeN. Since FID compares the distribution of generated and real images, creating images with uncommon combinations of attributes that do not appear in a training set may be scored lower, which partially explains the relation between accuracy and FID obtained by PluGeN and StyleFlow. In consequence, FID is not an adequate metric for measuring the quality of arbitrary image manipulations considered here, because it is too closely tied to the distribution of input images ([15] shows similar findings).

It is worth mentioning that PluGeN obtains these very good results using NICE model, which is the simplest type of INFs. In contrast, StyleFlow uses continuous INF, which is significantly more complex and requires using an ODE solver leading to unstable training. Moreover, to modify even a single attribute, StyleFlow needs to determine the values of all labels, since they represent the conditioning factors of INF. In consequence, every modification requires applying an auxiliary classifier to predict all image labels. The usage of PluGeN is significantly simpler, as subsequent coordinates in the latent space of INF correspond to the labeled attributes and they are automatically determined by PluGeN. Finally, our approach is less computationally expensive as we verified that, using the same hardware, PluGeN can be trained 3 times faster than StyleFlow and is around 100 times faster in inference.

4.2 PGGAN backbone

PluGeN is not restricted to the StyleGAN backbone and can be applied to other types of GAN models. In this experiment, we used PGGAN trained on images taken from the CelebAHQ dataset as a backbone model. PGGAN has a simpler architecture than StyleGAN and, in particular, does not contain the style space $\mathbf{W}$. In this case, PluGeN was directly attached to the latent space $\mathbf{Z}$, which has a prior Gaussian distribution. We use analogical NICE architecture for instantiated PluGeN.

TABLE 1: Attributes modification using StyleGAN backbone. High accuracy means the PluGeN correctly modifies the requested attributes while low values of FR MSE and AF MSE show that it preserves most features of the original image.

	Plugen			StyleFlow		
Requsted value	Acc ↑	FR ↓ MSE	AF ↓ MSE	Acc ↑	FR ↓ MSE	AF ↓ MSE
gender	**0.94**	**0.26**	**0.38**	0.91	0.35	0.57
glasses	**0.95**	**0.23**	**0.27**	0.85	0.33	0.50
bald	**0.77**	**0.32**	**0.53**	**0.77**	0.34	0.60
beard	**0.86**	**0.32**	**0.45**	0.83	0.40	0.65
smile	**0.98**	**0.20**	**0.19**	0.96	0.25	0.32
AVG	**0.90**	**0.27**	**0.36**	0.86	0.34	0.53
AVG FID	25.95			18.55		

PluGeN was trained on 100,000 images generated from PGGAN, which were labeled using an independent face attribute classifier. Due to licensing issues, we could not use the Microsoft Face API to label training images and evaluate the resulting model. For this reason, we trained the ResNet-18 model [34] on the FFHQ dataset. The model was trained with 8 attributes in a multi-label manner, treating the Microsoft Face API labels as targets.

As a baseline, we selected InterFaceGAN, which reports state-of-the-art performance for manipulating face attributes in the case of the PGGAN model. InterFaceGAN learns a disentangled representation after some linear transformations and proposes a technique for semantic face editing. Following the experimental setup of InterFaceGAN, we evaluated the modification of 5 face attributes. For evaluation, we used the accuracy of modification and MSE between the image embedding taken from the FR and AF models. For continuous attributes, such as age and yaw, we calculated the accuracy of exceeding a fixed threshold, e.g., age lower than 25 corresponds to negative labels, while values higher than 50 denote positive label.

The results presented in Table 2 show that PluGeN obtains at least 94% accuracy for manipulating 4 attributes. Although the score for modifying the age attribute is lower than 50%, it is still higher than that achieved by InterFaceGAN. The highest discrepancy between PluGeN and InterFaceGAN is observed for the smile attribute. A significantly higher score of PluGeN shows that the smile attribute

TABLE 2: Attributes modification using PGGAN backbone.

	PluGeN			InterFaceGAN		
	Acc ↑	FR ↓ MSE	AF ↓ MSE	Acc ↑	FR ↓ MSE	AF ↓ MSE
gender	0.95	**0.54**	**0.92**	**0.98**	0.55	**0.92**
glasses	0.94	**0.51**	**0.85**	**0.99**	0.59	1.04
age	**0.48**	**0.57**	**1.00**	0.46	0.60	1.08
smile	**0.99**	**0.33**	0.41	0.77	**0.33**	**0.37**
yaw	0.98	0.41	0.57	**1.00**	**0.40**	**0.53**
AVG	**0.87**	**0.47**	**0.75**	0.84	0.49	0.79

TABLE 3: Attributes disentanglement measured by the accuracy (higher is better). For each image, we change of the values of attributes listed in rows and verify whether the remaining attributes (listed in columns) stay unchanged.

	gender	glasses	age	smile	yaw	AVG
	PluGeN					
gender	-	**0.91**	0.80	**0.85**	0.85	**0.87**
glasses	**0.85**	-	0.83	**0.84**	0.84	**0.86**
age	**0.40**	**0.45**	-	**0.41**	0.42	**0.43**
smile	**0.93**	**0.96**	**0.91**	-	**0.89**	**0.94**
yaw	0.90	0.96	0.90	0.87	-	0.92
AVG	**0.81**	**0.85**	0.78	**0.79**	**0.80**	**0.80**
	InterFaceGAN					
gender	-	**0.91**	**0.90**	0.66	**0.88**	0.86
glasses	0.63	-	**0.90**	0.82	**0.89**	0.85
age	0.31	0.44	-	0.30	**0.44**	0.39
smile	0.70	0.76	0.73	-	0.73	0.74
yaw	**0.93**	**0.98**	**0.93**	**0.90**	-	**0.95**
AVG	0.71	0.82	**0.79**	0.69	0.79	0.76

cannot be fully controlled using the linear operation applied by InterFaceGAN and the nonlinear model implemented by PluGeN is necessary. MSE values delivered by the FR and AF models confirm that PluGeN is less invasive to the image content than InterFaceGAN. These effects are illustrated in Figure 6.

Additionally, we verified the disentanglement between labeled attributes in a strict quantitative way. That is, we forced the change of a single attribute and verified whether the values of other labeled attributes changed as well. Ideally, the values of the remaining attributes should remain intact. We applied a standard accuracy measure, which tests whether the classifier keeps its original prediction on non-modified attributes. The results presented in Table 3 confirm that PluGeN provides better disentanglement than InterFaceGAN. In consequence, the proposed non-linear mapping works better than the linear transformation applied by InterFaceGAN.

4.3 Image manipulation on VAE backbone

In the following experiment, we show that PluGeN can be combined with autoencoder models to effectively manipulate image attributes. We use CelebA database, where every image of the size 256×256 is annotated with 40 binary labels.

We compare PluGeN to MSP [7], a strong baseline, which uses a specific loss for disentangling the latent space of VAE. Moreover, we also use flow plug-in network (FPN) [6], which analogically to StyleFlow uses a conditional INF to introduce labeled attributes. The architecture of the base VAE and the evaluation protocol were taken from the original MSP paper. More precisely, for every input image, we manipulate the values of two attributes (we inspect 20 combinations in total). The success of the requested manipulation is verified using a multi-label ResNet-56 classifier trained on the original CelebA dataset.

TABLE 4: Accuracy and FID scores of image manipulation performed on the VAE backbone.

Requested value	PluGeN	MSP	FPN
male x beard	0.80	0.79	**0.85**
female x beard	**0.59**	0.33	0.31
male x no-beard	0.88	**0.92**	0.91
female x no-beard	0.85	0.82	**0.95**
male x makeup	**0.44**	0.43	0.29
male x no-makeup	0.72	0.92	**0.96**
female x makeup	0.42	0.41	**0.58**
female x no-makeup	0.55	0.40	**0.85**
smile x open-mouth	0.97	**0.99**	0.79
no-smile x open-mouth	0.79	**0.82**	0.77
smile x calm-mouth	0.84	**0.91**	0.72
no-smile x calm-mouth	0.96	0.97	**0.99**
male x bald	0.26	**0.41**	0.34
male x bangs	0.58	**0.74**	0.45
female x bald	0.19	0.13	**0.39**
female x bangs	0.52	0.49	**0.60**
no-glasses x black-hair	0.92	**0.93**	0.74
no-glasses x golden-hair	**0.92**	0.91	0.81
glasses x black-hair	0.76	**0.90**	0.58
glasses x golden-hair	0.75	**0.85**	0.61
Average Acc	0.69	**0.70**	0.67
Average FID	**28.07**	30.67	39.68

The sample results presented in Figure 7 demonstrate that PluGeN attached to VAE produces high-quality images satisfying the constraints imposed on the labeled attributes. The quantitative comparison shown in Table 4 confirms that PluGeN is extremely efficient in creating uncommon combinations of attributes, while FPN performs well only for the usual combinations. At the same time, the quality of images produced by PluGeN and MSP is better than in the case of FPN. Although both PluGeN and MSP focus on disentangling the latent space of the base model, MSP has to be trained jointly with the base VAE model and it was designed only to autoencoder models. In contrast, PluGeN is a separate module, which can be attached to arbitrary pre-trained models. Due to the use of invertible neural networks, it preserves the reconstruction quality of the base model, while adding manipulation functionalities. In the following experiment, we show that PluGeN also performs well at generating entirely new images, which is not possible using MSP.

4.4 Image generation with VAE backbone

In addition to manipulating the labeled attributes of existing images, PluGeN generates new examples with desired attributes. To verify this property, we use the same VAE architecture as before trained on CelebA dataset and compare our results with FPN. We exclude MSP from the comparison because it cannot generate new images, but only manipulate the attributes of existing ones (see supplementary materials for a detailed explanation).

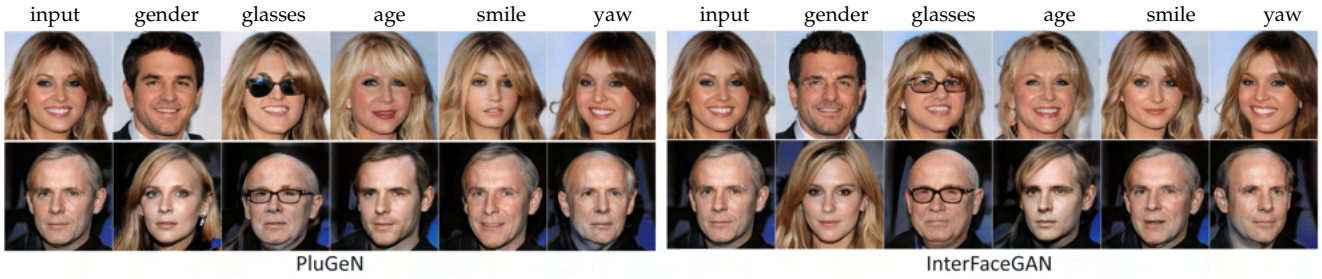


Fig. 6: Attributes manipulation performed using the PGAGAN backbone.

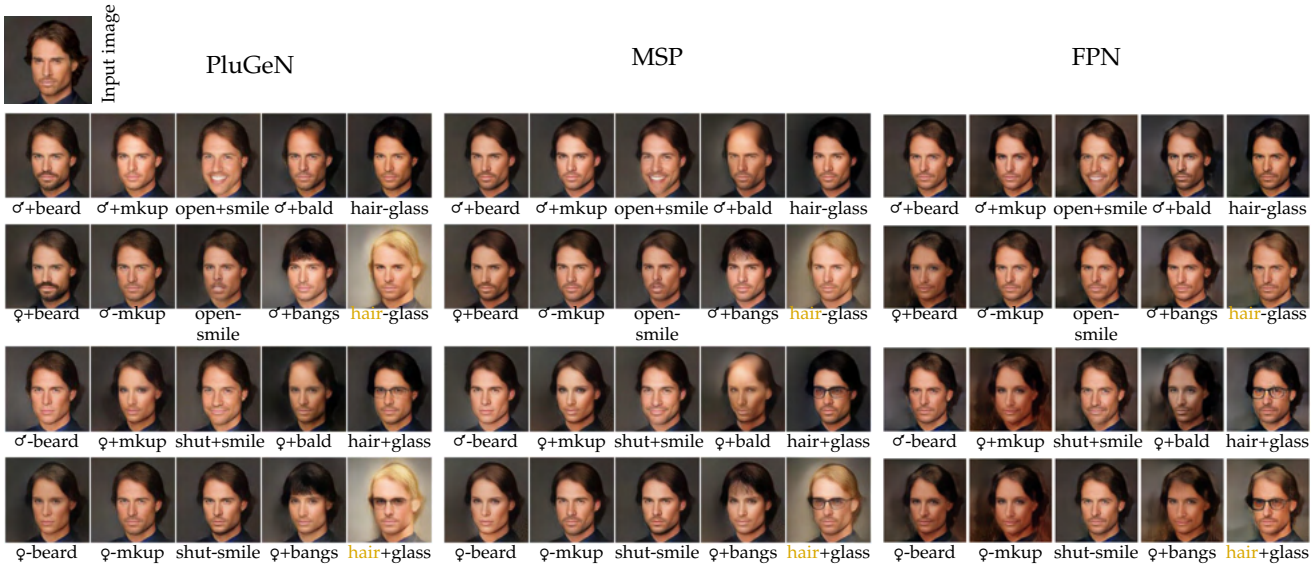


Fig. 7: Examples of image attribute manipulation using VAE backbone.

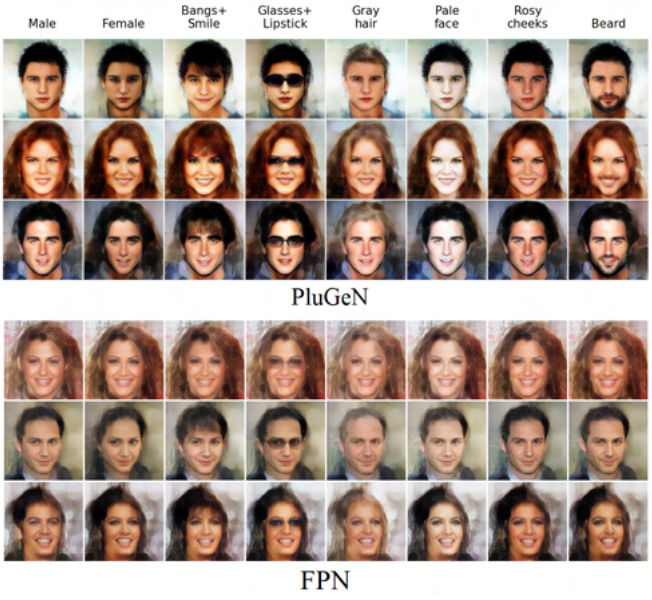


Fig. 8: Examples of conditional generation using VAE backbone. Each row contains the same person (style variables) with modified attributes (label variables).

Figure 8 presents sample results of image generation with the specific conditions. In each row, we fix the style variables $\mathbf{s}$ and vary the label variables $\mathbf{c}$ in each column, generating the same person but with different characteristics such as hair color, eyeglasses, etc. PluGeN and FPN generate images of similar quality, but only PluGeN is able to correctly manipulate the labeled attributes. The lower quality of generated images is caused by the poor generation abilities of VAE backbone, which does not work well with high dimensional images (see supplementary materials for a discussion). For this reason, it is especially notable that PluGeN can improve the generation performance of the backbone model in contrast to MSP.

4.5 Disentangling the attributes

The attributes in the CelebA dataset are strongly correlated and at times even contradictory, e.g. attributes 'bald' and 'blond hair' cannot both be present at the same time. In this challenging task, we aim to disentangle the attribute space as much as it is possible to allow for generating examples with arbitrary combinations of attributes. For this purpose, we sample the conditional variables c_i independently, effectively ignoring the underlying correlations of attributes, and use them to generate images. Since the attributes in the CelebA dataset are often imbalanced (e.g. only in 6.5% of

TABLE 5: Results of the independent conditional generation using VAE backbone.

	PluGeN	FPN
F1	**0.44**	0.29
AUC	**0.76**	0.68

examples the person wears glasses), we calculate F1 and AUC scores for each attribute.

The quantitative analysis of the generated images presented in Table 5 confirms that PluGeN outperforms the baseline FPN with respect to classification scores. The overall metrics are quite low for both approaches, which is due to the difficulty of disentanglement mentioned above, as well as the inaccuracy of the ResNet attribute classifier. Deep learning models often fail when the correlations in the training data are broken, e.g. the classifier might use the presence of a beard to predict gender, thus introducing noise in the evaluation [35].

5 Experiments on partially labeled images

In this part, we evaluate the effects of training PluGeN on partially labeled data.

5.1 Comparing semi-supervised strategies on CelebA

First, we compare the proposed two strategies (KDE and uniform) of estimating the total probability (from Section 3.4) to enable a semi-supervised training of PluGeN on partially labeled data. We use CelebA dataset and VAE backbone. To generate a partially labeled dataset, we randomly select a subset of labels for every image, which are available for training. The remaining labels are kept hidden. We consider the case of 20, 10, 5 and 1 labels per image (out of 40 labels). The last situation of 1 label corresponds to using only 2.5% of all labels.

As shown in Figure 9a both strategies give very good results when at least 5 labels per image (12.5%) are available. When 20 or 10 labels are given, then the performance is very similar to the fully-labeled case, while for 5 labels we observe 2 percentage point decrease in the accuracy. Given only 1 label per image, both models deteriorate their results and obtain the accuracy slightly above 50%. In consequence, PluGeN reduces the need for the labeled data being able to maintain its performance using only 12.5% of all labels.

More detailed inspection of Figure 9a reveals that KDE estimate works slightly better when more labels are available, while the uniform density gives higher accuracy for the case of 1 label. Given many labels, KDE produces reliable density estimate and thus it is useful for inferring the values of unknown labels. On the other hand, KDE can marginalize the role of minor values (or even ignore them completely), which might explain its lower accuracy in the case of 1 label per image. Applying uniform distribution does not introduce any bias to the labeling process and treats all values equally.

5.2 Evaluation on the StyleGAN backbone

In this experiment, we focus on comparing the performance of semi-supervised version of PluGeN with StyleFlow. We follow the setup introduced in Section 4.1 and train all models on partially labeled FFHQ dataset using StyleGAN backbone.

To generate partially labeled dataset, we consider two strategies for labels removal. In the first one (refer to as *partial*), we randomly remove a fixed percentage of labels for every image. In consequence, every example contains the same number of labels, but not all labels are visible. This strategy is analogical to the one considered in Section 5.1. In the second case (refer to as *split*), we keep all labels for a fixed percentage of examples while the labels of remaining images are removed. Thus we divide the set of images into two subsets: fully-labeled and unlabeled. Observe that unlabeled data can be still used for training PluGeN.

Following results from the previous section, we use the uniform strategy for estimating the total probability in PluGeN, given its simplicity. Since StyleFlow cannot be trained on partially labeled dataset, we decided to train it only on the set of fully labeled examples. In consequence, for StyleFlow, we only considered the case where a fixed percentage of data contains all labels while the remaining ones are completely removed from training. For completeness, we also consider the same setup for PluGeN referred to as *labeled*.

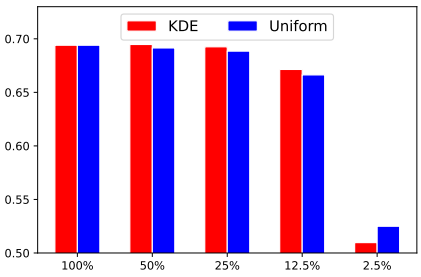
Figure 9b presents the accuracy of modification discrete attributes using 50% (4 labels), 25% (2 labels), and 12.5% (1 label) of labels, respectively. It is evident that the performance of PluGeN is stable even when only 1 of 8 labels is available for training. We illustrate the manipulation effects with sample images in Figure 10. Observe that StyleFlow, which was trained only on fully labeled examples, keeps its performance only on 50% of the data. For 25% of the data, it significantly decreases its accuracy, while for the level of 12.5%, it completely fails to generate face images during modification. We verify that PluGeN also decreases its performance when the unlabeled data are completely removed from the training, but the degradation is not as high as in the case of StyleFlow.

We also verified how much of the image content was changed when manipulating a given attribute. Figures 9c and 9d show that the MSE values for all versions of PluGeN behave very stable. On the contrary, the MSE is considerably higher for StyleFlow than for PluGeN when all labels are available. Furthermore, the MSE for StyleFlow decreases as the number of labels is lower, reaching a very low level for 12.5% of labels. Further inspection revealed that for a smaller number of labels, StyleFlow was completely unable to perform most of the manipulations (resulting in low accuracy) and returned the input image unchanged.

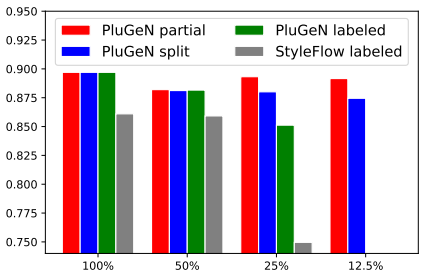
In conclusion, PluGeN is capable of correctly manipulating face images even in challenging settings where it has to leverage unlabeled or partially labeled data, making it better suited to a wide range of real-world problems.

5.3 Biased partial labeling

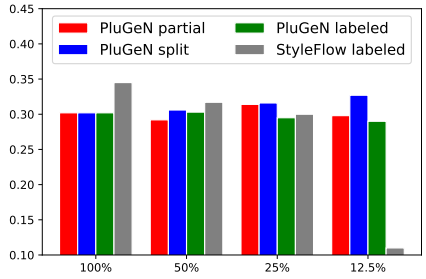
To further analyze the training of PluGeN on partially labeled data, we consider the case of biased labeling. In other words, instead of randomly removing a portion of labels, we delete more labels with a given value, e.g. remove more people with glasses than without. We verify whether PluGeN is robust to the biased partial labeling process.



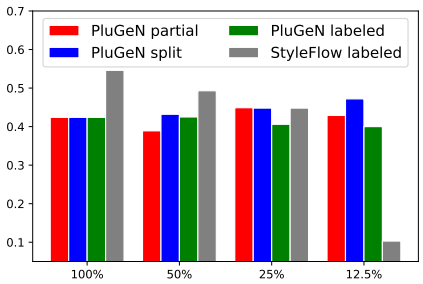
(a) Accuracy of two strategies to include partial labels in the VAE backbone and the CelebA dataset.



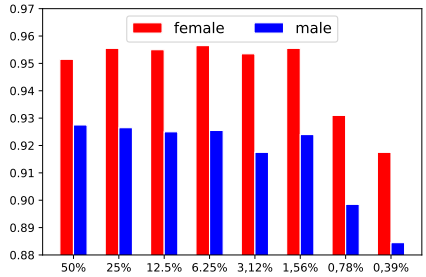
(b) Accuracy on partially labeled FFHQ dataset using StyleGAN backbone.



(c) FR MSE on partially labeled FFHQ dataset using StyleGAN backbone.



(d) AF MSE on partially labeled FFHQ dataset using StyleGAN backbone.



(e) Accuracy on partially labeled FFHQ dataset when the number of female label is decreasing.

Fig. 9: Evaluation of attributes modification in a semi-supervised setting.

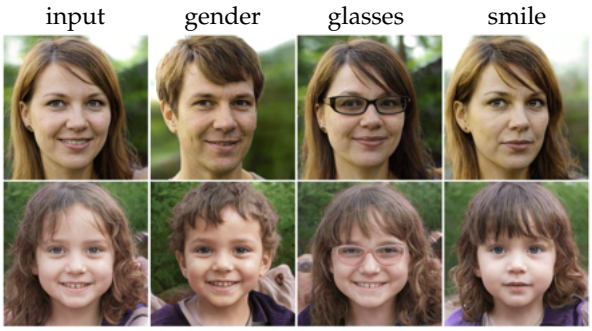


Fig. 10: Manipulation of binary attributes performed by the semi-supervised version of PluGeN on the StyleGAN codes.

We first randomly remove 75% of labels (6 labels) for every image in the FFHQ data set. Next, we only change the distribution of the label describing the gender and gradually remove labels with value "female". Since we initially removed 75% of all labels, then the model cannot easily deduce that hidden values of label gender denotes the females. Otherwise, it would be a trivial task.

Figure 9e shows that PluGeN is extremely robust to biased labeling. Performance remains stable until the level of 6.25%. Below this value, the accuracy starts to slowly decrease, but it is still above 90% when only 0.4% of the female label is visible.

6 Experiments on non-image data

In this section, we present two practical use-cases of using PluGeN on chemical molecule generation using CharVAE and 3D point cloud modeling using PointFlow.

6.1 Chemical molecules modeling

In this part, we apply PluGeN to generate chemical molecules with the requested properties. As a backbone model, we use CharVAE [2], which is a type of recurrent network used for processing SMILES [36], a textual representation of molecules. It was trained on ZINC 250k database [37] of commercially available chemical compounds. For every molecule, we model 3 physio-chemical continuous (not binary) labels: logP, SAS, TPSA, which values were calculated using RDKit package [38]. Additional explanations and more examples are given in the supplementary materials.

First, we imitate a practical task of de novo design [39], [40], where we force the model to generate new compounds with desirable properties. For every attribute, we generate 25k molecules with 3 different values: for logP we set the label of generated molecules to: 1.5, 3.0, 4.5; for TPSA we set generated labels to: 40, 60, 80; for SAS we set them to: 2.0, 3.0, 4.0, which gives 9 scenarios in total. From density plots of labels of generated and original molecules presented in Figure 11, we can see that PluGeN changes the distribution of values of the attributes and moves it towards the desired value. A slight discrepancy between desired and generated values may follow from the fact that values of labeled attributes were sampled independently, which could make some combinations physically contradictory.

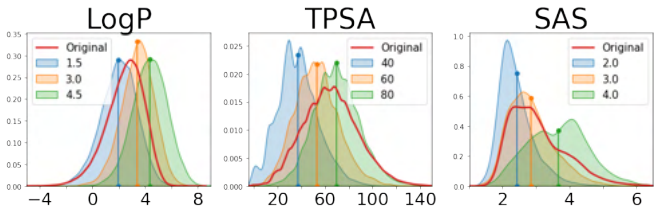
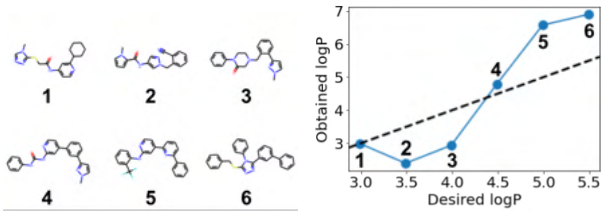


Fig. 11: Distribution of attributes of generated molecules, together with distribution for the training dataset. Each color shows the value of a labeled attribute that was used for generation. PluGeN is capable of moving the density of generated molecules' attributes towards the desired value. The average of every distribution is marked with a vertical line.



(a) Molecules decoded from path (b) LogP of presented molecules

Fig. 12: Molecules obtained by the model during an optimization phase (left side), and their LogP (right side).

Next, we consider the setting of lead optimization [41], [42], where selected compounds are improved to meet certain criteria. For this purpose, we encode a molecule into the latent representation of INF and force PluGeN to gradually increase the value of logP by 3 and decode the resulting molecules. The obtained molecules together with their logP are shown in Figure 12. As can be seen, PluGeN generates molecules that are structurally similar to the initial one, however with optimized desired attributes.

Obtained results show that PluGeN is able to model the physio-chemical molecular features, which is a non-trivial task that could speed up a long and expensive process of designing new drugs.

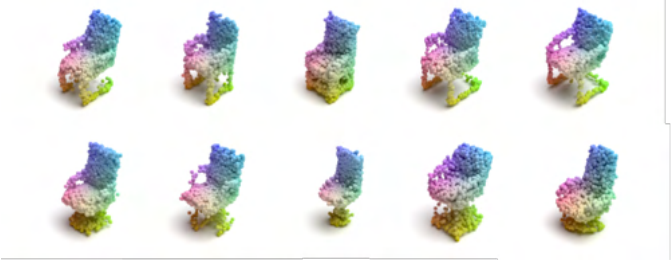


Fig. 13: The samples generated from PluGeN assuming particular attributes. In the first row, we generate cantilever arm chairs, and in the second row, swivel arm chairs.

6.2 3D point clouds modelling

In this experiment, we investigate the quality of PluGeN in application to conditional point cloud generation. We

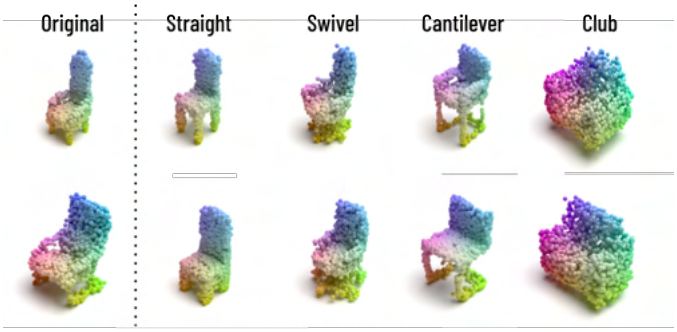


Fig. 14: The results for attribute manipulation using PluGeN applied to point clouds. We modify the original point cloud from the test split by switching the particular attributes responsible for the chair type.

use a standard benchmark dataset named ShapeNet [43], and we focus on the *chair* class. For this particular class, we distinguish 7 attributes that can describe chairs: *arms*, *straight*, *club*, *cantilever*, *swivel*, *folding*, and *rocking*. As pre-trained frozen models, we use PointFlow [44] and LION [45], both trained directly on train split for chair class. For PointFlow model, we postulate to use a pre-trained CNF as a PluGeN component because this type of normalizing flow was used to enrich the prior distribution over the latent space of the PointFlow. Therefore, during training PluGeN, we initialize the model with pre-trained weights and freeze the entire PointFlow, except the flow component plugged-in to latent, which is trained according to the PluGeN paradigm using the defined attributes for *chairs*. For the LION model, we train CNF from scratch over the shape latent space following PluGeN training methodology. For both backbones, we evaluate the quality of PluGeN and compare the results with Flow Plugin Network (FPN) [6], which uses additional embedding and a conditional variant of the flow to model conditional distribution.

In Figure 13, we provide qualitative results for conditional generation using the PluGeN model and PointFlow backbone. In Figure 14, we present the results for point cloud attribute manipulation, where the original point cloud is modified by switching the value of the single attribute representing the particular chair shape. In the first row, we provide the samples for the attributes *arms* and *cantilever* switched on, and in the second row, we used *arms* and *swivel* for conditional generation. The results of conditional generation and modification for FPN are presented in the supplementary materials.

In Table 6, we provide the results of quantitative analysis of the conditional generative capabilities of both models for PointFlow and LION backbones. We analyze the generative capabilities in the conditional framework using the test split of *chair* subset of ShapeNet. We use a conditional model to generate the same number of samples as in the test set, preserving the same distribution of attributes. In order to evaluate the generative capabilities of the model, we use standard metrics used for point clouds, including Minimum Matching Distance (MMD), Coverage (COV), 1-nearest neighbor accuracy (1-NN) (each of them calculated using both Chamfer Distance (CD), Earth-Mover Distance

TABLE 6: Results of conditional point cloud generation using Flow Plugin Network (FPN) and PluGeN (ours) for PointFlow and LION backbones. **Bold** results are the best results for the specific backbone, whereas underlined results are the best results overall.

	CLASS-CD	CLASS-EMD	MMD-CD	MMD-EMD	COV-CD	COV-EMD	1-NN-CD	1-NN-EMD	JSD
PointFlow Backbone									
FPN	0.6562	0.5769	0.0019	0.0731	**0.5471**	**0.5421**	0.5793	**0.5471**	**0.0193**
PluGeN	**0.7174**	**0.6496**	**0.0016**	**0.0682**	0.5322	0.5240	**0.5719**	0.5760	0.0301
LION Backbone									
FPN	0.5388	0.4628	0.0027	0.0178	0.3785	0.4066	0.6562	0.7256	0.0726
PluGeN	**0.5917**	**0.5140**	**0.0023**	**0.0151**	**0.4496**	**0.4727**	**0.5992**	**0.6397**	**0.0324**

(EMD)) and Jensen-Shannon Divergence (JSD). The definitions of the selected measures are provided in [44]. In order to evaluate the quality of conditioning in the process of generating the samples, we use classification accuracy (CLASS), which also uses both of the considered distance measures. To calculate the accuracy, we take each generated sample, find the closest neighbor in the test set according to the considered distance measure, and check if the vectors of attributes for the test example and generated sample are the same. The number of exact matches is further divided by the number of test cases, and the final metric is calculated. We can observe that generative capabilities on the PointFlow backbone for both of the considered models are comparable. While evaluating the ability of the model to create the samples with a given vector of attributes measured by CLASS-CD and CLASS-EMD, the PluGeN beats reference FPN by a large margin. Moreover, for the LION backbone, PluGeN is better than FPN for all considered metrics. Finally, we can observe that results for the PointFlow backbone for both methods are usually better than for the LION backbone.

7 Conclusion

We proposed a novel approach for disentangling the latent space of pre-trained generative models, which works perfectly for generating new samples with desired conditions as well as for manipulating the attributes of existing examples. In contrast to previous works, we demonstrated that PluGeN performs well across diverse domains, including chemical molecule modeling and 3D point clouds generation, and can be combined with various architectures, such as GANs and VAEs backbones. Moreover, it can be trained in a semi-supervised manner, which further increases its applicability in practical use-cases.

Appendix A
Parametrization of PluGeN

Modeling imbalanced binary labels. In many cases, the class labels are imbalanced, which means that the number of examples from one class significantly exceeds the other class (e.g., only 6.5% examples in CelebA dataset have the 'glasses' label). To deal with imbalanced data, we scale the variance of Gaussian density modeling the conditional distribution $p_{C_i|Y_i=y_i}$.

We consider the conditional density of i-th attribute represented by:

$$p_{C_i|Y_i=y_i} = \mathcal{N}(m_0, \sigma_0)^{(1-y_i)} \cdot \mathcal{N}(m_1, \sigma_1)^{y_i}, \tag{7}$$

where $m_0 = -1$ and $m_1 = 1$. We assume that p_0, p_1 are the fractions of examples with class 0 and 1, respectively. To deal with imbalanced classes we put

$$\sigma_i = \sigma\sqrt{2p_i},$$

where $\sigma > 0$ is a fixed parameter. For a majority class, standard deviation becomes higher, which introduces a lower penalty in the case of negative log-likelihood loss. The minority class has a higher penalty because we need to stop the mixture from collapsing into a single component.

Let us calculate the log-likelihood of our conditional prior density $p_{C_i|Y_i=y_i}$ using the parametrization $\sigma_i = \sigma\sqrt{2p_i}$. We have

$$-\log p_{C_i|Y_i=y_i}(c) = y_i \cdot \lambda_1 \frac{(c-m_1)^2}{2\sigma^2} + (1-y_i) \cdot \lambda_0 \frac{(c-m_0)^2}{2\sigma^2} + const, \tag{8}$$

where $\lambda_i = \frac{1}{2p_i}$ is an extra weighting factor.

We observe that, for our selection of σ_i, the expected value of the weighting factors with respect to labeling variable y equals 1. In consequence,

$$\mathbb{E}_y[-\log p_{C_i|Y_i=y_i}(c)] = p_0\frac{(c-m_1)^2}{2\sigma^2} + p_1\frac{(c-m_0)^2}{2\sigma^2} + const,$$

which is a typical log-likelihood of Gaussian distribution assuming class proportion p_i.

Reducing σ in a training of PluGeN. Here, we describe the schedule for parameter σ used for modeling conditional distribution $\mathcal{N}(m, \sigma)$. We want to ensure the flexibility of the INF at the beginning of the training, but we also need the attribute values to be strictly separated. In order to achieve both of these conditions, we impose a schedule on the standard deviation σ. Starting with high σ we allow for great flexibility of our model, and then we get class separation by reducing the value of σ. Namely, we use the following schedule for the standard deviation σ of the class normal distributions:

$$\sigma(t) = \sigma_0 \cdot \gamma^t,$$

where t is the index of the current epoch and σ_0, γ are hyperparameters setting, respectively, the starting point and the speed of value decay. The selection process of σ_0 and γ is described in the following sections.

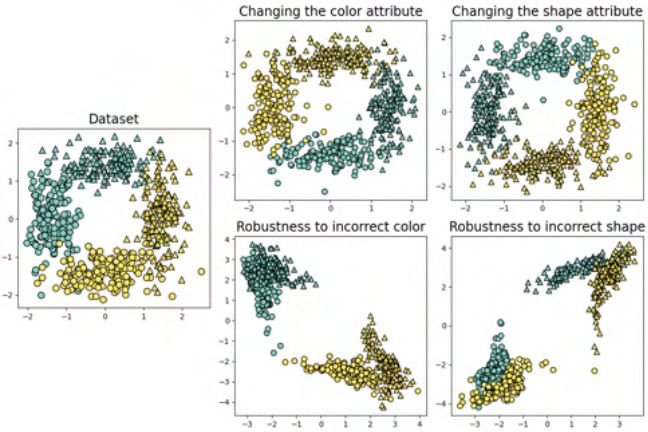


Fig. 15: Additional experiments on the 2D dataset with the baseline conditional INF model. On the left we show the training dataset. In the top row we show how the dataset is reconstructed if we flip the attributes of all examples. In the bottom row, we show what happens if the examples are encoded with one of the attributes flipped incorrectly and then decoded with the correct attribute.

Additional analysis of conditional INF. In this paragraph, we present additional experiments with the two-dimensional toy dataset. For these experiments we use a simple NICE flow with 4 layers and 4 blocks per layer, each of dimenstionality 64. Here, we train the baseline model on the whole dataset (including the previously missing combination of attributes) and check how it performs at the attribute manipulation task. Results presented in Figure 15 show that the model is able to correctly perform attribute manipulation, but if we encode examples with incorrect labels and then decode with the correct one, it tends to extrapolate outside the dataset.

APPENDIX B
DETAILS OF IMAGE EXPERIMENTS

All experiments were run on a single NVIDIA DGX Station with Ubuntu 20.04 LTS. The full hardware specification includes 8 Tesla V100 GPUs with 32GB VRAM, 512GB RAM, and Intel(R) Xeon(R) CPU E5-2698 v4. Each experiment was run using a single GPU. The code is based on the PyTorch [46] framework.

B.1 Architectures of the models

StyleGAN backbone. Our experiments were performed using the pre-trained, publicly available StyleGAN2 trained on the FFHQ dataset [47].

PGGAN backbone. For our experiments, we used the PGGAN backbone trained on the Celeba-HQ 1024 dataset provided in the Genforce repository: https://github.com/genforce/genforce.

PluGeN for PGGAN and StyleGAN backbones. We use NICE architecture with 4 coupling layers with 4 layers in each and width 512. We use Adam optimizer with learning rate 10^{-4} and train model for 1000 epochs. The hyperparameters σ_0 and γ used for modeling conditional distributions, are set to 0.4 and 0.999, respectively.

VAE backbone. For our experiments, we reuse the VAE architecture from [7]. We use an encoder with 5 convolutional layers starting with 128 filters and doubling. The decoder is symmetrical to the decoder. We use leakyReLU activations. We train the network for 50 epochs with batch size 40 and Adam optimizer with the learning rate set to 10^{-4}. We additionally train a PatchGAN model [48] to improve the sharpness of the images.

PluGeN for VAE backbone. As previously, we use NICE architecture with 4 coupling layers with 4 layers in each and width 256. We train the model for 50 epochs using Adam optimizer with learning rate 10^{-4} and σ_0. The hyperparameters γ are set to 0.7 and 0.99, respectively.

FPN. We train FPN model also on top of the base network. We use Conditional Masked Autoregressive Flow with 5 layers of each consisting of reverse permutation and MADE component with 2 residual blocks. Moreover, we have been encoding attributes using 1 linear layer which was after that passed as a context input to the flow. We train the model for 50 epochs using Adam optimizer witht learning rate 10^{-3}. During sampling, the temperature trick was used with $T = 0.7$.

ResNet classifier. To evaluate the correctness of attribute manipulation in the case of CelebA dataset, we used a standard ResNet-56 classifier. We trained it on the task of multi-label classification, with class weighting to correct for class imbalance. We used the Adam optimizer with the learning rate set to $3 \cdot 10^{-4}$, batch size 256 and trained it for 50 epochs.

Attribute classifier for FFHQ dataset To evaluate the proposed an independent face attribute classifier was constructed. We train the ResNet-18 model on the FFHQ and 10 000 randomly generated StyleGAN face images. The model is trained with 8 outputs in a multi-label manner, treating the Microsoft Face API labels as targets. We standardize the labels as well as apply the shrinkage loss as we find that it helps with dataset imbalance. We use the same loss for binary and continuous labels as this works equally well for classification.

B.2 Additional Results

In this subsection, we present additional results and models comparison, which were not included in the main paper because of space restrictions.

Manipulating the StyleGAN latent codes. In Figures 16 and 17, we present additional results of attribute manipulations performed by PluGeN and StyleFlow on the latent codes of StyleGAN backbone. In most cases, PluGeN modifies only the requested attribute leaving the remaining ones unchanged, which is not always the case of StyleFlow (compare 4th row of Figure 16 or 3rd row of Figure 17). This confirms that the latent space produced by PluGeN is more disentangled than the one created by StyleFlow.

Manipulating images using VAE backbone. In Figure 18, we show additional results of image manipulation performed by PluGeN, MSP, and FPN using VAE backbone. One can observe that PluGeN and MSP perform the requested modification more accurately than FPN.

Manipulating attributes intensity of generated images. In this experiment, we consider images fully generated by

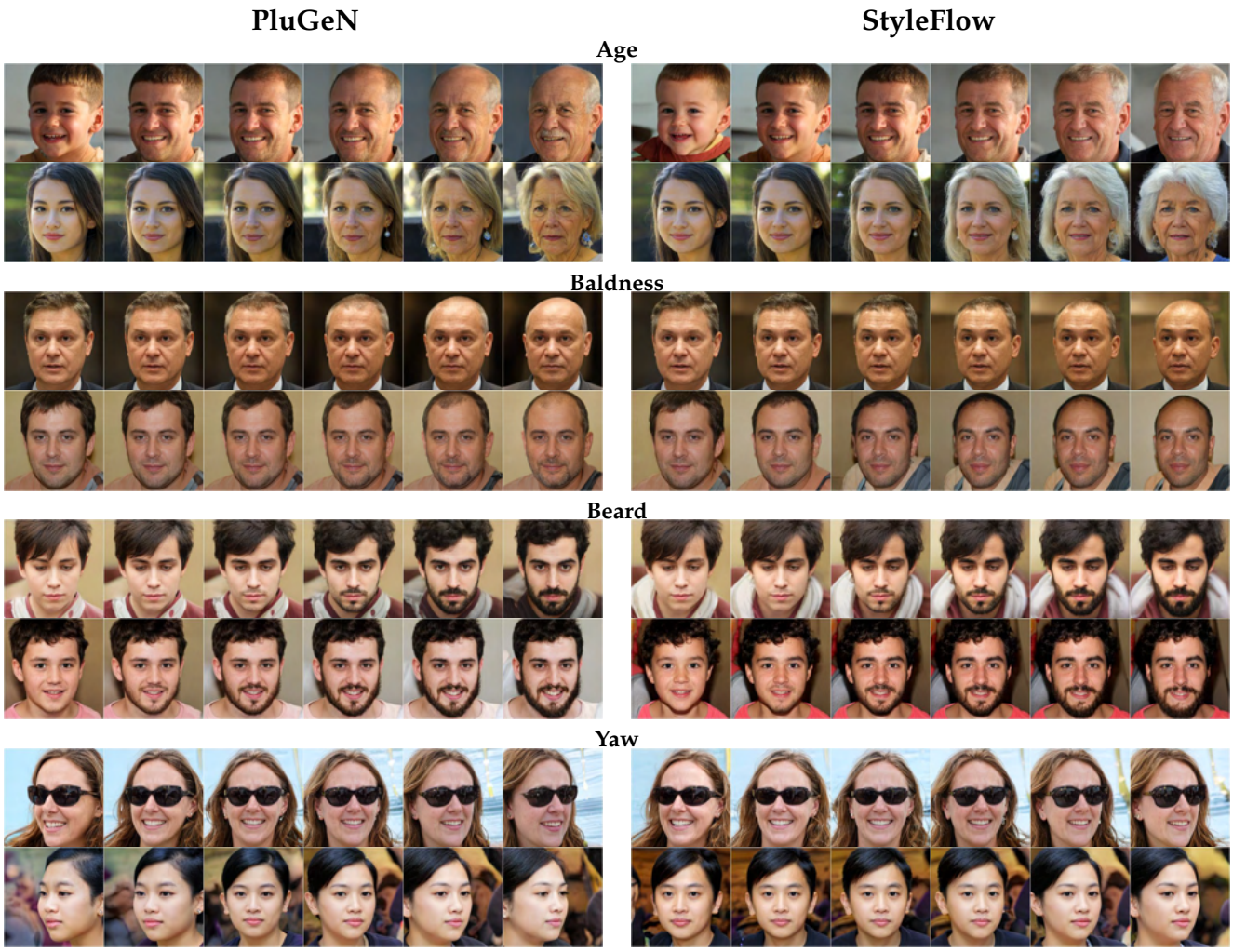


Fig. 16: Gradual modification of attributes (age, baldness, beard, and yaw, respectively) performed by PluGeN (left) and StyleFlow (right) using the StyleGAN backbone.

PluGeN (not reconstructed images) attached to the VAE backbone. More precisely, we sample a single style variable $\mathbf{s}$ from the prior distribution and manipulate the label variables of CelebA attributes. It is evident in Figure 21 that PluGeN freely interpolates between binary values of each attribute and even extrapolates outside the data distribution. This is possible thanks to the continuous form of prior distribution we are using in the latent space, which enables us to choose the intensity of each attribute. We emphasize that this information is not encoded in the dataset, where labels are represented as binary variables. However, in reality, an attribute such as 'narrow eyes' covers a whole spectrum of possible eyelid positions, from eyes fully closed, through half-closed to wide open. PluGeN is able to recover this property without explicit supervision. Interestingly, we also see cases of extrapolation outside of the dataset, e.g. setting a significantly negative value of the 'bangs' attribute, which can be interpreted as an illogical condition 'extreme absence of bangs', creates a white spot on the forehead.

Figure 22 shows that the shape of the empirical distributions in the latent space of PluGeN allows for this continuous change. While the positive and negative classes of boolean attributes such as the presence of a hat or eyeglasses are clearly separated, in more continuous variables like youth and attractiveness they overlap significantly, allowing for smooth interpolations. This phenomenon emerges naturally, even though CelebA provides only binary labels for all the attributes.

Generation capabilities of MSP and the VAE backbone. In Figure 23 (top), we demonstrate that the base VAE model taken from the MSP paper [7] cannot generate new face images, but only manipulate the attributes of input examples. In consequence, it works similar to the autoencoder model. For this reason it is especially notable that PluGeN can improve the generation performance of the backbone model (see the main paper). In contrast, MSP cannot generate new face images using this VAE model as shown in the bottom row of Figure 23. For very low temperatures, MSP generates typical (not diverse) faces.

Generating images with attributes combinations taken from test set. We present additional quantitative results for generating images with the requested combinations of attributes. In this experiment, we focus on typical combinations, which appear in a dataset. For this purpose, we gen-

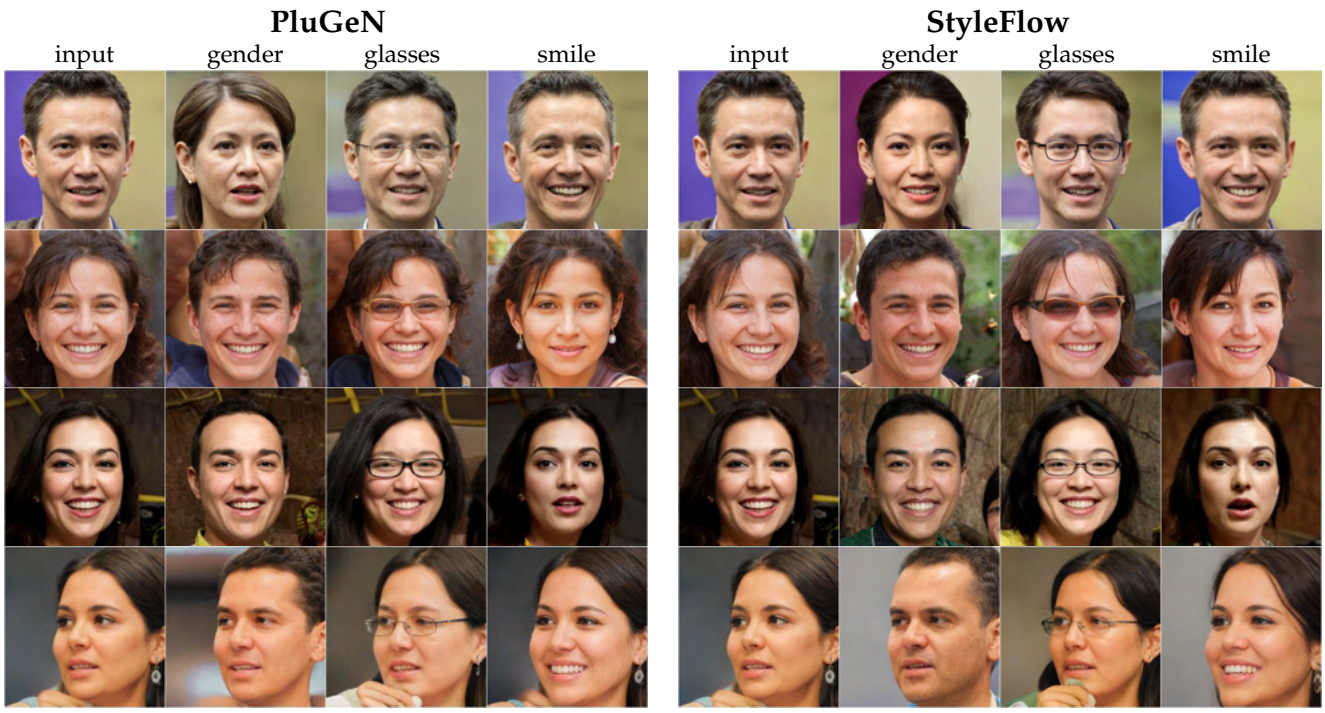


Fig. 17: Attributes manipulation performed by PluGeN (left) and StyleFlow (right) using the StyleGAN backbone.

TABLE 7: Average classification metrics for generating images with the combinations of attributes taken from the test set of CelebA.

	PluGeN	FPN
F1	**0.69**	0.49
AUC	**0.92**	0.85

erate 20,000 images with the same attribute combinations as in the CelebA test set. The results presented in Table 7 show that PluGeN outperforms both FPN, cVAE, and Δ-GAN in terms of classification scores.

Semi-supervised generation. In Figure 24 we show the effects of continuous attributes manipulation performed on sample images by semi-supervised version of PluGeN.

TABLE 8: Average classification metrics and FIDs for generating images with the combinations of attributes taken from the test set of CelebA.

	NICE	CNF
FID	72.72	68.96
F1	0.69	0.63
AUC	0.92	0.89

TABLE 9: Average classification metrics and FIDs for generating images, when the values of attributes were sampled independently.

	NICE	CNF
FID	77.48	73.31
F1	0.44	0.41
AUC	0.78	0.75

B.3 Ablations

CNF vs NICE. In our main experiments, we use the NICE [30] approach to flow-based models. This choice was motivated by the computational and conceptual simplicity of the approach. However, we also empirically evaluated a more complex approach of continuous normalizing flows [49] which cast the distribution modeling task as a problem of solving differential equations. The CNF implementation consisted of 2 stacked CNFs, each containing 3 concat-squash layers with a hidden dimension 2048. Table 8 shows the results of both approaches in the task of multi-label conditional generation using VAE backbone. We use the same combinations of attributes as in the CelebA test set. Table 9 shows an analogical comparison when the attributes were sampled independently, which is more challenging setting. For both of these settings results on NICE and CNF are comparable. Although CNF samples get better FIDs, they also score worse on the classification metrics, which suggests that the model might be worse at enforcing the class conditions. Overall, both models perform similarly and because of that, we use NICE as the approach is less expensive computationally.

Different autoencoder backbones. In order to investigate how the structure of the latent space of the backbone autoencoder impacts the performance of our model, we check multiple β-VAE models with varying values of β. For each model we trained three architectures of INFs (small, medium, big) and picked the best performing ones for evaluation. The results presented in Table 10 show that the FID scores get worse as the value of β increases. This is caused by the drop in the reconstructive power of the base model, which focuses more on the latent space regularization instead. Interestingly, the statistics also fall as the value

Fig. 18: Examples of image attribute manipulation using VAE backbone.

TABLE 10: Results for PluGeN using β-VAE backbone for different values of β.

β	0.5	1	2	4	8	16
FID	61.86	55.11	61.96	65.76	77.94	110.46
F1	0.45	0.66	0.63	0.59	0.57	0.53
AUC	0.79	0.90	0.88	0.87	0.86	0.83

of β gets too low. The flow-based model cannot disentangle factors of variation from latent space which is not already at least partially structured. This experiment shows limitations of our model in respect to its reliance on the performance of the backbone autoencoder. However, PluGeN is still quite robust as it achieves good results for a wide range of β values.

APPENDIX C
DETAILS OF MOLECULES GENERATION EXPERIMENTS

C.1 Background

Designing a new drug is a long and expensive process that could cost up to 10 billion dollars and lasts even 10 years [50]. The recent spread of SARS-CoV-2 virus and the pandemic it caused have shown how important it is to speed up this process. Recently, deep learning is gaining popularity in the cheminformatics community, where it is used to propose new drug candidates. However, using neural networks in the drug generation task is not easy and is fraught with problems. The complexity of the chemical space is high and thus training generative and predictive models is challenging. Although there are around 10^{60} of possible molecules [51], detailed information (such as class labels) is known only about a small percentage of them.

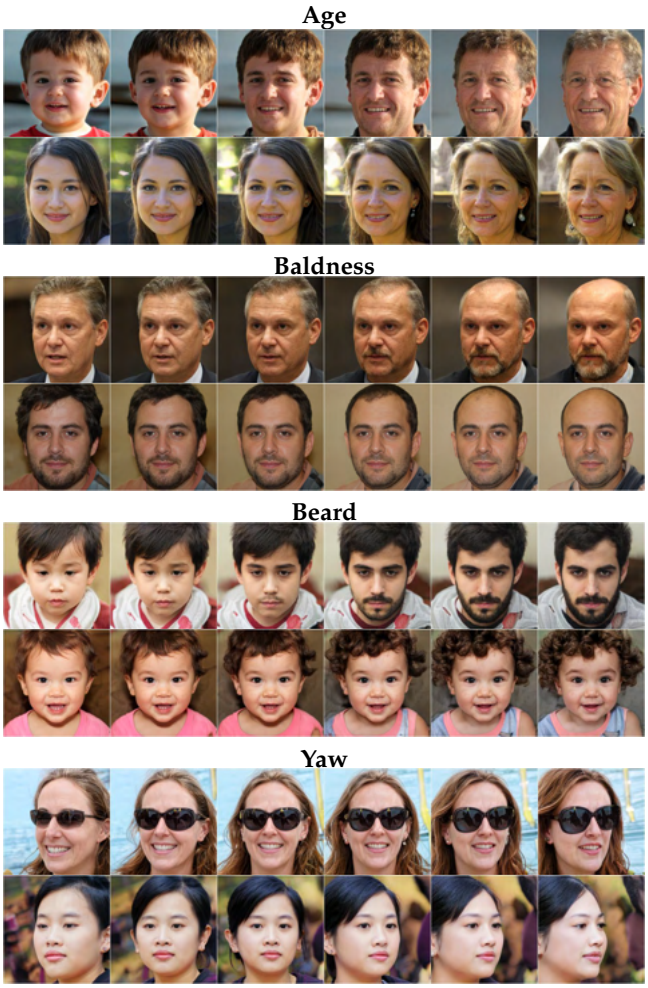


Fig. 19: Gradual modification of attributes (age, baldness, beard, and yaw, respectively) performed by PluGeN on partially labeled data using the StyleGAN backbone.

For example, the ChEMBL database [52], one of the biggest databases with information about the molecular attributes, contains data for 2.1 M chemical compounds. Moreover, since obtaining labeled data requires long and costly laboratory experiments, the amount of labeled molecules in the training datasets is usually really small (often less than 1000), which is often not sufficient to train a good model. This poses an important research problem.

Deep neural networks are mostly used in cheminformatics for the following tasks:

- virtual screening – the search for potentially active compounds in the libraries of commercially available molecules using predictive models [53], [54],
- de novo design – generating new compounds with desirable properties that are not present in the above-mentioned libraries [39], [40],
- lead optimization – improving selected promising compounds to meet certain criteria [41], [42].

PluGeN can be used for the two latter tasks, as our model can generate molecules with specified values of given attributes as well as optimize molecules by changing the value of selected labels.

Fig. 20: Attributes manipulation performed by PluGeN on partially labeled data using the StyleGAN backbone.

TABLE 11: Correlations of attributes for chemical molecules modeling.

	logP	TPSA	SAS
logP	1.00	-0.16	0.51
TPSA	-0.16	1.00	-0.18
SAS	-0.51	-0.18	1.00

SMILES representation. SMILES [36] (simplified molecular-input line-entry system) is a notation, for describing the structure of chemical species using a sequence of characters. SMILES representation consists of a specially defined grammar, which guarantees that a correct SMILES defines a unique molecule. The opposite is not actually true, as a molecule could be encoded by multiple SMILES representations. In order to add this property, the community introduced the canonicalization algorithm, which returns the canonical SMILES that is unique for each molecule.

In Figure 25 we show two molecules together with their canonical SMILES as well as other SMILES representations.

Modeled attributes. In our chemistry experiments, we modeled 3 chemical attributes: logP, TPSA, and SAS. Below, we describe their responsibilities:

- logP – logarithm of the partition coefficient. Describes the molecule solubility in fats. It shows how well the molecule is passing through membranes.
- TPSA – the topological polar surface area of a molecule is the surface sum over all polar atoms or molecules (together with their attached hydrogen atoms). TPSA could be used as a metric of the ability of a drug to permeate cells.
- SAS – synthetic accessibility score defines the ease of synthesis of a drug-like molecule. When generating a drug candidate, one would rather want it to be easily

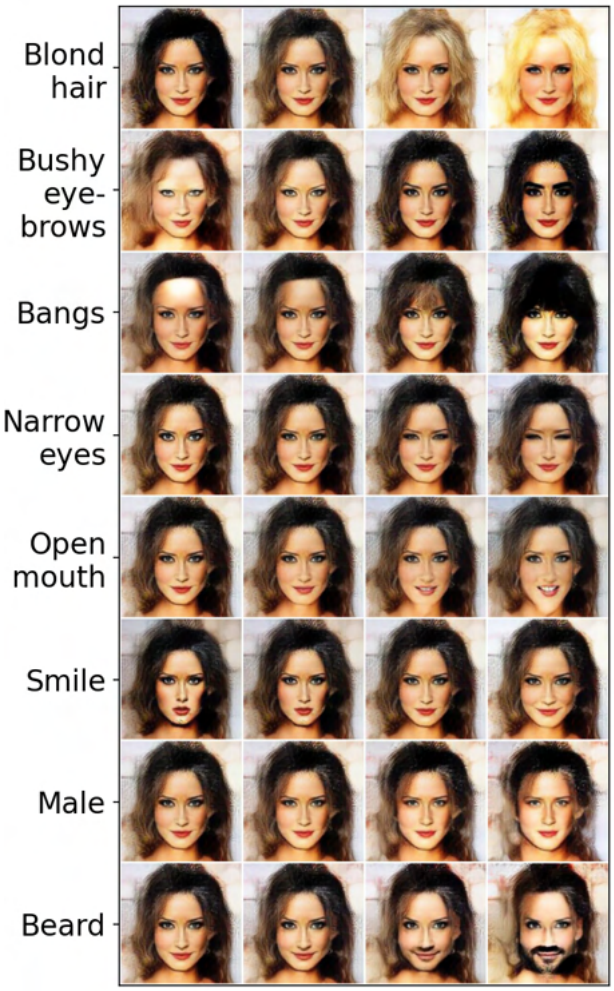


Fig. 21: Manipulating the intensity of labeled attributes of the generated sample. Since PluGeN models the values of the attributes with continuous distributions, it can control the intensity of each attribute and even sometimes extrapolate outside the data distribution (e.g. very bright blond hair).

synthesized so that it can be obtained in the laboratory.

Dataset. We conducted our chemistry experiments using a dataset of 250k molecules sampled from the ZINC database [37], which is a dataset of commercially available chemical compounds. The mean number of SMILES tokens in our dataset is equal to 38.31, with a standard deviation equal to 8.46.

Figure 26 shows the distribution of attributes of molecules that make up our training dataset.

Since the values of the chemical attributes are related to the structure of the molecule, many of them will be correlated in some way. In Table 11 we present the correlations between the chemistry attributes. The correlations suggest that it might be difficult or even impossible to manipulate logP and SAS attributes independently, setting a difficult challenge for PluGeN.

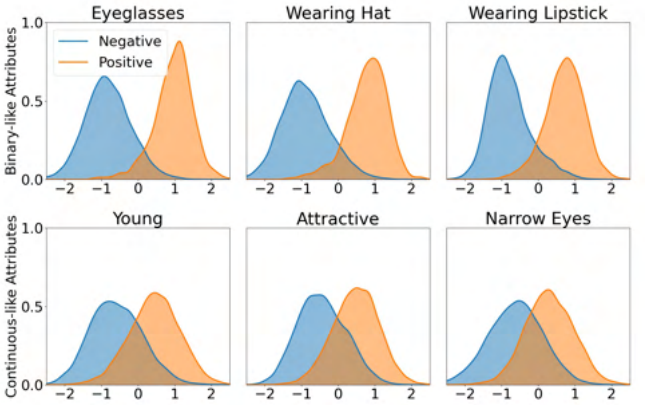


Fig. 22: The density of the positive and negative samples for chosen attributes in the flow latent space estimated using all examples from the CelebA test set. Binary attributes (top row) are clearly separated while continuous attributes (bottom row) overlap significantly.

C.2 Hyperparameters

VAE. The encoder consists of 3 bi-GRU [55] layers, with hidden size equal to 256 and output size (latent dimensionality) equal to 100. The decoder consists of 3 GRU layers with the hidden size equal to 256. The architecture of the backbone model is significantly different from the one used in the image domain, which partially confirms that PluGeN can be combined with various autoencoder models.

We trained the VAE model for 100 epochs, using batch size of 256 and learning rate equal to 1e-4.

NICE. The flow model consisted of 6 coupling layers, each of which consists of 6 dense layers with a hidden size equal to 256. We trained NICE for 50 epochs, with learning rate equal to 1e-4 and batch size 256. We used $\sigma_0 = 1.0$ and $\gamma = 0.9$.

C.3 Additional experiments

In the following subsection, we show additional results for the chemistry-based experiments, for both conditional generation as well as latent space traversal. Furthermore, we show how PluGeN works with the conditional normalizing flow instead of NICE as a base flow model.

Conditional generation. In the main paper, we presented results for conditional generation in the setting of a single attribute condition (where the value of the remaining attributes was sampled from their prior distribution). Here we also show results for a situation where set conditions on all attributes at the same time.

In particular, we tested 3 different settings:

1) LogP set to 1.0, TPSA set to 60.0, SAS set to 5.0.
2) LogP set to 3.0, TPSA set to 75.0, SAS set to 3.0.
3) LogP set to 5.0, TPSA set to 50.0, SAS set to 2.0.

The density plots of the attributes of the molecules generated in these settings are presented in Figure 27.

Latent space traversal. We also present more results for latent space traversals, which is a task that imitates the interclass interpolation experiments from the image domain. For this purpose, we tested how PluGeN can traverse the

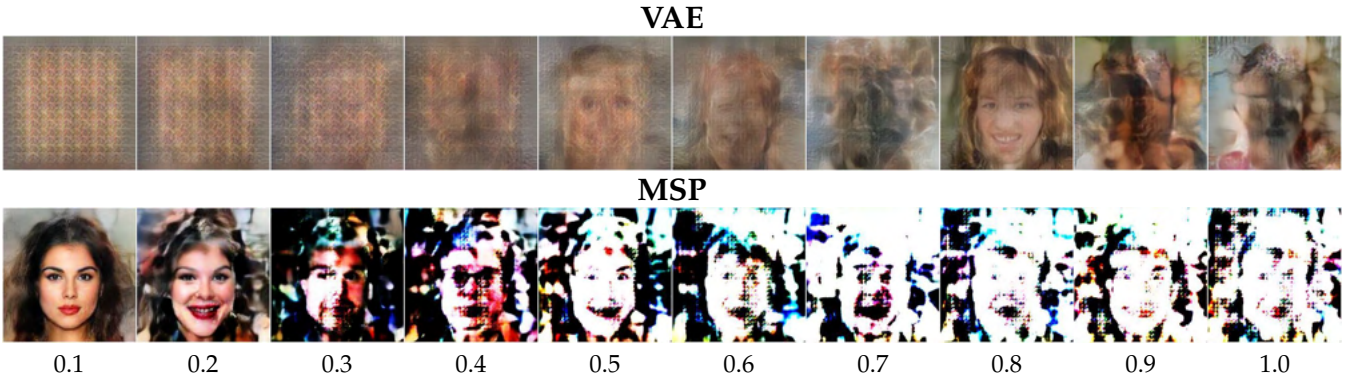
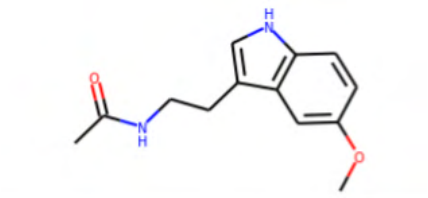


Fig. 23: Samples from the base VAE (top row) and MSP (bottom row) models using increasing values of the temperature parameter (bottom line). MSP generates typical face images only for a very low temperature, while VAE does not generate face images at all.

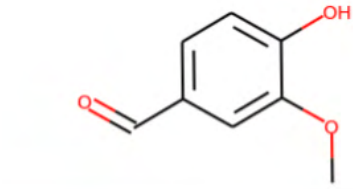
Fig. 24: Manipulation of continuous (age, baldness, yaw) performed by the semi-supervised version of PluGeN on the StyleGAN latent codes.



Canonical SMILES:
COc1ccc2[nH]cc(CCNC(C)=O)c2c1

Other SMILES:
CC(=O)NCCC1=CNc2c1cc(OC)cc2
CC(=O)NCCc1c[nH]c2ccc(OC)cc12

(a) Melatonin



Canonical SMILES:
COc1cc(C=O)ccc1O

Other SMILES:
O=Cc1ccc(O)c(OC)c1

(b) Vanillin

Fig. 25: Sample molecules together with their SMILES representations.

latent space of CharVAE. Therefore, we selected a few random molecules from our dataset, and for every one, we forced PluGeN to gradually increase the value of the specified attribute by some value and decoded the resulting molecules back into the latent space. The goal of this task is to generate the molecules that are structurally similar to the initial one, except for changes in the desired attributes. This is an important challenge in the *lead optimization stage* of the drug discovery process.

LogP For LogP, we forced PluGeN to increase the molecular attribute value by 3. Figures 28 and 29 show the obtained molecules, together with the optimized attribute values.

TPSA For TPSA, we forced PluGeN to increase the molecular attribute value by 40. Figures 30 and 31 show the obtained molecules, together with the optimized attribute values.

SAS. For SAS, we forced PluGeN to increase the molecular attribute value by 2. Figures 32 and 33 show the obtained molecules, together with the optimized attribute values.

CNF vs NICE. We also tested how replacing NICE [30] with conditional normalizing flow [49] affects the process of molecular generation using PluGeN. For this purpose, we repeated the chemistry-based conditional generation experiments from the main text, but with CNF as our backbone flow model. Results are presented in Figure 34. One can see, that in this version PluGeN is also capable of moving the density of the attributes of the generated molecules towards the desired value. The obtained changes, however, are worse than in the case of NICE as a flow backbone.

Appendix D
Details of 3D point cloud generation

Comparison with FPN. In this paragraph, we conduct a short qualitative comparison in the area of point clouds.

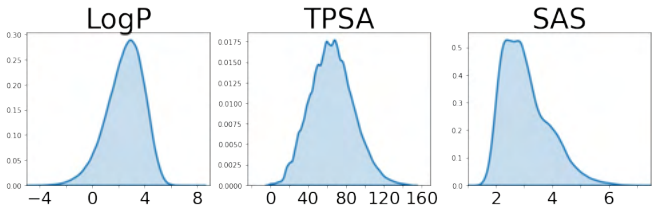
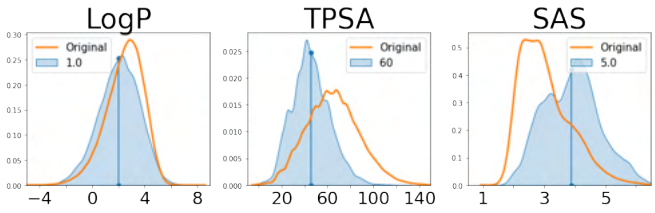
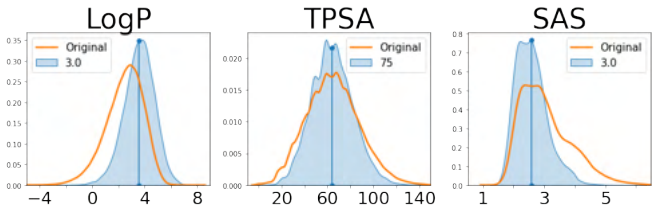


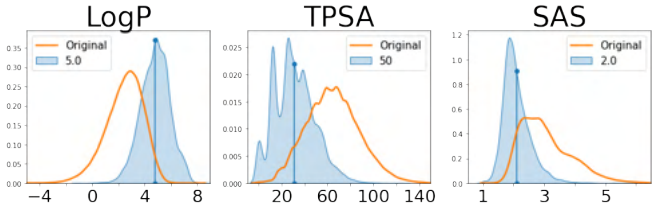
Fig. 26: Density plots of chemistry attributes present in the training dataset.



(a) LogP = 1.0, TPSA = 60.0, SAS = 5.0



(b) LogP = 3.0, TPSA = 75.0, SAS = 3.0



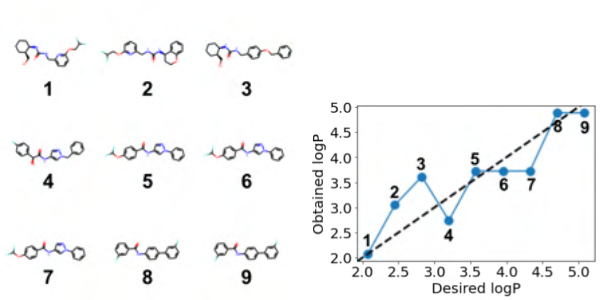
(c) LogP = 5.0, TPSA = 50.0, SAS = 2.0

Fig. 27: Distribution of labeled attributes for generated molecules (for the experiment with multiple attributes condition), together with distribution for the training dataset. The average of every distribution is marked with a vertical line.

The results of the Flow Plug-in Network (FPN) for the conditional generation experiment are presented in Figure 35 and for the attribute manipulation in Figure 36. Comparing the results of FPN with PluGeN (Figures 12 and 13 in the main paper), we can observe that both methods produce results of similar quality; however, PluGeN method gives the impression of better label preservation. The results in Table 4 (in the main paper) confirm that observation, as most of the metrics are comparable, except for label classification accuracy that is in favor of PluGeN and suggests that indeed it better behaves in terms of the correct attribution.

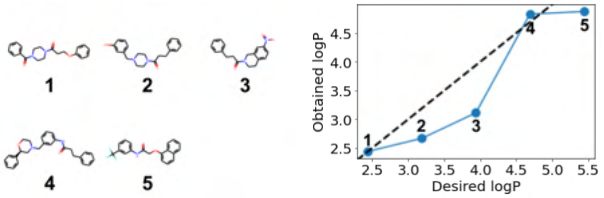
Acknowledgments

The authors thank the Reviewers and the Editor for their valuable comments and insightful feedback, which helped us further improve the paper.



(a) Molecules decoded from path (b) LogP of presented molecules

Fig. 28: Molecules obtained by the model during an optimization phase (left side), together with their LogP (right side).



(a) Molecules decoded from path (b) LogP of presented molecules

Fig. 29: Molecules obtained by the model during an optimization phase (left side), together with their LogP (right side).

The research of M. Wołczyk was supported by the Foundation for Polish Science co-financed by the European Union under the European Regional Development Fund in the POIR.04.04.00-00-14DE/18-00 project carried out within the Team-Net program. The work carried out by Maciej Zieba was supported by the National Centre of Science (Poland) Grant No. 2020/37/B/ST6/03463. The work of Ł. Maziarka was supported by the National Science Centre (Poland) grant no. 2019/35/N/ST6/02125. The research of M. Śmieja was funded by the National Science Centre (Poland) grant no. 2022/45/B/ST6/01117.

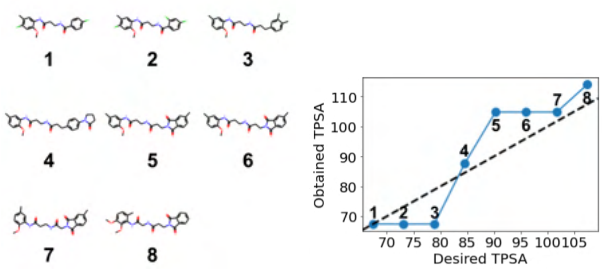
References

[1] A. Brock, J. Donahue, and K. Simonyan, "Large scale gan training for high fidelity natural image synthesis," in International Conference on Learning Representations, 2018.

[2] R. Gómez-Bombarelli, J. N. Wei, D. Duvenaud, J. M. Hernández-Lobato, B. Sánchez-Lengeling, D. Sheberla, J. Aguilera-Iparraguirre, T. D. Hirzel, R. P. Adams, and A. Aspuru-Guzik, "Automatic chemical design using a data-driven continuous representation of molecules," ACS central science, vol. 4, no. 2, pp. 268–276, 2018.

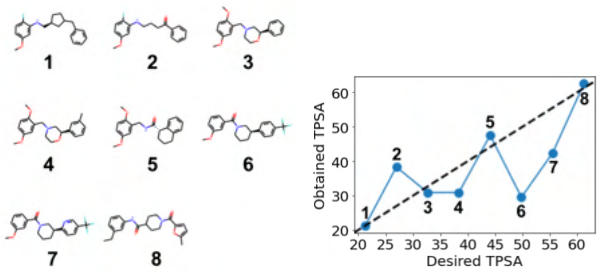
[3] M. Wołczyk, B. Wójcik, K. Bałazy, I. Podolak, J. Tabor, M. Śmieja, and T. Trzciński, "Zero time waste: Recycling predictions in early exit neural networks," arXiv preprint arXiv:2106.05409, 2021.

[4] S. Rebuffi, H. Bilen, and A. Vedaldi, "Learning multiple visual domains with residual adapters," in Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA,



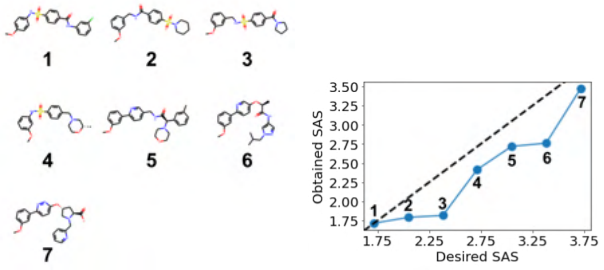
(a) Molecules decoded from path (b) TPSA of presented molecules

Fig. 30: Molecules obtained by the model during an optimization phase (left side), together with their TPSA (right side).



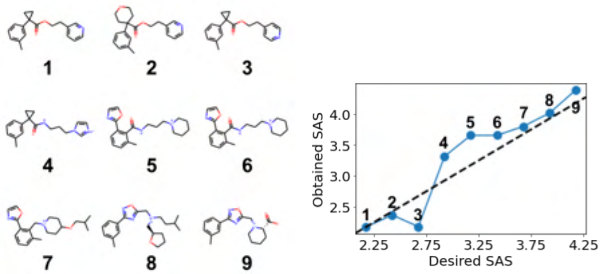
(a) Molecules decoded from path (b) TPSA of presented molecules

Fig. 31: Molecules obtained by the model during an optimization phase (left side), together with their TPSA (right side).



(a) Molecules decoded from path (b) SAS of presented molecules

Fig. 32: Molecules obtained by the model during an optimization phase (left side), together with their SAS (right side).



(a) Molecules decoded from path (b) SAS of presented molecules

Fig. 33: Molecules obtained by the model during an optimization phase (left side), together with their SAS (right side).

USA, I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, Eds., 2017, pp. 506–516.

[5] R. Abdal, P. Zhu, N. J. Mitra, and P. Wonka, "Styleflow: Attribute-conditioned exploration of stylegan-generated images using conditional continuous normalizing flows," ACM Transactions on Graphics (TOG), vol. 40, no. 3, pp. 1–21, 2021.

[6] P. Wielopolski, M. Koperski, and M. Zieba, "Flow plugin network for conditional generation," arXiv preprint arXiv:2110.04081, 2021.

[7] X. Li, C. Lin, R. Li, C. Wang, and F. Guerin, "Latent space factorisation and manipulation via matrix subspace projection," in International Conference on Machine Learning. PMLR, 2020, pp. 5916–5926.

[8] M. Wołczyk, M. Proszewska, Ł. Maziarka, M. Zieba, P. Wielopolski, R. Kurczab, and M. Smieja, "Plugen: Multi-label conditional generation from pre-trained models," in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 36, no. 8, 2022, pp. 8647–8656.

[9] D. P. Kingma, D. J. Rezende, S. Mohamed, and M. Welling, "Semi-supervised learning with deep generative models," arXiv preprint arXiv:1406.5298, 2014.

[10] J. Klys, J. Snell, and R. Zemel, "Learning latent subspaces in variational autoencoders," arXiv preprint arXiv:1812.06190, 2018.

[11] S. Kang and K. Cho, "Conditional molecular design with deep generative models," Journal of chemical information and modeling, vol. 59, no. 1, pp. 43–52, 2018.

[12] Z. He, W. Zuo, M. Kan, S. Shan, and X. Chen, "Attgan: Facial attribute editing by only changing what you want," IEEE Transactions on Image Processing, vol. 28, no. 11, pp. 5464–5478, 2019.

[13] N. Kodali, J. Abernethy, J. Hays, and Z. Kira, "On convergence and stability of gans," arXiv preprint arXiv:1705.07215, 2017.

[14] Y. Choi, Y. Uh, J. Yoo, and J.-W. Ha, "Stargan v2: Diverse image synthesis for multiple domains," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 8188–8197.

[15] O. Tov, Y. Alaluf, Y. Nitzan, O. Patashnik, and D. Cohen-Or, "Designing an encoder for stylegan image manipulation," ACM Transactions on Graphics (TOG), vol. 40, no. 4, pp. 1–14, 2021.

[16] G. Lample, N. Zeghidour, N. Usunier, A. Bordes, L. Denoyer, and M. Ranzato, "Fader networks: Manipulating images by sliding attributes," arXiv preprint arXiv:1706.00409, 2017.

[17] R. Liu, Y. Liu, X. Gong, X. Wang, and H. Li, "Conditional adversarial generative flow for controllable image synthesis," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 7992–8001.

[18] D. P. Kingma and P. Dhariwal, "Glow: Generative flow with invertible 1x1 convolutions," arXiv preprint arXiv:1807.03039, 2018.

[19] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 4401–4410.

[20] Y. Gao, F. Wei, J. Bao, S. Gu, D. Chen, F. Wen, and Z. Lian, "High-fidelity and arbitrary face editing," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 16 115–16 124.

[21] A. Tewari, M. Elgharib, F. Bernard, H.-P. Seidel, P. Pérez, M. Zollhöfer, and C. Theobalt, "Pie: Portrait image embedding for semantic control," ACM Transactions on Graphics (TOG), vol. 39, no. 6, pp. 1–14, 2020.

[22] Y. Nitzan, A. Bermano, Y. Li, and D. Cohen-Or, "Disentangling in latent space by harnessing a pretrained generator," arXiv preprint arXiv:2005.07728, vol. 2, no. 3, 2020.

[23] Y. Shen, C. Yang, X. Tang, and B. Zhou, "Interfacegan: Interpreting the disentangled face representation learned by gans," IEEE transactions on pattern analysis and machine intelligence, 2020.

[24] Y. Shen, J. Gu, X. Tang, and B. Zhou, "Interpreting the latent space of gans for semantic face editing," in Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2020, pp. 9243–9252.

[25] H.-P. Wang, N. Yu, and M. Fritz, "Hijack-gan: Unintended-use of pretrained, black-box gans," in Proceedings of the IEEE/CVF

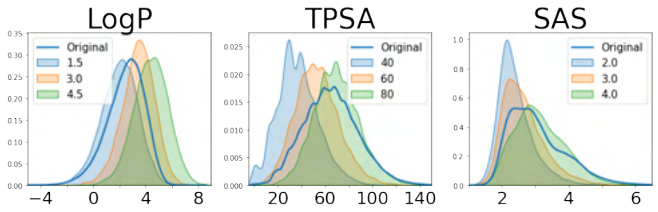


Fig. 34: Distribution of labeled attributes for generated molecules for PluGeN with the conditional normalizing flow, together with distribution for the training dataset. Each color shows the value of the labeled attribute that was used for generation.

Conference on Computer Vision and Pattern Recognition, 2021, pp. 7872–7881.

[26] E. Härkönen, A. Hertzmann, J. Lehtinen, and S. Paris, "Ganspace: Discovering interpretable gan controls," arXiv preprint arXiv:2004.02546, 2020.

[27] Y. Shen and B. Zhou, "Closed-form factorization of latent semantics in gans," in CVPR, 2021.

[28] H. Kim and A. Mnih, "Disentangling by factorising," in International Conference on Machine Learning. PMLR, 2018, pp. 2649–2658.

[29] R. T. Chen, X. Li, R. Grosse, and D. Duvenaud, "Isolating sources of disentanglement in vaes," in Proceedings of the 32nd International Conference on Neural Information Processing Systems, 2019, pp. 2615–2625.

[30] L. Dinh, D. Krueger, and Y. Bengio, "Nice: Non-linear independent components estimation," arXiv preprint arXiv:1410.8516, 2014.

[31] F. Locatello, S. Bauer, M. Lucic, G. Raetsch, S. Gelly, B. Schölkopf, and O. Bachem, "Challenging common assumptions in the unsupervised learning of disentangled representations," in international conference on machine learning. PMLR, 2019, pp. 4114–4124.

[32] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," arXiv preprint arXiv:1611.01144, 2016.

[33] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," in Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 2017, pp. 6626–6637.

[34] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[35] S. Beery, G. V. Horn, and P. Perona, "Recognition in terra incognita," in ECCV, 2018.

[36] D. Weininger, "Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules," Journal of chemical information and computer sciences, vol. 28, no. 1, pp. 31–36, 1988.

[37] T. Sterling and J. J. Irwin, "Zinc 15–ligand discovery for everyone," Journal of chemical information and modeling, vol. 55, no. 11, pp. 2324–2337, 2015.

[38] G. Landrum et al., "Rdkit: Open-source cheminformatics," 2006.

[39] M. Olivecrona, T. Blaschke, O. Engkvist, and H. Chen, "Molecular de-novo design through deep reinforcement learning," Journal of cheminformatics, vol. 9, no. 1, pp. 1–14, 2017.

[40] M. Popova, O. Isayev, and A. Tropsha, "Deep reinforcement learning for de novo drug design," Science advances, vol. 4, no. 7, p. eaap7885, 2018.

[41] W. Jin, K. Yang, R. Barzilay, and T. Jaakkola, "Learning multimodal graph-to-graph translation for molecular optimization," International Conference on Learning Representations, 2019.

[42] Ł. Maziarka, A. Pocha, J. Kaczmarczyk, K. Rataj, T. Danel, and M. Warchoł, "Mol-cyclegan: a generative model for molecular optimization," Journal of Cheminformatics, vol. 12, no. 1, pp. 1–18, 2020.

[43] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su et al., "Shapenet: An information-rich 3d model repository," arXiv preprint arXiv:1512.03012, 2015.

[44] G. Yang, X. Huang, Z. Hao, M.-Y. Liu, S. Belongie, and B. Hariharan, "Pointflow: 3d point cloud generation with continuous normalizing flows," in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 4541–4550.

[45] X. Zeng, A. Vahdat, F. Williams, Z. Gojcic, O. Litany, S. Fidler, and K. Kreis, "LION: latent point diffusion models for 3d shape generation," in NeurIPS, 2022.

[46] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in Advances in Neural Information Processing Systems 32, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[47] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, "Analyzing and improving the image quality of StyleGAN," in Proc. CVPR, 2020.

[48] C. Li and M. Wand, "Precomputed real-time texture synthesis with markovian generative adversarial networks," in European conference on computer vision. Springer, 2016, pp. 702–716.

[49] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, "Neural ordinary differential equations," arXiv preprint arXiv:1806.07366, 2018.

[50] J. Mestre-Ferrandiz, J. Sussex, A. Towse et al., "The r&d cost of a new medicine," Monographs, 2012.

[51] R. S. Bohacek, C. McMartin, and W. C. Guida, "The art and practice of structure-based drug design: a molecular modeling perspective," Medicinal research reviews, vol. 16, no. 1, pp. 3–50, 1996.

[52] A. Gaulton, A. Hersey, M. Nowotka, A. P. Bento, J. Chambers, D. Mendez, P. Mutowo, F. Atkinson, L. J. Bellis, E. Cibrián-Uhalte et al., "The chembl database in 2017," Nucleic acids research, vol. 45, no. D1, pp. D945–D954, 2017.

[53] C. W. Coley, R. Barzilay, W. H. Green, T. S. Jaakkola, and K. F. Jensen, "Convolutional embedding of attributed molecular graphs for physical property prediction," Journal of chemical information and modeling, vol. 57, no. 8, pp. 1757–1772, 2017.

[54] K. Yang, K. Swanson, W. Jin, C. Coley, P. Eiden, H. Gao, A. Guzman-Perez, T. Hopper, B. Kelley, M. Mathea et al., "Analyzing learned molecular representations for property prediction," Journal of chemical information and modeling, vol. 59, no. 8, pp. 3370–3388, 2019.

[55] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," arXiv preprint arXiv:1406.1078, 2014.

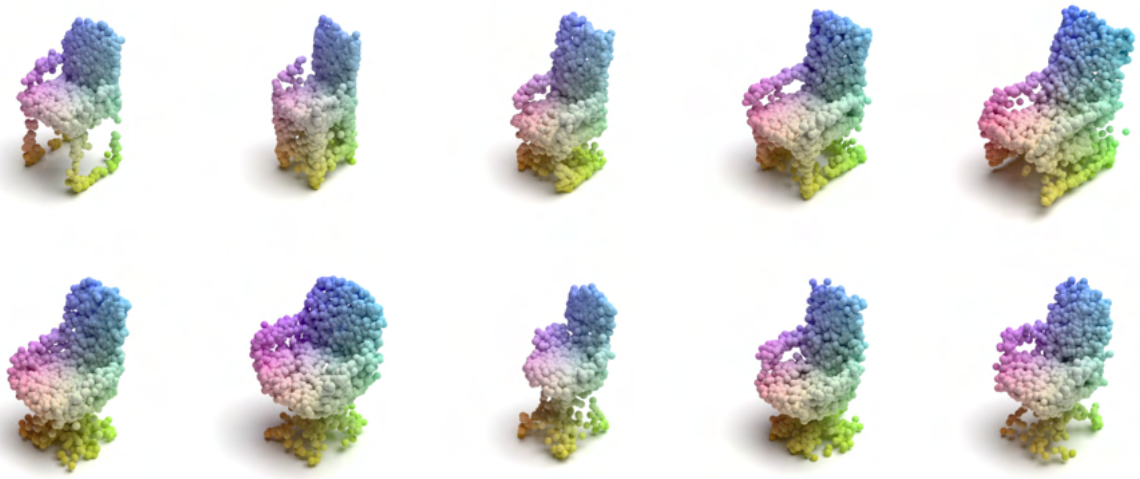


Fig. 35: The results of conditional point cloud generation experiment using reference method - Flow Plug-in Network (FPN). The first row is generated using cantilever arm chairs label combination, and the second row uses swivel arm chairs.

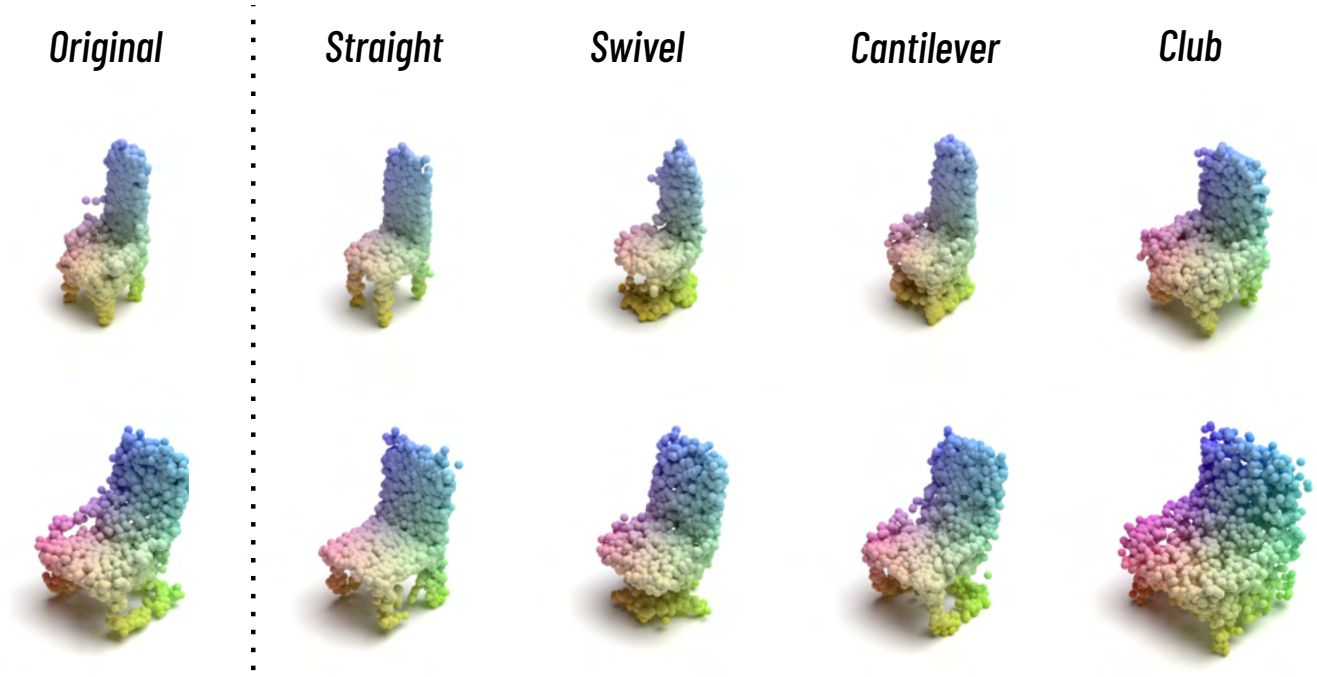


Fig. 36: Point cloud attribute manipulation results for the Flow Plug-in Network (FPN) - reference method to PluGeN in the area of point cloud domain.

Probabilistically Plausible Counterfactual Explanations with Normalizing Flows

Patryk Wielopolski[a,*], **Oleksii Furman**[a], **Jerzy Stefanowski**[b] **and Maciej Zięba**[a,c]

[a]Wrocław University of Science and Technology
[b]Poznań University of Technology
[c]Tooploox Sp. z o.o.

Abstract. We present PPCEF, a novel method for generating probabilistically plausible counterfactual explanations (CFs). PPCEF advances beyond existing methods by combining a probabilistic formulation that leverages the data distribution with the optimization of plausibility within a unified framework. Compared to reference approaches, our method enforces plausibility by directly optimizing the explicit density function without assuming a particular family of parametrized distributions. This ensures CFs are not only valid (i.e., achieve class change) but also align with the underlying data's probability density. For that purpose, our approach leverages normalizing flows as powerful density estimators to capture the complex high-dimensional data distribution. Furthermore, we introduce a novel loss function that balances the trade-off between achieving class change and maintaining closeness to the original instance while also incorporating a probabilistic plausibility term. PPCEF's unconstrained formulation allows for an efficient gradient-based optimization with batch processing, leading to orders of magnitude faster computation compared to prior methods. Moreover, the unconstrained formulation of PPCEF allows for the seamless integration of future constraints tailored to specific counterfactual properties. Finally, extensive evaluations demonstrate PPCEF's superiority in generating high-quality, probabilistically plausible counterfactual explanations in high-dimensional tabular settings.

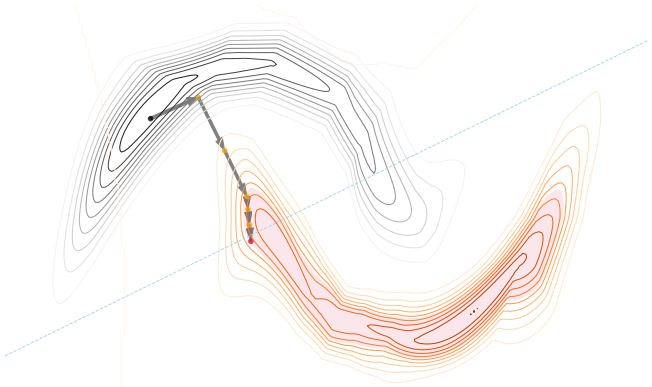


Figure 1: Probabilistically Plausibile Counterfactual Explanation Estimation Process on the Moons Dataset. We show an evolution of an instance from the initial instance (black dot) to the final counterfactual (red dot) against the linear classifier's decision boundary (blue line) and density threshold contours, highlighting the method's trajectory towards achieving target classification and probabilistic plausibility condition.

1 Introduction

Counterfactual explanations (briefly *counterfactuals*, and abbreviated as CF) are one particular type of such explanations of black box model predictions that provide information about how feature values of an example should be changed to obtain a more desired prediction of the model (i.e., to change its target decision) [30]. On the one hand, by interacting with the model using counterfactuals, the user can better understand how the system works by exploring "what would have happened if..." scenarios. On the other hand, a good counterfactual provides a practical recommendation to the user about what changes are needed in order to achieve the desired outcome.

There are many practical applications for counterfactual explanations, including loan or insurance decisions [31], recruitment processes [21], the discovery of chemical compounds [32], medical diagnosis [17], and many others, see, e.g., the recent survey [10].

More formally, a counterfactual explanation is an alternative input instance, denoted as $\mathbf{x}'$, which is minimally modified from the description of the original instance $\mathbf{x}_0$, such that the output of the classifier h changes from the original decision $y = h(\mathbf{x}_0)$ to a specific desired outcome $y' = h(\mathbf{x}')$.

Up to now, several algorithms for generating counterfactual explanations have been introduced. They are based on different principles, and for comprehensive surveys, see, e.g., [10, 30]. Depending on the specific method, some properties of counterfactuals are expected to be met, such as *validity* of the decision change, *proximity* to the input instance, *sparsity* of recommended changes, their *actionability*, i.e., the counterfactual should not modify immutable features or violate monotonic constraints, and *plausibility* of locating the counterfactual within a high-density region of the data, ensuring that the proposed counterfactuals are realistic and feasible within the context of the observed data distribution.

Many of these methods are inspired by the formulation of Wachter et al. [31], which proposed framing counterfactual explanations as an unconstrained optimization problem. For a prediction function h and an input $\mathbf{x}_0 \in \mathbb{R}^d$, a counterfactual $\mathbf{x}' \in \mathbb{R}^d$ is computed by solving:

$$\arg\min_{\mathbf{x}' \in \mathbb{R}^d} \ell(h(\mathbf{x}'), y') + C \cdot d(\mathbf{x}_0, \mathbf{x}'). \tag{1}$$

* Corresponding Author. Email: patryk.wielopolski@pwr.edu.pl.

In this formulation, $\ell(\cdot, \cdot)$ represents a classification loss function, $d(\cdot, \cdot)$ is a penalty for deviation from the original input $\mathbf{x}_0$, and the term $C \geq 0$ serves as the regularization strength modifier.

An alternative approach [2] frames counterfactual explanations as a constrained optimization problem. This perspective focuses on directly finding the minimal perturbation required to achieve the target prediction under the constraint that the model's prediction for the counterfactual instance meets the specified criterion. Mathematically, this is represented as:

$$\arg\min_{\mathbf{x}' \in \mathbb{R}^d} d(\mathbf{x}_0, \mathbf{x}') \quad \text{s.t.} \quad h(\mathbf{x}') = y'. \tag{2}$$

In our study, we want to pay special attention to the *plausibility* of counterfactuals. Referring to arguments of [10], a counterfactual is plausible if the feature values describing the example are coherent (sufficiently similar) with those present in the original data X. This means it should be located in sufficiently dense regions of original instances in X from the target class. Plausibility helps in increasing users' trust in the explanation: it would be hard to trust a counterfactual if it is a combination of features that are unrealistic with respect to existing examples.

In previous works, plausibility has often been verified by simple k-neighbourhood analysis of the counterfactual with respect to the original data [26, 14, 27]. Few other approaches [3] model the conditional density in the target class and try to find the counterfactual example with the density value above the given threshold. Although the problem is quite well mathematically defined, the current methods apply simple approaches like kernel density estimators or a mixture of Gaussians to model conditional distributions that are difficult to apply for high-dimensional data. Moreover, the problem of estimating valid and plausible counterfactuals is defined as a complex constrained optimization problem with strict convexity assumptions [3]. Finally, the currently proposed methods, while providing valid counterfactuals, struggle to consistently produce observations that fulfill the plausibility criteria.

In this paper, we introduce PPCEF: Probabilistically Plausibile Counterfactual Explanations using Normalizing Flows - a novel approach to estimate counterfactual explanations for differentiable classifiers tailored for tabular problems. It includes a novel, unconstrained formulation of the problem that enables direct estimation of the plausibility property - to the best of our knowledge, a characteristic previously not achieved in the literature. For that purpose, we design loss functions to satisfy both validity and plausibility constraints and minimize the distance to the original example in a balanced way (see an example in Fig. 1). Our approach incorporates plausibility in the probabilistic sense by targeting observations with a probability density exceeding a predefined threshold [3]. Unlike existing methods limited to specific estimators of families of density functions, ours employs any differentiable conditional density model. Moreover, we postulate to utilize *conditional normalizing flows* for density estimation [24], ensuring independence from specific parameterized distribution families while enabling direct calculation of density values for complex, high-dimensional data. Finally, PPCEF leverages efficient batch processing utilizing gradient-based optimization techniques, leading to significant computational gains compared to previous methods.

To summarize, our contributions are as follows:

- The formulation of counterfactual explanations within an unconstrained optimization framework employing direct optimization of plausibility and novel loss functions.
- The utilization of normalizing flows as density estimators to capture the complex high-dimensional data distribution effectively.
- The experimental evaluations demonstrating PPCEF's ability to efficiently generate high-quality, probabilistically plausible counterfactuals in high-dimensional tabular datasets for both binary and multiclass classification problems, outperforming existing reference methods.

2 Related Works

2.1 Plausible Counterfactual Explanations

The approaches for obtaining plausible counterfactual explanations are primarily categorized into *endogenous* and *exogenous* ones [10]. Endogenous counterfactuals are crafted using feature values from existing data instances, ensuring their naturally occurring status and grounding them in real-world contexts, thereby enhancing their plausibility. In contrast, exogenous counterfactuals are generated through methods such as interpolations or random data generation, which do not strictly rely on existing dataset features. While this offers greater flexibility, it does not inherently assure the plausibility of these counterfactuals, as they might represent feature combinations not found in actual data.

2.1.1 Endogenous Counterfactual Explanations

Endogenous approaches to counterfactual explanations revolve around leveraging existing instances within the dataset to generate plausible counterfactuals. These methods, which include instance-based or case-based approaches, primarily utilize nearest neighbors' techniques to identify instances that closely resemble the input but yield different outcomes.

Examples of endogenous methods include the Nearest-Neighbor Counterfactual Explainer (NNCE) [26], selecting similar yet outcome-divergent instances from the dataset as counterfactuals. The Case-Based Counterfactual Explainer (CBCE) [14] forms 'explanation cases' by pairing similar instances with contrasting outcomes, creating counterfactuals by merging features from these pairs. Extending this concept, the approach by Smyth and Keane [27] adapts to k-nearest neighbors, utilizing multiple nearest candidates for generating counterfactuals. Feasible and Actionable Counterfactual Explanations (FACE) [23] constructs a graph over data points, applying user-defined parameters to find actionable paths to desired outcomes. Lastly, PROPLACE [12] employs bi-level optimization and Mixed-Integer Linear Programming, generating robust counterfactuals from Δ-robust nearest neighbors that closely align with data distribution and model robustness.

2.1.2 Exogenous Counterfactual Explanations

In the landscape of exogenous counterfactual explanations, methods generally involve introducing external modifications to original instances, diverging from reliance on existing instances and their features. These approaches utilize a range of computational techniques, such as autoencoders, linear programming, gradient-based methods, and generative models, to ensure that the resulting counterfactuals are plausible.

Firstly, the Contrastive Explanation Method (CEM) [5] innovates by adding perturbations to an instance and utilizing an autoencoder to verify the closeness of the modified instance to known data, ensuring plausibility. Meanwhile, the Diverse Coherent Explanations (DCE) [25] method leverages linear programming to create varied counterfactuals, with additional linear constraints to maintain both

diversity and plausibility. Further, the Distribution-Aware Counterfactual Explanation (DACE) [13] method incorporates the Mahalanobis distance and Local Outlier Factor (LOF) in its loss function, focusing on minimizing this distance while keeping a low LOF score to signify higher plausibility. The Diverse Counterfactual Explanations (DICE) [18] approach involves solving an optimization problem to generate multiple counterfactuals, with a specific emphasis on the diversity and actionability of these counterfactuals to determine their plausibility. Additionally, Counterfactual Explanations Guided by Prototypes (CEGP) [15] adopts a similar loss function to CEM but introduces a prototype-based loss term. This guides perturbations towards a counterfactual that aligns with the data distribution of a specific class, using the encoder of an autoencoder based on the average encoding of the nearest instances in the latent space with the same class label.

Within the field of exogenous counterfactual explanations, a subcategory particularly relevant to our work utilizes deep generative models. Variational Autoencoders (VAEs) are exploited in methods like Example-Based Counterfactual (EBCF) [16] and the approach by Vercheval and Pizurica [29]. EBCF incorporates known causal relationships into the VAE, promoting realistic counterfactuals. The method by Vercheval and Pizurica [29] enables visual counterfactual generation through VAE-based latent space exploration. Generative Adversarial Networks (GANs) play a crucial role in the PCATTGAN approach [1]. It utilizes adversarial examples within a multi-objective optimization framework to create plausible counterfactuals, considering validity, minimality, and a notion of plausibility defined as human-understandable, non-automated changes. Diffusion models underpin methods proposed in [11, 4]. While these approaches specialize in visual counterfactual generation, their focus lies primarily on counterfactual sampling, not controlling plausibility via density-based optimization. Lastly, Normalizing Flow-based methods [8, 9] center on pinpointing counterfactuals within their latent spaces. These methods leverage the invertible nature of normalizing flows to explore counterfactual regions in the latent representation of the data.

All of the reference methods, except Artelt and Hammer [3], do not provide an explicit probabilistic formulations of plausibility. Compared to Artelt and Hammer [3], we propose an alternative problem formulation in unconstrained form with no prior constraints on the density model.

3 Background

In this work, we consider the problem formulation of probabilistically plausible counterfactual explanations introduced by Artelt and Hammer [3]. This approach extends the problem formulation given by eq. (2) by adding a target-specific density constraint to enforce the plausibility of counterfactuals using a probabilistic framework. The constrained optimization problem is formulated as follows:

$$\arg\min_{\mathbf{x}' \in \mathbb{R}^d} d(\mathbf{x}_0, \mathbf{x}') \tag{3a}$$

$$\text{s.t.} \quad h(\mathbf{x}') = y' \tag{3b}$$

$$\delta \leq p(\mathbf{x}'|y'), \tag{3c}$$

where $p(\mathbf{x}'|y')$ denotes conditional probability of the counterfactual explanation $\mathbf{x}'$ under desired target class value y' and δ represents the density threshold.

This approach's crucial aspect is finding the proper model to represent the conditional density function $p(\mathbf{x}|y)$. Typically, kernel density estimators (KDEs) are used to model conditional densities, but the use of non-linear kernels results in the highly non-convex optimization problem formulation. Gaussian Mixture Model (GMM) can be applied alternatively, but convexity constraints are still not satisfied. To facilitate the desired optimization process, the authors of Artelt and Hammer [3] propose to approximate the density value $p(\mathbf{x}'|y')$ using a component-wise maximum of GMM components:

$$\hat{p}_G(\mathbf{x}'|y') = \max_j \Big(\pi_{j,y'} \mathcal{N}(\mathbf{x}'|\boldsymbol{\mu}_{j,y'}, \boldsymbol{\Sigma}_{j,y'}) \Big), \tag{4}$$

where $\boldsymbol{\mu}_{j,y'}$, $\boldsymbol{\Sigma}_{j,y'}$ and $\pi_{j,y'}$ are means, covariances and prior values for component j considering class y.

This approximation is transformed into a convex quadratic constraint for each GMM component j, resulting in the following formula:

$$(\mathbf{x}' - \boldsymbol{\mu}_{j,y'})^T \boldsymbol{\Sigma}_{j,y'} (\mathbf{x}' - \boldsymbol{\mu}_{j,y'}) + c_j \leq \delta', \tag{5}$$

where c_j is constant from the Gaussian normalization factor and $\delta' = -2 \log \delta$.

For each component j, the optimization problem is solved, resulting in a set of convex programs - one for each component. This step is crucial because knowing beforehand which component will produce a feasible and plausible counterfactual is impossible. Finally, the counterfactual $\mathbf{x}'$ that yields the smallest value for the objective function is selected.

However, this approach has few limitations. First, the number of components should be predefined for each class. Second, the family of parametrized distributions limits the ability to adjust to a data distribution. Third, the approach is difficult to be applied to high-dimensional data due to the Gaussian components.

In order to cope with the listed limitations, we postulate to model conditional density function $p(\mathbf{x}|y)$ using the normalizing flows [24]. This group of models can adjust to very complex, high-dimensional data distributions, which allows for calculating the density value from the change-of-variable formula. Moreover, we propose an alternative unconstrained problem formulation that allows solving using a gradient-based approach for any differentiable representation of conditional distribution $p(\mathbf{x}|y)$.

4 Method

This section introduces a novel approach to the problem of plausible counterfactual explanation formulated by eq. (3). First, we reformulate the problem of calculating counterfactuals as unconstrained optimization suitable for direct, gradient-based optimization. Next, we show how to train the flow model to estimate the class-conditional distributions. Finally, we show how the counterfactuals can be efficiently estimated using a gradient-based approach.

4.1 Unconstrained Probabilistically Plausible Counterfactual Explanations

We consider a binary classification problem, $y \in \{0, 1\}$. However, our considerations can be easily extended to the multiclass case. Further, we consider a discriminative differentiable model (e.g., Logistic Regression or MLP) $p_d(y|\mathbf{x})$ and reformulate the validity constraint $h(\mathbf{x}') = y'$ as $p_d(y'|\mathbf{x}') \geq 0.5 + \epsilon$, where $\epsilon \to 0$, practically represented as small enough value close to 0.

We postulate the following unconstrained optimization problem:

$$\arg\min_{\mathbf{x}' \in \mathbb{R}^d} d(\mathbf{x}_0, \mathbf{x}') + \lambda \cdot \Big(\ell_v(\mathbf{x}', y') + \ell_p(\mathbf{x}', y') \Big), \tag{6}$$

where $\lambda = \infty$, practically, is large enough.

The loss $\ell_v(\mathbf{x}', y')$ component controls the validity constraint and is defined as follows:

$$\ell_v(\mathbf{x}', y') = \max\Big(0.5 + \epsilon - p_d(y'|\mathbf{x}'), 0\Big). \quad (7)$$

The Binary Cross Entropy (BCE) criterion can be used alternatively. However, using such criteria enforces 100% confidence of the discriminative model, while our approach aims at achieving the current classification accuracy with the margin controlled with the ϵ parameter. While using our criterion, the model can focus more on producing closer and more plausible counterfactuals, which we show in ablation studies.

Additionally, we extend the validity loss component to the multiclass scenario in the following way:

$$\ell_v(\mathbf{x}', y') = \max\Big(\max_{y \neq y'} p_d(y|\mathbf{x}') + \epsilon - p_d(y'|\mathbf{x}'), 0\Big), \quad (8)$$

where we replace the 0.5 threshold value with the highest probability value returned by the discriminative model, excluding the value for target class y'. This guarantees that $p_d(y|\mathbf{x}')$ will be higher than the most probable class among the remaining classes by the ϵ margin.

The loss component $\ell_p(\mathbf{x}', y')$ controls probabilistic plausibility constraint ($\delta \leq p(\mathbf{x}'|y')$) and is defined as:

$$\ell_p(\mathbf{x}', y') = \max\Big(\delta - p(\mathbf{x}'|y'), 0\Big), \quad (9)$$

where δ is the density threshold calculated in the same way as in [3], i.e., by utilizing the median of the training dataset. The conditional distribution $p(\mathbf{x}|y)$ can be represented by any differentiable model (e.g., Mixture of Gaussians, KDE). In this work, we postulate to model the distribution using conditional normalizing flow due to the flexibility and ability to adjust to multidimensional complex distributions. Thanks to the unconstrained problem formulation given by eq. (6) and differentiation assumption for the models, the counterfactuals can be easily calculated using a gradient-based approach.

4.2 Probabistically Plausible Counterfactual Explanations via Normalizing Flow-based Density Estimation

KDE or GMMs can be used to model the conditional distributions. However, those models have limited modeling capabilities due to the parametrized (usually Gaussian) form of $p(\mathbf{x}|y)$ or the inability to model high-dimensional data (KDE). Therefore, in this work, we postulate the use of a conditional normalizing flow model [24] to estimate the density for the joint distribution of the attributes for each class.

Normalizing Flows have surged in popularity within generative models due to their adaptability and the simplicity of training via direct negative log-likelihood (NLL) optimization. Their adaptability stems from the change-of-variable technique, which transforms a latent variable $\mathbf{z}$ with a known prior distribution $p(\mathbf{z})$ into an observed space variable $\mathbf{x}$ with an unknown distribution. This transformation occurs through a sequence of invertible (parametric) functions: $\mathbf{x} = \mathbf{f}_K \circ \cdots \circ \mathbf{f}_1(\mathbf{z}, y)$. Assuming a known prior $p(\mathbf{z})$ for $\mathbf{z}$, the conditional log-likelihood for $\mathbf{x}$ is expressed as:

$$\log \hat{p}_F(\mathbf{x}|y) = \log p(\mathbf{z}) - \sum_{k=1}^{K} \log \left| \det \frac{\partial \mathbf{f}_k}{\partial \mathbf{z}_{k-1}} \right|, \quad (10)$$

where $\mathbf{z} = \mathbf{f}_1^{-1} \circ \cdots \circ \mathbf{f}_K^{-1}(\mathbf{x}, y)$ is a result of the invertible mapping. The biggest challenge in normalizing flows is the choice of the invertible functions $\mathbf{f}_K, \ldots, \mathbf{f}_1$. Several solutions have been proposed in the literature to address this issue with notable approaches, including NICE [6], RealNVP [7], and MAF [19].

For a given training set $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^N$ we simply train the conditional normalizing flow by minimizing negative log-likelihood:

$$Q = -\sum_{n=1}^{N} \log \hat{p}_F(\mathbf{x}_n|y_n), \quad (11)$$

where $\log \hat{p}_F(\mathbf{x}_n|y_n)$ is defined by eq. (10). The model is trained using a gradient-based approach applied to the flow parameters stored in $\mathbf{f}_k$ functions.

4.3 Estimating Counterfactuals

For a trained conditional normalizing flow, the counterfactual explanation can be easily calculated simply by optimizing the criterion given by eq. (6). The parameters of the flow model are frozen, and $\mathbf{x}'$ is optimized using the gradient-based procedure, starting from the point $\mathbf{x}_0$. To enhance the efficiency of our method, we have incorporated batch processing capabilities, allowing for the simultaneous calculation of multiple counterfactual explanations. This is achieved by aggregating instances and employing an average aggregation for loss calculation. Such a feature is notably absent in the other approaches compared to this study, providing our method with a distinct computational advantage.

5 Experiments

In this section, we aim to demonstrate and validate our counterfactual explanation method through a series of experiments. Initially, we illustrate our method's intuition with the Moons dataset and Logistic Regression model. Next, we compare our approach against the only reference method in a probabilistically plausible CFs area - Artelt and Hammer [3], as well as other established CF methods. This comparison focuses on the impact of plausibility on proximity metrics and time efficiency. Lastly, we conduct broader comparisons using other classifier models: Logistic Regression (LR), Multilayer Perceptron (MLP), and Neural Oblivious Decision Ensembles (NODE) [22]. The code for these experiments is publicly released on GitHub[1].

Datasets To evaluate PPCEF's effectiveness, we conducted experiments on seven numerical-only tabular datasets. Four datasets (Law, Heloc, Moons, and Audit) represent binary classification problems, whereas the first two datasets (Law and Heloc) are commonly used benchmarks for counterfactual explanation tasks. The remaining three datasets (Blobs, Digits, and Wine) address multiclass classification problems. Detailed descriptions of these datasets are available in the Appendix B [33]. Overall, they represent broad diversity in sample sizes (up to approximately 10.000), number of variables (up to 64), and number of classes (up to 10). For preprocessing purposes, we implemented two key steps to prepare the datasets. First, we addressed class imbalance by downsampling the majority class to match the size of the minority class. Second, we normalized all features across the datasets to a $[0, 1]$ range, enabling consistent scale and comparability among features. Thirdly, to ensure robust method evaluation, we employed stratified 5-fold cross-validation on each dataset. Finally, for clarity, the main manuscript reports average values, while the appendix [33] includes standard deviation for detailed analysis.

[1] https://github.com/ofurman/counterfactuals

Table 1: Comparative Results of Probabilistically Plausible Counterfactual Explanation Methods. We contrast the performance of PPCEF method with Artelt & Hammer [3] and other methods across Logistic Regression (LR) classifier. The results demonstrate our method's consistently valid and probabilistically plausible results and its ability to produce counterfactuals even in complex scenarios like high-dimensional data.

Dataset	Method	Coverage ↑	Validity ↑	Prob. Plaus. ↑	LOF	IsoForest	Log Dens. ↑	L1 ↓	L2 ↓	Time ↓
Moons	CBCE	**1.00**	**1.00**	0.10	1.06	0.03	-5.81	0.62	0.48	**0.07** s
	CEGP	**1.00**	**1.00**	0.09	1.36	0.00	-6.66	0.36	**0.28**	904.11 s
	CEM	**1.00**	**1.00**	0.14	2.03	-0.07	-10.09	0.55	0.50	211.56 s
	WACH	0.98	**1.00**	0.11	1.55	-0.01	-6.34	0.49	0.36	198.29 s
	ARTELT	**1.00**	**1.00**	0.08	1.53	-0.03	-8.74	**0.32**	0.32	4.15 s
	PPCEF	**1.00**	**1.00**	**1.00**	1.01	0.04	**1.69**	0.45	0.36	1.85 s
Law	CBCE	**1.00**	**1.00**	0.49	1.05	0.04	1.28	0.61	0.40	**0.23** s
	CEGP	**1.00**	**1.00**	0.49	1.07	0.04	1.08	0.23	0.18	1973.76 s
	CEM	**1.00**	**1.00**	0.26	1.26	-0.02	-0.56	0.33	0.31	368.10 s
	WACH	**1.00**	**1.00**	0.39	1.30	-0.01	-0.29	0.45	0.35	359.00 s
	ARTELT	**1.00**	**1.00**	0.40	1.12	0.02	0.54	**0.20**	**0.20**	4.02 s
	PPCEF	**1.00**	**1.00**	**1.00**	1.03	0.07	**2.05**	0.37	0.23	2.42 s
Audit	CBCE	**1.00**	**1.00**	0.79	11.70	0.14	**54.97**	2.55	1.24	**0.04** s
	CEGP	0.97	**1.00**	0.02	$6.08{\cdot}10^{7}$	0.02	8.09	1.56	0.57	561.04 s
	CEM	0.52	**1.00**	0.00	$8.28{\cdot}10^{6}$	-0.04	20.84	1.20	**0.37**	105.92 s
	WACH	**0.99**	**1.00**	0.02	$1.42{\cdot}10^{8}$	0.06	-40.34	1.78	0.80	101.27 s
	ARTELT	0.60	0.97	0.00	$4.09{\cdot}10^{8}$	0.10	-3585.76	**0.90**	0.88	43.84 s
	PPCEF	**1.00**	**0.99**	**0.99**	$4.25{\cdot}10^{7}$	0.08	51.64	2.04	0.79	7.01 s
Heloc	CBCE	**1.00**	**1.00**	0.54	1.10	0.07	28.01	2.84	0.82	**5.71** s
	CEGP	**1.00**	**1.00**	0.29	$3.50{\cdot}10^{7}$	0.04	24.75	**0.26**	**0.10**	9654.60 s
	CEM	**1.00**	**1.00**	0.07	$2.50{\cdot}10^{8}$	0.02	12.37	0.35	0.20	1639.16 s
	WACH	**1.00**	**1.00**	0.00	$2.65{\cdot}10^{8}$	0.03	-15.09	0.74	0.37	1600.28 s
	ARTELT	0.00	-	-	-	-	-	-	-	- s
	PPCEF	**1.00**	**1.00**	**1.00**	$6.47{\cdot}10^{7}$	0.07	**32.42**	0.90	0.23	12.44 s
Blobs	CBCE	**1.00**	**1.00**	0.27	1.02	0.03	-35.52	0.95	0.72	**0.13** s
	CEGP	**1.00**	**1.00**	0.00	2.43	-0.07	-9.08	**0.30**	**0.25**	1295.36 s
	CEM	0.96	**1.00**	0.00	3.51	-0.12	-14.95	0.46	0.45	512.56 s
	WACH	**1.00**	**1.00**	0.04	2.24	-0.06	-9.52	0.51	0.38	441.59 s
	ARTELT	**1.00**	**1.00**	0.00	2.11	-0.07	-3.51	0.39	0.33	6.62 s
	PPCEF	**1.00**	**1.00**	**1.00**	1.01	0.04	**3.00**	0.69	0.50	3.22 s
Digits	CBCE	**1.00**	**1.00**	0.18	1.02	0.04	23.72	16.28	3.09	**0.51** s
	CEGP	**1.00**	**1.00**	0.11	1.09	0.01	-0.39	2.53	**0.63**	1945.67 s
	CEM	**1.00**	0.98	0.01	1.23	-0.03	-86.77	5.28	1.38	852.05 s
	WACH	**1.00**	**1.00**	0.08	1.20	0.00	-34.97	**2.47**	1.20	651.00 s
	ARTELT	0.80	0.93	0.04	1.69	0.01	-54.72	3.30	2.43	238.28 s
	PPCEF	**1.00**	**1.00**	**1.00**	1.12	0.03	**44.42**	8.27	1.33	8.68 s
Wine	CBCE	**1.00**	**1.00**	0.37	1.06	0.05	2.13	3.38	1.12	**0.01** s
	CEGP	**1.00**	**1.00**	0.01	1.08	0.05	-0.15	0.82	0.32	191.09 s
	CEM	**1.00**	**1.00**	0.00	1.35	-0.02	-12.94	1.20	0.63	81.33 s
	WACH	**1.00**	**1.00**	0.01	1.27	0.00	-9.41	1.57	0.78	50.74 s
	ARTELT	**1.00**	0.97	0.01	1.33	0.02	-11.73	0.68	0.65	0.96 s
	PPCEF	**1.00**	**1.00**	**1.00**	1.01	0.09	**9.72**	1.65	0.53	2.03 s

Classification Models For the experiments, we include Logistic Regression (LR), 3-layer Multilayer Perceptron (MLP), and Neural Oblivious Decision Ensemble (NODE) catering to both linear and non-linear scenarios. LR aligns with linear assumptions prevalent in some baseline methods, MLP allows for the assessment of behaviors in non-linear model contexts, and NODE stands as an example of a complex ensemble of neural decision trees. This triple-model approach facilitates a thorough evaluation across varied model complexities. Crucially, all models are differentiable, which is essential in the context of our method.

Experiments Details For every combination of the classification model and dataset, we trained both the classification model and a Normalizing Flow as the density estimator, following the approach detailed in Section 4.2. We opted for the Masked Autoregressive Flow (MAF) architecture [19] as our choice for the Normalizing Flow. This decision was based on experimental findings indicating MAF's superior performance in accurately fitting data distributions. For a deeper analysis of these results, including in-depth model performance metrics like accuracy, please refer to the Appendix [33]. See Section C for a detailed exploration and Tab. 9 for specific performance figures. The final step involved generating counterfactual explanations for the entire set of test samples.

Reference Methods Our analysis includes several significant baselines, each selected for its relevance to the field. We first consider the method developed by Artelt and Hammer [3], notable for its focus on probabilistically plausible counterfactuals. Additionally, we evaluate the approach by Wachter et al. [31], widely recognized as a foundational baseline in counterfactual explanations research. To provide both endogenous and exogenous counterfactual explanations, we compare three methods: Case-Based Counterfactual Explainer (CBCE) [14], Contrastive Explanation Method (CEM) [5], and Counterfactual Explanations Guided by Prototypes (CEGP) [15].

Metrics Following related works, we chose a comprehensive set of metrics to assess the performance of counterfactual explanation methods. We include two success metrics: *coverage*, evaluating the method's ability to generate explanations across all instances, and *validity*, assessing the efficacy of counterfactuals in altering the model's decision. In terms of proximity, we measure the *L1* and *L2* distances to quantify the closeness between original instances and their counterfactuals. We evaluate plausibility using a combination of metrics. First of all, we measure the *Local Outlier Factor (LOF)* score, which, when significantly greater than 1, indicates an outlier, with values closer to 1 suggesting normalcy, highlighting anomalies through local density deviations. Secondly, we utilize *Isolation Forest*, which

assigns scores between -0.5 and 0.5, with values approaching -0.5 identifying anomalies due to the ease of isolation and scores above 0 indicating normal observations. We further access counterfactuals using *probabilistic plausibility* metric, the proportion of CFs meeting the criterion defined in Eq. 3c. Moreover, we calculate *log density*, which gauges the logarithmic probability density of counterfactuals under the target class, with higher values indicating greater plausibility. Finally, we calculate *time* metric representing time in seconds needed for the method to process the whole test dataset.

5.1 Method Intuition via Toy Example

In our illustrative example, we present the counterfactual generation process using the Moons dataset under a Logistic Regression model, as depicted in Figure 1. The initial observation is represented by a black dot, with intermediary observations during the optimization process (after every 150 iteration steps) shown as orange dots and the final counterfactual outcome marked by a red dot. The probability distributions are indicated by contour lines, with the filled red contour denoting the region exceeding the desired density threshold. The blue line illustrates the decision boundary of the classifier. This visualization effectively demonstrates how our method navigates toward the target classification and probabilistic plausibility regions, adjusting its trajectory to surpass the classifier's decision boundary by a predefined margin ϵ upon achieving the required density level.

5.2 Probabilistically Plausibile Counterfactual Explanations Methods Comparison

In this section, we conduct a focused comparison of our approach, PPCEF, against the method by Artelt and Hammer [3], which is the primary reference in the realm of probabilistically plausible counterfactual explanations. For that purpose, we utilize the datasets, metrics, and classifiers described in the previous section. The evaluation is centered on assessing and validating the accuracy of both methods in generating counterfactuals, their plausibility, and their proximity to original instances.

The results are presented in Tab. 1. Firstly, we can observe that our method always returns the results that are probabilistically plausible. That is not the case for Artelt's method, which struggles in high-dimensional datasets like Heloc (23 dimensions) or Digits (64 dimensions), doesn't support non-linear classifiers like MLPs, and wasn't able to consistently fulfill the probabilistic plausibility criterion. Secondly, in terms of distances, Artelt's method returns better results, which is expected due to the trade-off between distance and plausibility, i.e., the more plausible observations, the farther away they usually are. However, the results are not clearly worse, especially in terms of L2 distance, meaning PPCEF can balance both desired properties of counterfactuals. Thirdly, the log density values of the observations produced by PPCEF method are significantly better. Fourthly, our analysis using Local Outlier Factor (LOF) and Isolation Forest (IsoForest) metrics indicates that our methods generate inliers (except for Audit and Heloc, where almost all methods struggle to obtain reasonable values of LOF), whereas Artelt's method underperforms and can sometimes result in outliers. Fifthly, our method turned out to be significantly faster, with the speed up around x2-10 on relatively small datasets. Finally, our method was almost always able to produce valid counterfactual explanations for MLP and NODE, contrary to Artelt (see results in Tab. 2 and detailed results in Tab. 6 and 7 in Appendix [33]). It's worth mentioning that PPCEF almost always returned probabilistically plausible observations, which, in case of non-valid observations, might still be valuable insight for the final user, contrary to the lack of a response at all.

5.3 Counterfactual Explanations Methods Comparison

In this comparative analysis, we evaluate our method against well-established reference methods, with a particular focus on the impact of integrating probabilistically plausible conditions into the optimization process. Our primary objective is to assess our method's performance in terms of validity, plausibility, proximity metrics, and processing efficiency. We also explore whether methods not specifically designed for plausibility can still produce plausible counterfactuals across various classifiers such as Logistic Regression (LR), Multilayer Perceptron (MLP), and Neural Oblivious Decision Ensembles (NODE).

Results presented in Tab. 1 and Tab. 2 indicate that our model excels in validity, plausibility (considering both probabilistic formulation and outlier metrics), and processing times while maintaining reasonable distances compared to competing approaches across all datasets and classification methods. Specifically, Tab. 2 presents the evaluation results for two selected high-dimensional datasets (one for binary classification problem and one for multiclass problem) using two advanced classifiers, demonstrating that our method consistently produces valid results not only with a shallow model, such as LR but also with deeper models, including MLP and NODE. In contrast, the majority of existing methods encounter difficulties in producing valid counterfactual explanations for the Multilayer Perceptron. We conducted a comprehensive evaluation utilizing all methods and datasets mentioned earlier, applying three different classifiers. Detailed outcomes are presented in Appendix A [33]. Particularly, results for Logistic Regression are shown in Tab. 5, while findings for the MLP and NODE classifiers are detailed in Tab. 6 and 7, respectively.

Furthermore, our hypothesis that reference methods could inadvertently yield plausible outcomes without targeted optimization was not confirmed. In terms of proximity, CEGP achieves the most favorable outcomes, with our method typically ranking closely behind. This demonstrates our method's effectiveness in balancing proximity and plausibility constraints. Notably, our method's computational time efficiency closely parallels the CBCE method, which does not involve an optimization process. This efficiency is due to our batching strategy, which processes all datasets collectively, as opposed to the case-by-case optimization typical of other methods. Summarizing, our method generates probabilistically plausible counterfactuals with exceptional efficiency and minimal compromise on proximity. Its ability to process high-dimensional data quickly makes it ideal for resource-constrained, real-world applications.

6 Method Analysis

In this section, we delve into the analysis of two pivotal components of our proposed method: the loss function and the regularization hyperparameter λ. Adhering to the experimental framework established in the earlier sections, these studies are conducted specifically using the Logistic Regression model. Our focus is on evaluating the impact of these elements on the method's overall performance and efficacy.

6.1 Loss Function Ablation Study

In this ablation study, we examined the influence of discriminative loss function selection on the effectiveness of our proposed method. While Binary Cross Entropy (BCE) and Cross Entropy (CE) losses are conventional choices for binary and multiclass problems, respectively,

Table 2: Analysis of Counterfactual Methods Across Classification Models. We offer a detailed comparison of our method and other well-established reference methods across two classification models: a 3-layer Multilayer Perceptron (MLP), and a Neural Oblivious Decision Ensemble (NODE). The results emphasize the efficacy of our method in producing valid and plausible counterfactuals across various models, including those that are deeper and more complex.

Dataset	Method	Cov. ↑	Val. ↑	Prob. Plaus. ↑	LOF	IsoForest	Log Dens. ↑	L1 ↓	L2 ↓	Time ↓
					MLP					
Heloc	CBCE	**1.00**	**0.94**	0.54	1.09	0.08	28.85	2.87	0.82	**6.47** s
	CEGP	0.94	0.63	0.05	$4.15{\cdot}10^{8}$	0.01	-3.28	1.25	0.43	31309.33 s
	CEM	**1.00**	0.86	0.01	$7.71{\cdot}10^{8}$	-0.01	-89.39	1.32	0.58	6938.45 s
	WACH	**0.99**	0.81	0.00	$1.34{\cdot}10^{8}$	-0.06	-161.68	3.11	0.90	23392.40 s
	ARTELT	-	-	-	-	-	-	-	-	- s
	PPCEF	**1.00**	0.92	**1.00**	$1.42{\cdot}10^{8}$	0.07	**32.07**	**1.18**	**0.31**	25.32 s
Digits	CBCE	**1.00**	**1.00**	0.18	1.02	0.04	23.66	16.29	3.09	**0.54** s
	CEGP	0.95	0.46	0.02	1.24	-0.02	-138.62	6.39	**1.42**	2523.28 s
	CEM	**1.00**	0.42	0.01	1.44	-0.06	-481.57	**6.34**	1.76	1260.54 s
	WACH	**1.00**	0.72	0.00	1.50	-0.07	-516.44	11.04	2.13	3342.38 s
	ARTELT	-	-	-	-	-	-	-	-	- s
	PPCEF	**1.00**	**1.00**	**0.98**	1.13	0.03	**43.87**	8.78	**1.42**	25.09 s
					NODE					
Heloc	CBCE	**1.00**	**1.00**	0.55	1.09	0.08	28.88	2.85	0.82	**17.53** s
	CEM	0.94	**1.00**	0.10	1.35	0.05	9.00	**0.47**	0.29	14772.66 s
	WACH	0.96	**1.00**	0.10	$2.12{\cdot}10^{8}$	0.05	10.75	0.85	0.36	37254.33 s
	ARTELT	-	-	-	-	-	-	-	-	- s
	PPCEF	**1.00**	0.94	**1.00**	1.08	0.09	**31.85**	1.02	**0.28**	126.05 s
Digits	CBCE	**1.00**	**1.00**	0.18	1.02	0.04	24.00	16.27	3.09	**3.12** s
	CEM	**1.00**	**1.00**	0.03	1.32	-0.02	-39.458	4.07	1.44	5451.835 s
	WACH	**1.00**	**1.00**	0.16	1.12	0.02	7.02	**2.93**	**1.13**	15376.44 s
	ARTELT	-	-	-	-	-	-	-	-	- s
	PPCEF	**1.00**	**1.00**	**1.00**	1.15	0.02	**43.97**	7.76	1.36	69.45 s

Table 3: Ablation Study on Loss Function Selection.

Dataset	Loss	Cov.	Val.	PP	L1	L2	LD
Moons	Ours	**1.00**	**1.00**	**1.00**	**0.45**	**0.36**	1.69
	BCE	**1.00**	**1.00**	**0.99**	0.89	0.69	**1.74**
Law	Ours	**1.00**	**1.00**	**1.00**	**0.37**	**0.23**	**2.05**
	BCE	**1.00**	**1.00**	**0.98**	0.97	0.60	1.67
Audit	Ours	**1.00**	**0.99**	**0.99**	**2.04**	**0.79**	51.64
	BCE	**1.00**	**0.99**	**0.98**	3.01	1.25	**52.54**
Heloc	Ours	**1.00**	**0.99**	**0.99**	**0.85**	**0.23**	**37.50**
	BCE	**1.00**	**0.97**	**0.99**	1.91	0.54	34.50
Blobs	Ours	**1.00**	**1.00**	**1.00**	**0.69**	**0.50**	**3.00**
	CE	**1.00**	**1.00**	0.93	0.82	0.60	2.85
Digits	Ours	**1.00**	**1.00**	**1.00**	**8.27**	**1.33**	**44.42**
	CE	**1.00**	**1.00**	**1.00**	12.67	2.13	44.18
Wine	Ours	**1.00**	**1.00**	**1.00**	**1.65**	**0.53**	**9.72**
	CE	**1.00**	**1.00**	**0.99**	3.87	1.29	9.29

we compared them against our proposed discriminative loss function to understand their impacts on the results. The findings, detailed in Tab. 3, reveal a notable distinction in distance metrics. Our method, using the specialized loss function, demonstrated significantly better proximity to original observations compared to BCE and CE. This improvement is attributed to our loss function's design, which zeroes the classification component of the loss upon surpassing by ϵ a classification threshold. This allows for more rapid convergence to closer counterfactuals, while CE, by continually seeking points with higher classification confidence, tends to push counterfactuals further from the original samples. Consequently, this affects the final values in proximity metrics, underscoring the advantage of our approach in generating more proximate and plausible counterfactuals.

6.2 Regularization Hyperparameter λ Analysis

To evaluate the impact of the regularization hyperparameter λ on the fulfillment of validity and probabilistic plausibility conditions, we conducted a focused hyperparameter sensitivity analysis. While λ theoretically should extend to infinity, practical considerations necessitate setting a feasible value. Our objective is to identify an optimal λ that not only guarantees condition fulfillment but also to understand its influence on other metrics. Experiments were carried out on the Moons and Law datasets, exploring λ values within the set $\{1, 2, 5, 10, 100, 1000\}$. The results in Tab. 4 indicate that moderate values of λ, like 5 or 10, deliver satisfactory outcomes, while values around 100 or more almost invariably guarantee the fulfillment of the conditions, leading us to adopt the value of 100 for all preceding experiments. This experiment confirms the expected trade-off: higher strictness in counterfactual conditions leads to decreased proximity metrics, requiring larger deviations from the original data point.

Table 4: Ablation Study on Regularization Hyperparameter λ.

Dataset	λ	Cov.	Val.	PP	L1	L2	LD
Moons	1	1.00	0.46	0.78	0.43	0.34	1.61
	2	1.00	0.95	0.92	0.43	0.34	1.63
	5	1.00	0.99	0.98	0.43	0.34	1.66
	10	1.00	0.99	1.00	0.44	0.35	1.70
	100	1.00	1.00	1.00	0.45	0.36	1.70
	1000	1.00	1.00	1.00	0.45	0.36	1.70
Law	1	1.00	0.48	0.98	0.19	0.12	1.85
	2	1.00	0.99	0.99	0.28	0.18	1.88
	5	1.00	1.00	1.00	0.29	0.18	1.94
	10	1.00	1.00	1.00	0.30	0.18	2.00
	100	1.00	1.00	1.00	0.34	0.21	2.08
	1000	1.00	1.00	1.00	0.38	0.22	2.09

7 Conclusions

In this work, we present PPCEF, a novel method for generating counterfactual explanations that utilize normalizing flows as density estimators within an unconstrained optimization framework. This technique adeptly balances essential factors such as distance, validity, and probabilistic plausibility in the counterfactuals it produces. Notably, PPCEF is computationally efficient and capable of handling large datasets, making it highly applicable in real-world scenarios. The method's flexible design allows for future enhancements, including other desirable counterfactual attributes like actionability or sparsity, and to generate plausible counterfactuals in label-scarce environments.

Acknowledgements

Patryk Wielopolski, Oleksii Furman, and Maciej Zieba's work was supported by the National Science Centre (Poland) Grant No. 2021/43/B/ST6/02853, and Jerzy Stefanowski's work was supported by the National Science Centre (Poland) grant No. 2023/51/B/ST6/00545. Moreover, we gratefully acknowledge Polish high-performance computing infrastructure PLGrid (HPC Center: ACK Cyfronet AGH) for providing computer facilities and support within computational grant no. PLG/2023/016636.

References

[1] A. B. Arrieta and J. D. Ser. Plausible counterfactuals: Auditing deep learning classifiers with realistic adversarial examples. In *2020 International Joint Conference on Neural Networks, IJCNN 2020, Glasgow, United Kingdom, July 19-24, 2020*, pages 1–7. IEEE, 2020.

[2] A. Artelt and B. Hammer. On the computation of counterfactual explanations - A survey. *CoRR*, abs/1911.07749, 2019.

[3] A. Artelt and B. Hammer. Convex density constraints for computing plausible counterfactual explanations. In *Artificial Neural Networks and Machine Learning - ICANN 2020 - 29th International Conference on Artificial Neural Networks, Bratislava, Slovakia, September 15-18, 2020, Proceedings, Part I*, volume 12396 of *Lecture Notes in Computer Science*, pages 353–365. Springer, 2020.

[4] M. Augustin, V. Boreiko, F. Croce, and M. Hein. Diffusion visual counterfactual explanations. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

[5] A. Dhurandhar, P. Chen, R. Luss, C. Tu, P. Ting, K. Shanmugam, and P. Das. Explanations based on the missing: Towards contrastive explanations with pertinent negatives. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 590–601, 2018.

[6] L. Dinh, D. Krueger, and Y. Bengio. NICE: non-linear independent components estimation. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings*, 2015.

[7] L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using real NVP. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[8] A. Dombrowski, J. E. Gerken, K. Müller, and P. Kessel. Diffeomorphic counterfactuals with generative models. *CoRR*, abs/2206.05075, 2022. doi: 10.48550/ARXIV.2206.05075.

[9] T. D. Duong, Q. Li, and G. Xu. Ceflow: A robust and efficient counterfactual explanation framework for tabular data using normalizing flows. In *Advances in Knowledge Discovery and Data Mining - 27th Pacific-Asia Conference on Knowledge Discovery and Data Mining, PAKDD 2023, Osaka, Japan, May 25-28, 2023, Proceedings, Part II*, volume 13936 of *Lecture Notes in Computer Science*, pages 133–144. Springer, 2023.

[10] R. Guidotti. Counterfactual explanations and how to find them: literature review and benchmarking. *Data Mining and Knowledge Discovery*, pages 1–55, 04 2022.

[11] G. Jeanneret, L. Simon, and F. Jurie. Diffusion models for counterfactual explanations. In *Computer Vision - ACCV 2022 - 16th Asian Conference on Computer Vision, Macao, China, December 4-8, 2022, Proceedings, Part VII*, volume 13847 of *Lecture Notes in Computer Science*, pages 219–237. Springer, 2022.

[12] J. Jiang, J. Lan, F. Leofante, A. Rago, and F. Toni. Provably robust and plausible counterfactual explanations for neural networks via robust optimisation. *CoRR*, abs/2309.12545, 2023.

[13] K. Kanamori, T. Takagi, K. Kobayashi, and H. Arimura. DACE: distribution-aware counterfactual explanation by mixed-integer linear optimization. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 2855–2862. ijcai.org, 2020.

[14] M. T. Keane and B. Smyth. Good counterfactuals and where to find them: A case-based technique for generating counterfactuals for explainable AI (XAI). In *Case-Based Reasoning Research and Development - 28th International Conference, ICCBR 2020, Salamanca, Spain, June 8-12, 2020, Proceedings*, volume 12311 of *Lecture Notes in Computer Science*, pages 163–178. Springer, 2020.

[15] A. V. Looveren and J. Klaise. Interpretable counterfactual explanations guided by prototypes. In *Machine Learning and Knowledge Discovery in Databases. Research Track - European Conference, ECML PKDD 2021, Bilbao, Spain, September 13-17, 2021, Proceedings, Part II*, volume 12976 of *Lecture Notes in Computer Science*, pages 650–665. Springer, 2021.

[16] D. Mahajan, C. Tan, and A. Sharma. Preserving causal constraints in counterfactual explanations for machine learning classifiers. *CoRR*, abs/1912.03277, 2019.

[17] S. Mertes, T. Huber, K. Weitz, A. Heimerl, and E. André. Ganterfactual—counterfactual explanations for medical non-experts using generative adversarial learning. *Frontiers in Artificial Intelligence*, 2022.

[18] R. K. Mothilal, A. Sharma, and C. Tan. Explaining machine learning classifiers through diverse counterfactual explanations. In *FAT* '20: Conference on Fairness, Accountability, and Transparency, Barcelona, Spain, January 27-30, 2020*, pages 607–617. ACM, 2020.

[19] G. Papamakarios, I. Murray, and T. Pavlakou. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 2338–2347, 2017.

[20] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[21] J. Pearl, M. Glymour, and N. Jewell. *Causal Inference in Statistics: A Primer*. Wiley, 2016. ISBN 9781119186847.

[22] S. Popov, S. Morozov, and A. Babenko. Neural oblivious decision ensembles for deep learning on tabular data. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[23] R. Poyiadzi, K. Sokol, R. Santos-Rodríguez, T. D. Bie, and P. A. Flach. FACE: feasible and actionable counterfactual explanations. In *AIES '20: AAAI/ACM Conference on AI, Ethics, and Society, New York, NY, USA, February 7-8, 2020*, pages 344–350. ACM, 2020.

[24] D. Rezende and S. Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pages 1530–1538. PMLR, 2015.

[25] C. Russell. Efficient search for diverse coherent explanations. In danah boyd and J. H. Morgenstern, editors, *Proceedings of the Conference on Fairness, Accountability, and Transparency, FAT* 2019, Atlanta, GA, USA, January 29-31, 2019*, pages 20–28. ACM, 2019.

[26] G. Shakhnarovich, T. Darrell, and P. Indyk. Nearest-neighbor methods in learning and vision. *IEEE Trans. Neural Networks*, 19(2):377, 2008.

[27] B. Smyth and M. T. Keane. A few good counterfactuals: Generating interpretable, plausible and diverse counterfactual explanations. In *Case-Based Reasoning Research and Development - 30th International Conference, ICCBR 2022, Nancy, France, September 12-15, 2022, Proceedings*, volume 13405 of *Lecture Notes in Computer Science*, pages 18–32. Springer, 2022.

[28] G. Van Rossum and F. L. Drake Jr. *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.

[29] N. Vercheval and A. Pizurica. Hierarchical variational autoencoders for visual counterfactuals. In *2021 IEEE International Conference on Image Processing, ICIP 2021, Anchorage, AK, USA, September 19-22, 2021*, pages 2513–2517. IEEE, 2021.

[30] S. Verma, V. Boonsanong, M. Hoang, K. E. Hines, J. P. Dickerson, and C. Shah. Counterfactual explanations and algorithmic recourses for machine learning: A review. *arXiv preprint arXiv:2010.10596*, 2020.

[31] S. Wachter, B. D. Mittelstadt, and C. Russell. Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *CoRR*, abs/1711.00399, 2017.

[32] G. P. Wellawatte, A. Seshadri, and A. D. White. Model agnostic generation of counterfactual explanations for molecules. *Chem. Sci.*, 2022.

[33] P. Wielopolski, O. Furman, J. Stefanowski, and M. Zieba. Probabilistically plausible counterfactual explanations with normalizing flows. *CoRR*, abs/2405.17640, 2024. doi: 10.48550/ARXIV.2405.17640.

[34] L. F. Wightman. Lsac national longitudinal bar passage study. lsac research report series. Technical report, Law School Admission Council, Newtown, PA., 1998.

A Additional Results

This section supplements the main manuscript's experimental results with a more comprehensive analysis. We include means and standard deviations from five-fold cross-validation for greater statistical rigor and broaden the comparison with additional datasets and metrics. Tab. 5 compares the effectiveness and nuances of different approaches in generating counterfactual explanations for the Logistic Regression model. In Tab. 6, we delve into Multilayer Perceptron, presenting a similar analysis that highlights the unique aspects and performance metrics relevant to this model. Moving forward, we examine the performance of counterfactual methods for deep ensembles of oblivious differentiable decision trees. Results for NODE classifier are presented in Tab. 7. Notably, Artelt's method is incompatible with non-linear classifiers such as Multilayer Perceptrons (MLP) or NODE, precluding the acquisition of performance data for these models using this approach. Furthermore, the application of CEGP to the NODE classifier was prohibitively time-consuming, which prevented the generation of results for this method as well. In Tab. 8, we provide extended results for the Ablation Study on Loss Function with additional metrics: LOF and Isolation Forest.

Table 5: Detailed Comparative Results of Probabilistically Plausible Counterfactual Explanation Methods for **Logistic Regression** classifier. We present a comprehensive comparison of our method with other established reference methods across various datasets. The results presented include the mean and standard deviation obtained from a five-fold cross-validation.

Dataset	Method	Coverage ↑	Validity ↑	Prob. Plaus. ↑	LOF	IsoForest	Log Dens. ↑	L1 ↓	L2 ↓	Time ↓
Moons	CBCE	**1.00±0.00**	**1.00±0.00**	0.10±0.23	1.06±0.02	0.03±0.00	-5.81±3.74	0.62±0.07	0.48±0.05	**0.07±0.01** s
	CEGP	**1.00±0.00**	**1.00±0.00**	0.09±0.04	1.36±0.03	0.00±0.00	-6.66±0.82	0.36±0.02	**0.28±0.01**	904.11±11.12 s
	CEM	**1.00±0.00**	**1.00±0.00**	0.14±0.03	2.03±0.12	-0.07±0.01	-10.09±6.62	0.55±0.03	0.50±0.02	211.56±1.50 s
	WACH	0.98±0.03	**1.00±0.01**	0.11±0.05	1.55±0.08	-0.01±0.00	-6.34±2.41	0.49±0.02	0.36±0.01	198.29±3.66 s
	ARTELT	**1.00±0.00**	**1.00±0.00**	0.08±0.03	1.53±0.09	-0.03±0.01	-8.74±3.57	**0.32±0.02**	0.32±0.02	4.15±0.69 s
	PPCEF	**1.00±0.00**	**1.00±0.00**	**1.00±0.00**	1.01±0.02	0.04±0.01	**1.69±0.07**	0.45±0.01	0.36±0.01	1.85±0.01 s
Law	CBCE	**1.00±0.00**	**1.00±0.00**	0.49±0.35	1.05±0.02	0.04±0.02	1.28±0.41	0.61±0.03	0.40±0.02	**0.23±0.00** s
	CEGP	**1.00±0.00**	**1.00±0.00**	0.49±0.04	1.07±0.00	0.04±0.00	1.08±0.04	0.23±0.01	**0.18±0.01**	1973.76±11.09 s
	CEM	**1.00±0.00**	**1.00±0.00**	0.26±0.01	1.26±0.00	-0.02±0.00	-0.56±0.08	0.33±0.01	0.31±0.01	368.10±51.57 s
	WACH	**1.00±0.00**	**1.00±0.00**	0.39±0.03	1.30±0.02	-0.01±0.00	-0.29±0.13	0.45±0.01	0.35±0.01	359.00±41.37 s
	ARTELT	**1.00±0.00**	**1.00±0.00**	0.40±0.02	1.12±0.01	0.02±0.00	0.54±0.08	**0.20±0.01**	0.20±0.01	4.02±0.42 s
	PPCEF	**1.00±0.00**	**1.00±0.00**	**1.00±0.00**	1.03±0.00	0.07±0.00	**2.05±0.02**	0.37±0.01	0.23±0.01	2.42±0.10 s
Audit	CBCE	**1.00±0.00**	**1.00±0.00**	0.79±0.28	11.70±20.10	0.14±0.00	**54.97±3.89**	2.55±0.18	1.24±0.10	**0.04±0.01** s
	CEGP	0.97±0.02	**1.00±0.00**	0.02±0.03	$6.08{\cdot}10^{7}$±$5.58{\cdot}10^{7}$	0.02±0.02	8.09±13.27	1.56±0.13	**0.57±0.06**	561.04±13.04 s
	CEM	0.52±0.03	**1.00±0.00**	0.00±0.01	$8.28{\cdot}10^{6}$±$1.85{\cdot}10^{7}$	-0.04±0.03	20.84±10.32	1.20±0.06	0.37±0.02	105.92±13.20 s
	WACH	**0.99±0.01**	**1.00±0.00**	0.02±0.02	$1.42{\cdot}10^{8}$±$2.99{\cdot}10^{7}$	0.06±0.01	-40.34±34.83	1.78±0.08	0.80±0.05	101.27±10.95 s
	ARTELT	0.60±0.22	0.97±0.05	0.00±0.01	$4.09{\cdot}10^{8}$±$5.89{\cdot}10^{8}$	0.10±0.02	-3585.76±7834.29	**0.90±0.14**	0.88±0.10	43.84±31.22 s
	PPCEF	**1.00±0.00**	**0.99±0.01**	**0.99±0.01**	$4.25{\cdot}10^{7}$±$9.32{\cdot}10^{7}$	0.08±0.01	51.64±4.53	2.04±0.15	0.79±0.12	7.01±1.08 s
Heloc	CBCE	**1.00±0.00**	**1.00±0.00**	0.54±0.01	1.10±0.08	0.07±0.03	28.01±3.31	2.84±0.39	0.82±0.11	**5.71±0.41** s
	CEGP	**1.00±0.00**	**1.00±0.00**	0.29±0.03	$3.50{\cdot}10^{7}$±$7.28{\cdot}10^{6}$	0.04±0.00	24.75±0.52	**0.26±0.03**	**0.10±0.01**	9654.60±81.96 s
	CEM	**1.00±0.00**	**1.00±0.00**	0.07±0.01	$2.50{\cdot}10^{8}$±$4.00{\cdot}10^{7}$	0.02±0.01	12.37±2.74	0.35±0.05	0.20±0.02	1639.16±9.35 s
	WACH	**1.00±0.00**	**1.00±0.00**	0.00±0.00	$2.65{\cdot}10^{8}$±$3.04{\cdot}10^{7}$	0.03±0.01	-15.09±5.86	0.74±0.06	0.37±0.01	1600.28±17.36 s
	ARTELT	-	-	-	-	-	-	-	-	-
	PPCEF	**1.00±0.00**	**1.00±0.00**	**1.00±0.00**	$6.47{\cdot}10^{7}$±$2.16{\cdot}10^{7}$	0.07±0.00	**32.42±0.34**	0.90±0.03	0.23±0.01	12.44±2.36 s
Blobs	CBCE	**1.00±0.00**	**1.00±0.00**	0.27±0.15	1.02±0.03	0.03±0.02	-35.52±15.68	0.95±0.01	0.72±0.01	**0.13±0.00** s
	CEGP	**1.00±0.00**	**1.00±0.00**	0.00±0.00	2.43±0.12	-0.07±0.01	-9.08±2.29	**0.30±0.01**	**0.25±0.01**	1295.36±30.32 s
	CEM	0.96±0.02	**1.00±0.00**	0.00±0.00	3.51±0.09	-0.12±0.00	-14.95±2.91	0.46±0.04	0.45±0.03	512.56±5.56 s
	WACH	**1.00±0.00**	**1.00±0.00**	0.04±0.02	2.24±0.19	-0.06±0.01	-9.52±1.80	0.51±0.02	0.38±0.02	441.59±15.39 s
	ARTELT	**1.00±0.00**	**1.00±0.00**	0.00±0.00	2.11±0.16	-0.07±0.01	-3.51±0.33	0.39±0.01	0.33±0.01	6.62±0.09 s
	PPCEF	**1.00±0.00**	**1.00±0.00**	**1.00±0.00**	1.01±0.01	0.04±0.01	**3.00±0.11**	0.69±0.05	0.50±0.04	3.22±0.84 s
Digits	CBCE	**1.00±0.00**	**1.00±0.00**	0.18±0.04	1.02±0.06	0.04±0.00	23.72±4.73	16.28±0.62	3.09±0.02	**0.51±0.16** s
	CEGP	**1.00±0.00**	**1.00±0.00**	0.11±0.03	1.09±0.01	0.01±0.00	-0.39±4.80	2.53±0.11	**0.63±0.02**	1945.67±22.30 s
	CEM	**1.00±0.00**	0.98±0.04	0.01±0.00	1.23±0.01	-0.03±0.01	-86.77±17.24	5.28±4.93	1.38±0.75	852.05±27.68 s
	WACH	**1.00±0.00**	**1.00±0.00**	0.08±0.03	1.20±0.01	0.00±0.01	-34.97±7.01	**2.47±0.05**	1.20±0.02	651.00±7.97 s
	ARTELT	0.80±0.12	0.93±0.02	0.04±0.03	1.69±0.05	0.01±0.01	-54.72±11.81	3.30±0.13	2.43±0.14	238.28±33.28 s
	PPCEF	**1.00±0.00**	**1.00±0.00**	**1.00±0.00**	1.12±0.01	0.03±0.01	**44.42±1.87**	8.27±0.24	1.33±0.04	8.68±3.65 s
Wine	CBCE	**1.00±0.00**	**1.00±0.00**	0.37±0.05	1.06±0.05	0.05±0.02	2.13±1.28	3.38±0.14	1.12±0.05	**0.01±0.00** s
	CEGP	**1.00±0.00**	**1.00±0.00**	0.01±0.01	1.08±0.03	0.05±0.01	-0.15±1.54	0.82±0.08	**0.32±0.03**	191.09±4.00 s
	CEM	**1.00±0.00**	**1.00±0.00**	0.00±0.00	1.35±0.03	-0.02±0.01	-12.94±1.94	1.20±0.09	0.63±0.05	81.33±1.88 s
	WACH	**1.00±0.00**	**1.00±0.00**	0.01±0.01	1.27±0.04	0.00±0.01	-9.41±1.95	1.57±0.10	0.78±0.05	50.74±5.63 s
	ARTELT	**1.00±0.00**	0.97±0.04	0.01±0.02	1.33±0.03	0.02±0.00	-11.73±2.36	**0.65±0.05**	0.96±0.08 s	
	PPCEF	**1.00±0.00**	**1.00±0.00**	**1.00±0.00**	1.01±0.01	0.09±0.01	**9.72±0.62**	1.65±0.09	0.53±0.04	2.03±0.47 s

Table 6: Detailed Comparative Results of Probabilistically Plausible Counterfactual Explanation Methods for **Multilayer Perceptron** classifier. We provide an in-depth comparison between our approach and other well-established methods, utilizing a range of datasets. This analysis includes both the average values and the standard deviations derived from five-fold cross-validation.

Dataset	Method	Coverage ↑	Validity ↑	Prob. Plaus. ↑	LOF	IsoForest	Log Dens. ↑	L1 ↓	L2 ↓	Time ↓
Moons	CBCE	**1.00±0.00**	0.84±0.24	0.58±0.16	1.03±0.03	0.02±0.01	1.23±0.27	0.71±0.16	0.53±0.11	**0.08±0.00** s
	CEGP	0.97±0.03	0.33±0.13	0.06±0.07	1.39±0.10	0.00±0.01	-3.67±1.16	0.29±0.07	**0.23±0.05**	1562.34±119.65 s
	CEM	0.99±0.02	0.45±0.15	0.03±0.04	2.29±0.16	-0.08±0.01	-11.92±6.48	0.49±0.04	0.48±0.04	784.34±33.83 s
	WACH	1.00±0.01	0.56±0.06	0.04±0.05	1.52±0.12	-0.01±0.01	-3.71±2.54	**0.28±0.09**	0.24±0.08	1452.01±99.08 s
	ARTELT	-	-	-	-	-	-	-	-	- s
	PPCEF	**1.00±0.00**	**0.98±0.01**	**1.00±0.00**	0.99±0.02	0.03±0.01	**1.62±0.04**	0.44±0.05	0.34±0.04	20.44±1.75 s
Law	CBCE	**1.00±0.00**	0.79±0.11	0.36±0.40	1.05±0.02	0.03±0.03	1.18±0.45	0.67±0.08	0.44±0.05	**0.28±0.02** s
	CEGP	0.93±0.02	0.28±0.01	0.53±0.05	1.07±0.00	0.04±0.00	1.23±0.16	**0.24±0.02**	**0.17±0.01**	2986.44±60.71 s
	CEM	**1.00±0.00**	0.61±0.04	0.27±0.01	1.26±0.01	-0.02±0.00	-0.22±0.11	0.29±0.01	0.28±0.01	1413.03±97.23 s
	WACH	**1.00±0.00**	0.74±0.04	0.42±0.03	1.14±0.01	0.01±0.00	0.58±0.09	0.38±0.01	0.29±0.01	2459.88±78.30 s
	ARTELT	-	-	-	-	-	-	-	-	- s
	PPCEF	**1.00±0.00**	**0.95±0.01**	**1.00±0.00**	1.03±0.00	0.07±0.00	**2.04±0.02**	0.40±0.01	0.24±0.01	20.63±1.08 s
Audit	CBCE	**1.00±0.00**	0.93±0.07	0.61±0.21	11.10±13.30	0.14±0.00	55.02±3.49	2.77±0.25	1.34±0.14	**0.16±0.01** s
	CEGP	**0.55±0.02**	0.48±0.02	0.02±0.03	$1.45{\cdot}10^{2}$±$3.86{\cdot}10^{1}$	0.00±0.03	-85.64±41.77	1.42±0.12	**0.57±0.03**	1426.33±55.75 s
	CEM	0.53±0.02	0.54±0.05	0.01±0.01	$6.52{\cdot}10^{7}$±$8.76{\cdot}10^{7}$	0.04±0.03	-132.43±98.13	**1.19±0.14**	0.61±0.04	441.85±7.56 s
	WACH	0.49±0.02	0.78±0.06	0.00±0.00	$1.24{\cdot}10^{2}$±$2.75{\cdot}10^{1}$	-0.03±0.03	-155.20±74.93	1.49±0.07	0.65±0.01	601.77±23.34 s
	ARTELT	-	-	-	-	-	-	-	-	- s
	PPCEF	**1.00±0.00**	**0.99±0.02**	**0.99±0.01**	$1.44{\cdot}10^{7}$±$1.88{\cdot}10^{7}$	0.08±0.01	51.72±4.59	2.14±0.11	0.83±0.10	46.60±3.82 s
Heloc	CBCE	**1.00±0.00**	**0.94±0.01**	0.54±0.03	1.09±0.08	0.08±0.03	28.85±3.95	2.87±0.40	0.82±0.11	**6.47±0.69** s
	CEGP	0.94±0.01	0.63±0.01	0.05±0.01	$4.15{\cdot}10^{8}$±$1.97{\cdot}10^{8}$	0.01±0.00	-3.28±4.79	1.25±0.11	0.43±0.03	31309.33±3342.40 s
	CEM	**1.00±0.00**	0.86±0.01	0.01±0.00	$7.71{\cdot}10^{8}$±$1.41{\cdot}10^{8}$	-0.01±0.01	-89.39±56.89	1.32±0.22	0.58±0.07	6938.45±79.08 s
	WACH	0.99±0.00	0.81±0.03	0.00±0.00	$1.34{\cdot}10^{8}$±$4.44{\cdot}10^{7}$	-0.06±0.01	-161.68±104.43	3.11±0.24	0.90±0.04	23392.40±3677.14 s
	ARTELT	-	-	-	-	-	-	-	-	- s
	PPCEF	**1.00±0.00**	0.92±0.03	**1.00±0.00**	$1.42{\cdot}10^{8}$±$3.90{\cdot}10^{7}$	0.07±0.00	**32.07±0.63**	**1.18±0.11**	**0.31±0.03**	25.32±6.41 s
Blobs	CBCE	**1.00±0.00**	**1.00±0.00**	0.27±0.15	1.02±0.03	0.03±0.02	-35.45±15.48	0.95±0.01	0.72±0.01	**0.16±0.00** s
	CEGP	0.98±0.02	0.47±0.04	0.00±0.00	2.48±0.16	-0.07±0.00	-9.54±3.30	0.35±0.01	**0.29±0.01**	3047.79±38.09 s
	CEM	0.97±0.04	0.71±0.07	0.00±0.00	3.36±0.12	-0.11±0.00	-18.58±10.39	0.46±0.02	0.45±0.01	1046.35±14.92 s
	WACH	0.99±0.01	0.57±0.04	0.00±0.00	2.67±0.15	-0.07±0.01	-10.72±3.68	**0.34±0.01**	**0.29±0.01**	1731.10±62.81 s
	ARTELT	-	-	-	-	-	-	-	-	- s
	PPCEF	**1.00±0.00**	**1.00±0.00**	**1.00±0.00**	1.03±0.02	0.04±0.00	**2.91±0.04**	0.65±0.01	0.47±0.01	19.55±0.30 s
Digits	CBCE	**1.00±0.00**	**1.00±0.00**	0.18±0.04	1.02±0.06	0.04±0.00	23.66±3.76	16.29±0.61	3.09±0.02	**0.54±0.16** s
	CEGP	0.95±0.01	0.46±0.03	0.02±0.01	1.24±0.00	-0.03±0.00	-138.62±14.93	6.39±0.16	**1.42±0.03**	2523.28±54.03 s
	CEM	**1.00±0.00**	0.42±0.01	0.01±0.01	1.44±0.01	-0.06±0.01	-481.57±68.49	**6.34±0.55**	1.76±0.09	1260.54±51.03 s
	WACH	**1.00±0.00**	0.72±0.03	0.00±0.00	1.50±0.03	-0.07±0.01	-516.44±72.51	11.04±0.36	2.13±0.07	3342.38±104.14 s
	ARTELT	-	-	-	-	-	-	-	-	- s
	PPCEF	**1.00±0.00**	**1.00±0.00**	**0.98±0.01**	1.13±0.01	0.03±0.01	**43.87±2.38**	8.78±0.29	**1.42±0.05**	25.09±0.40 s
Wine	CBCE	**1.00±0.00**	0.96±0.04	0.37±0.05	1.11±0.03	0.03±0.02	1.71±1.93	4.05±0.22	1.31±0.06	**0.01±0.00** s
	CEGP	0.98±0.02	0.42±0.05	0.06±0.08	1.12±0.03	0.04±0.01	-1.36±3.40	1.31±0.18	**0.48±0.07**	486.70±46.60 s
	CEM	**1.00±0.00**	0.44±0.06	0.00±0.00	1.40±0.05	-0.02±0.01	-17.90±4.72	**1.24±0.09**	0.67±0.06	117.67±3.86 s
	WACH	**1.00±0.00**	0.80±0.16	0.01±0.01	1.29±0.10	0.00±0.02	-10.69±7.38	2.07±0.23	0.80±0.06	224.63±6.95 s
	ARTELT	-	-	-	-	-	-	-	-	- s
	PPCEF	**1.00±0.00**	**0.97±0.03**	**0.99±0.01**	1.01±0.01	0.09±0.01	**9.79±0.59**	1.71±0.10	0.55±0.04	10.32±0.73 s

Table 7: Detailed Comparative Results of Probabilistically Plausible Counterfactual Explanation Methods for **Neural Oblivious Decision Ensembles** classifier. We offer a detailed comparison of our method with other established approaches using various datasets. This evaluation presents both the mean values and the standard deviations obtained through five-fold cross-validation.

Dataset	Method	Coverage ↑	Validity ↑	Prob. Plaus. ↑	LOF	IsoForest	Log Dens. ↑	L1 ↓	L2 ↓	Time ↓
Moons	CBCE	**1.00±0.00**	**1.00±0.00**	0.50±0.01	1.04±0.03	0.02±0.01	0.90±1.11	0.62±0.06	0.48±0.04	**0.41±0.09** s
	CEGP	-	-	-	-	-	-	-	-	- s
	CEM	**0.99±0.01**	**1.00±0.00**	0.07±0.02	2.00±0.09	-0.07±0.01	-4.36±2.79	0.61±0.03	0.57±0.02	1011.68±15.44 s
	WACH	**1.00±0.00**	**1.00±0.00**	0.02±0.02	1.29±0.04	0.01±0.00	-0.48±0.33	**0.33±0.01**	**0.26±0.01**	1736.54±19.44 s
	ARTELT	-	-	-	-	-	-	-	-	- s
	PPCEF	**1.00±0.00**	**1.00±0.00**	**0.99±0.00**	0.99±0.01	0.03±0.01	**1.60±0.03**	0.42±0.03	0.33±0.03	39.68±4.48 s
Law	CBCE	**1.00±0.00**	**1.00±0.00**	0.48±0.36	1.05±0.02	0.04±0.02	1.27±0.43	0.61±0.03	0.40±0.02	**1.08±0.04** s
	CEGP	-	-	-	-	-	-	-	-	- s
	CEM	0.97±0.01	**1.00±0.00**	0.29±0.01	1.27±0.02	-0.02±0.00	-0.41±0.14	**0.33±0.02**	0.31±0.01	2103.30±31.69 s
	WACH	**0.99±0.01**	**1.00±0.00**	0.45±0.06	1.08±0.01	0.03±0.00	1.16±0.06	0.41±0.01	0.29±0.00	4401.60±240.03 s
	ARTELT	-	-	-	-	-	-	-	-	- s
	PPCEF	**1.00±0.00**	**1.00±0.00**	**1.00±0.00**	1.03±0.00	0.07±0.00	**2.04±0.01**	0.38±0.01	**0.23±0.01**	61.67±2.88 s
Audit	CBCE	**1.00±0.00**	**1.00±0.00**	0.79±0.28	11.80±20.40	0.14±0.00	54.92±3.92	2.55±0.19	1.24±0.10	**0.38±0.08** s
	CEGP	-	-	-	-	-	-	-	-	- s
	CEM	0.52±0.03	**1.00±0.00**	0.00±0.01	$1.52{\cdot}10^{8}{\pm}1.43{\cdot}10^{8}$	0.11±0.00	12.63±14.72	**1.06±0.10**	**0.56±0.04**	906.63±24.07 s
	WACH	0.98±0.03	**1.00±0.00**	0.03±0.04	$6.90{\cdot}10^{7}{\pm}7.94{\cdot}10^{7}$	0.06±0.01	-33.50±50.95	1.59±0.14	0.97±0.04	2347.91±227.09 s
	ARTELT	-	-	-	-	-	-	-	-	- s
	PPCEF	**1.00±0.00**	**1.00±0.00**	**0.99±0.01**	$2.32{\cdot}10^{7}{\pm}2.99{\cdot}10^{7}$	0.09±0.01	**51.67±4.53**	2.06±0.12	0.80±0.09	80.34±21.47 s
Heloc	CBCE	**1.00±0.00**	**1.00±0.00**	0.55±0.03	1.09±0.08	0.08±0.03	28.88±4.06	2.85±0.40	0.82±0.11	**17.53±0.92** s
	CEGP	-	-	-	-	-	-	-	-	- s
	CEM	0.94±0.01	**1.00±0.00**	0.10±0.01	1.35±0.01	0.05±0.01	9.00±3.61	**0.47±0.08**	**0.29±0.02**	14772.66±226.75 s
	WACH	0.96±0.03	**1.00±0.00**	0.10±0.02	$2.12{\cdot}10^{8}{\pm}4.23{\cdot}10^{8}$	0.05±0.01	10.75±10.46	0.85±0.05	0.36±0.04	37254.33±3666.87 s
	ARTELT	-	-	-	-	-	-	-	-	- s
	PPCEF	**1.00±0.00**	0.94±0.01	**1.00±0.00**	1.08±0.00	0.09±0.00	**31.85±0.41**	1.02±0.06	0.28±0.02	126.05±33.10 s
Blobs	CBCE	**1.00±0.00**	**1.00±0.00**	0.27±0.15	1.02±0.03	0.03±0.02	-35.52±15.68	0.95±0.01	0.72±0.01	**0.77±0.08** s
	CEGP	-	-	-	-	-	-	-	-	- s
	CEM	0.90±0.04	**1.00±0.00**	0.00±0.00	3.10±0.09	-0.11±0.00	-24.98±10.42	0.58±0.02	0.52±0.02	1329.91±14.61 s
	WACH	0.94±0.02	**1.00±0.00**	0.00±0.00	2.65±0.11	-0.08±0.00	-11.76±3.58	**0.37±0.02**	**0.33±0.01**	2406.40±59.59 s
	ARTELT	-	-	-	-	-	-	-	-	- s
	PPCEF	**1.00±0.00**	**1.00±0.00**	**1.00±0.00**	1.02±0.02	0.04±0.00	**2.94±0.04**	0.65±0.01	0.47±0.01	47.69±5.21 s
Digits	CBCE	**1.00±0.00**	**1.00±0.00**	0.18±0.04	1.02±0.06	0.04±0.00	24.00±4.14	16.27±0.65	3.09±0.03	**3.12±0.24** s
	CEGP	-	-	-	-	-	-	-	-	- s
	CEM	**1.00±0.00**	**1.00±0.00**	0.03±0.01	1.32±0.01	-0.02±0.00	-39.45±10.78	4.07±0.14	1.44±0.03	5451.83±19.15 s
	WACH	**1.00±0.00**	**1.00±0.00**	0.16±0.03	1.12±0.01	0.02±0.01	7.02±5.00	**2.93±0.05**	**1.13±0.01**	15376.44±290.44 s
	ARTELT	-	-	-	-	-	-	-	-	- s
	PPCEF	**1.00±0.00**	**1.00±0.00**	**1.00±0.00**	1.15±0.01	0.02±0.01	**43.97±1.95**	7.76±0.20	1.36±0.03	69.45±3.94 s
Wine	CBCE	**1.00±0.00**	**1.00±0.00**	0.39±0.03	1.11±0.04	0.02±0.02	2.04±1.86	4.05±0.22	1.31±0.06	**0.10±0.07** s
	CEGP	-	-	-	-	-	-	-	-	- s
	CEM	**1.00±0.00**	**1.00±0.00**	0.00±0.00	1.34±0.04	0.00±0.01	-21.02±14.00	1.11±0.19	0.65±0.06	225.63±11.18 s
	WACH	**0.99±0.01**	**1.00±0.00**	0.08±0.06	1.12±0.04	0.04±0.01	-1.05±2.49	**1.10±0.04**	0.59±0.02	459.91±13.22 s
	ARTELT	-	-	-	-	-	-	-	-	- s
	PPCEF	**1.00±0.00**	**1.00±0.00**	**1.00±0.00**	1.01±0.01	0.09±0.00	**9.82±0.66**	1.69±0.09	**0.55±0.03**	19.39±1.06 s

Table 8: Detailed Ablation Study on Loss Function Selection. This table includes additional metrics: LOF and IsoForest

Dataset	Loss	Cov.↑	Val.↑	Prob. Plaus.↑	LOF	IsoForest	Log Dens.↑	L1↓	L2↓
Moons	Ours	**1.00±0.00**	**1.00±0.00**	**1.00±0.00**	1.01±0.02	0.04±0.01	1.69±0.07	**0.45±0.01**	**0.36±0.01**
	BCE	**1.00±0.00**	**1.00±0.00**	**0.99±0.00**	1.04±0.03	-0.01±0.01	**1.74±0.09**	0.89±0.03	0.69±0.02
Law	Ours	**1.00±0.00**	**1.00±0.00**	**1.00±0.00**	1.03±0.00	0.07±0.00	**2.05±0.02**	**0.37±0.01**	**0.23±0.01**
	BCE	**1.00±0.00**	**1.00±0.00**	**0.98±0.01**	0.99±0.00	0.01±0.01	1.67±0.01	0.97±0.02	0.60±0.01
Audit	Ours	**1.00±0.00**	**0.99±0.01**	**0.99±0.01**	$4.25{\cdot}10^7{\pm}9.32{\cdot}10^7$	0.08±0.01	51.64±4.53	**2.04±0.15**	**0.79±0.12**
	BCE	**1.00±0.00**	**0.99±0.01**	**0.98±0.01**	$3.59{\cdot}10^8{\pm}1.16{\cdot}10^8$	0.09±0.01	**52.54±4.54**	3.01±0.20	1.25±0.10
Heloc	Ours	**1.00±0.00**	**1.00±0.00**	**1.00±0.00**	$6.47{\cdot}10^7{\pm}2.16{\cdot}10^7$	0.07±0.00	**32.42±0.34**	**0.90±0.03**	**0.23±0.01**
	BCE	**1.00±0.00**	**1.00±0.00**	**0.99±0.00**	$1.67{\cdot}10^8{\pm}6.18{\cdot}10^7$	0.05±0.01	32.11±0.45	2.77±0.13	0.78±0.05
Blobs	Ours	**1.00±0.00**	**1.00±0.00**	**1.00±0.00**	1.01±0.01	0.04±0.01	**3.00±0.11**	**0.69±0.05**	**0.50±0.04**
	CE	**1.00±0.00**	**1.00±0.00**	0.93±0.01	1.05±0.02	0.03±0.01	2.85±0.04	0.82±0.02	0.60±0.01
Digits	Ours	**1.00±0.00**	**1.00±0.00**	**1.00±0.00**	1.12±0.01	0.03±0.01	**44.42±1.87**	**8.27±0.24**	**1.33±0.04**
	CE	**1.00±0.00**	**1.00±0.00**	**1.00±0.00**	1.12±0.01	0.02±0.01	44.18±2.09	12.67±0.15	2.13±0.04
Wine	Ours	**1.00±0.00**	**1.00±0.00**	**1.00±0.00**	1.01±0.01	0.09±0.01	**9.72±0.62**	**1.65±0.09**	**0.53±0.04**
	CE	**1.00±0.00**	**1.00±0.00**	**0.99±0.01**	1.06±0.03	0.02±0.01	9.29±0.71	3.87±0.18	1.29±0.06

Table 9: Dataset Characteristics and Model Performances. This table provides an overview of the datasets used in our experiments, including the number of samples (N), number of features (d), number of classes (C), accuracy of Logistic Regression (LR Acc.), Multi-Layer Perceptron (MLP Acc.), and the log density of the Masked Autoregressive Flow (MAF Log Dens.).

Dataset	N	d	C	LR Acc.	MLP Acc.	MAF Log Dens.
Moons	1,024	2	2	0.85	0.98	1.38
Law	2,220	3	2	0.75	0.75	1.23
Audit	610	23	2	0.95	0.98	48.15
Heloc	10,459	23	2	0.70	0.70	28.67
Wine	178	13	3	0.90	0.97	7.21
Blobs	1,500	2	3	1.00	1.00	2.58
Digits	5,620	64	10	0.94	0.95	35.80

B Datasets

In Tab. 9, we provide detailed descriptions of the datasets utilized in our study: Moons, Law[2], Audit[3], Heloc[4], Wine[5], Blobs and Digits[6]. The Moons dataset is an artificially generated set comprising two interleaving half-circles. It includes a standard deviation of Gaussian noise set at 0.01. The Law dataset originates from the Law School Admissions Council (LSAC) and is referred to in the literature as the Law School Admissions dataset [34]. For our analysis, we selected three features that exhibit the highest correlation with the target variable: entrance exam scores (LSAT), grade-point average (GPA), and first-year average grade (FYA). The Audit dataset, which encompasses comprehensive one-year non-confidential data of firms in the years 2015 to 2016, is collected from the Auditor Office of India to build a predictor for classifying suspicious firms. The Heloc dataset, initially utilized in the 'FICO xML Challenge', consists of Home Equity Line of Credit (HELOC) applications submitted by real homeowners. This dataset comprises various numeric features that encapsulate information from the applicant's credit report. The primary objective is to predict whether the applicant will repay their HELOC account within a two-year period. This prediction is instrumental in determining the applicant's qualification for a line of credit. The Wine dataset comprises chemical analysis results for wines originating from the same region in Italy, produced from three distinct cultivars. This analysis quantified 13 different constituents present in each of the three wine varieties. The Blobs dataset is an artificially generated isotropic Gaussian blobs, characterized by equal variance. The Digits dataset is utilized for the optical recognition of handwritten digits. It consists of 32x32 bitmap images that are segmented into non-overlapping 4x4 blocks. Within each block, the count of 'on' pixels is recorded, resulting in an 8x8 input matrix. Each element of this matrix is an integer between 0 and 16.

C Density Estimator - Additional Results

This experiment assesses the efficacy of Normalizing Flow models, particularly in high-dimensional datasets, against traditional Kernel Density Estimation (KDE). We compared KDE with three Normalizing Flow architectures: RealNVP, NICE, and Masked Autoregressive Flow (MAF). The mean log density results for the test datasets are detailed in Tab. 10. For lower-dimensional datasets like Moons, Law and Blobs, KDE and MAF show comparable performance, significantly outperforming RealNVP and NICE. However, in high-dimensional datasets such as Audit, Heloc and Digits, MAF demonstrates a substantial advantage over the other density estimators. These findings reinforce our proposition to employ Normalizing Flows, especially the MAF architecture, as effective density estimators in our method.

[2] https://www.kaggle.com/datasets/danofer/law-school-admissions-bar-passage
[3] https://archive.ics.uci.edu/dataset/475/audit+data
[4] https://community.fico.com/s/explainable-machine-learning-challenge
[5] https://archive.ics.uci.edu/dataset/109/wine
[6] https://archive.ics.uci.edu/dataset/80/optical+recognition+of+handwritten+digits

Table 10: Comparative Analysis of Density Estimators. We present the mean log density results for KDE, RealNVP, NICE, and MAF across various datasets. It highlights the superior performance of MAF in high-dimensional datasets (Audit and Heloc) and its comparable efficacy to KDE in lower-dimensional datasets (Moons and Law), underscoring the effectiveness of MAF as a density estimator in our method.

DATASET	KDE	REALNVP	NICE	MAF
MOONS	0.95±0.01	-1.85±0.00	-1.86±0.00	**1.38±0.07**
LAW	1.16±0.03	-2.79±0.00	-2.80±0.00	**1.23±0.05**
AUDIT	10.75±28.15	-21.25±0.13	-21.34±0.22	**48.15±8.41**
HELOC	22.44±0.32	-21.12±0.00	-21.19±0.00	**28.67±0.42**
WINE	5.95±0.57	-12.05±0.02	-12.08±0.01	**7.21±0.80**
BLOBS	2.10±0.02	-1.84±0.00	-1.84±0.00	**2.58±0.05**
DIGITS	22.66±1.14	-59.33±0.05	-59.79±0.09	**35.80±3.30**

D Implementation details

In the implementation of our experiments, we utilized Python [28] as the primary programming language. The core of our computational framework was PyTorch [20], a popular open-source machine learning library. A key feature of our implementation was the gradient optimization approach, designed to be executed in batches. This approach was particularly effective, allowing us to process entire test sets in a single batch. Our experiments were conducted on an M1 Apple Silicon CPU paired with 16GB of RAM. This hardware setup was more than sufficient for our experiment needs, providing enough capacity for the computational demands of our algorithm while ensuring fast processing speeds.