Politechnika Wrocławska

FIELD OF SCIENCE: Engineering and Technology

DISCIPLINE OF SCIENCE: Information and Communication Technology

DOCTORAL DISSERTATION

“Methods for multi-object representation learning in images and videos”

mgr inż. Piotr Zieliński

Supervisor:
dr hab. inż. Tomasz Kajdanowicz, prof. uczelni

Keywords: representation learning, generative modelling, computer vision

WROCŁAW 2025

Abstract

Visual representation learning facilitates the extraction of useful information from scenes presented in images and videos, which is essential for numerous computer vision applications. Global representation learning methods typically merge all scene elements into a single embedding, limiting the ability to distinguish individual objects clearly. Multi-object representation learning explicitly models scenes as collections of distinct entities, enabling structured, human-like understanding crucial for tasks such as visual reasoning, object tracking, and robotics.

Despite significant advancements, several challenges persist in this area. Early approaches relied on sequential inference, limiting scalability to visually complex scenes. Convolutional grid-based methods improved object discovery efficiency but still required sequential processing of object glimpses and were constrained by fixed spatial resolution, making it difficult to represent objects of varying sizes. Fully unsupervised models often fail to disentangle objects reliably in realistic settings, frequently encoding multiple objects as a single entity. Temporal extensions for videos further amplified these challenges, additionally exhibiting computational complexity, which restricts their practical applicability.

This thesis addresses these gaps by integrating advances from modern one-stage object detection architectures into multi-object representation learning frameworks, which was motivated by an investigation into using visual features from an object detection network to guide deep reinforcement learning in robotic navigation. It proposes SSDIR, a novel method leveraging multi-scale feature maps within a parallel spatial grid-based encoding strategy, using pre-trained object detectors as a robust foundation for unsupervised representation learning and precise object localisation in complex, real-world settings. Additionally, RDIR introduces an implicit temporal extension through a recurrent architecture, ensuring consistent object representations across video frames. Furthermore, a diffusion-based model, DetDiff, conditioned on detection-guided representations enhances generative quality, enabling controllable image synthesis. Extensive experiments confirm improvements in representation quality and their performance in downstream tasks, demonstrating the efficacy and versatility of the proposed approaches across diverse visual domains.

Streszczenie

Uczenie reprezentacji wizualnej umożliwia wydobycie istotnych informacji ze scen zawartych na obrazach i materiałach wideo, co stanowi fundament wielu zastosowań w dziedzinie wizji komputerowej. Globalne metody uczenia reprezentacji zazwyczaj łączą wszystkie elementy sceny w jedno osadzenie, co utrudnia precyzyjne rozróżnianie poszczególnych obiektów. Uczenie reprezentacji wielu obiektów modeluje sceny jako zbiory odrębnych encji, co pozwala na ustrukturyzowaną analizę scen, podobną do ludzkiego sposobu ich postrzegania. Jest to kluczowe dla zadań takich jak rozumowanie wizualne, śledzenie obiektów czy robotyka.

Pomimo znaczących postępów, w tym obszarze nadal istnieje kilka wyzwań. Wczesne rozwiązania polegały na wnioskowaniu sekwencyjnym, co ograniczało skalowalność do złożonych wizualnie scen. Metody oparte na konwolucyjnych siatkach cech poprawiły efektywność wykrywania obiektów, jednak wciąż wymagały sekwencyjnego przetwarzania wycinków obiektów oraz nie potrafiły tworzyć reprezentacji przy zmiennych rozmiarach obiektów z powodu sztywnej rozdzielczości map cech. W pełni nienadzorowane podejścia często nie mogły niezawodnie rozróżniać poszczególnych obiektów w realistycznych scenach, łącząc wiele z nich jako jedno osadzenie. Rozszerzenia temporalne dla filmów wideo pogłębiły te wyzwania, dodatkowo zwiększając złożoność obliczeniową, co ogranicza ich praktyczne zastosowanie.

Niniejsza rozprawa podejmuje te wyzwania, integrując postępy z jednoetapowych architektur wykrywania obiektów z metodami uczenia reprezentacji wielu obiektów, co było zainspirowane badaniami nad zastosowaniem cech z modeli detekcji w nawigacji robotycznej z użyciem głębokich modeli uczenia ze wzmocnieniem. Zaproponowano metodę SSDIR, wykorzystującą wieloskalowe mapy cech w metodzie kodowania opartej na przestrzennej siatce cech, wykorzystując wstępnie wytrenowaną sieć do detekcji obiektów jako fundament do nienadzorowanego uczenia reprezentacji i precyzyjnego ustalania położenia obiektów w złożonych, rzeczywistych warunkach wizualnych. Metoda RDIR rozszerza model na wideo, wprowadzając architekturę rekurencyjną dla spójnych reprezentacji obiektów w kolejnych klatkach. Model dyfuzyjny DetDiff, warunkowany reprezentacjami wzbogaconymi przez detekcje, poprawia zdolności generatywne, umożliwiając kontrolowane tworzenie obrazów. Szerokie eksperymenty potwierdzają polepszenie jakość reprezentacji i ich efektywne zastosowanie w zdaniach docelowych, wykazując skuteczność i wszechstronność proponowanych podejść w różnych dziedzinach wizualnych.

Acknowledgments

I would like to thank my supervisor, Prof. Tomasz Kajdanowicz, for his extensive knowledge and steady support throughout my PhD studies. His advice and direction were crucial in shaping the course of my research. I am also grateful to Prof. Urszula Markowska-Kaczmar, for her encouragement at the very beginning, which gave me the confidence to pursue doctoral studies. Thanks as well to my colleagues at the Department of Artificial Intelligence at Wroclaw University of Science and Technology for the collaborative atmosphere and many fruitful conversations, making this journey productive, but also enjoyable.

To my wife, Katarzyna, whose unconditional support, understanding, and love have been essential, especially during challenging moments – I simply couldn't have done it without her, special thanks! Finally, I want to thank my parents for always believing in me and supporting me in every step of my academic path.

Symbols

AP	Average Precision (area under the precision-recall curve for object detection)
α_t	Complement of variance schedule, defined as $1 - \beta_t$
$\bar{\alpha}_t$	Cumulative product of α_t
$\mathbf{B}$	Set of predicted bounding boxes, $\mathbf{B} = \{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n\}$
$\mathbf{b}_i = [x_i, y_i, w_i, h_i]$	Bounding box for the i-th object, $\mathbf{b}_i \in \mathbb{R}^4$ (centre coordinates (x_i, y_i), width w_i, and height h_i)
β_t	Variance schedule parameter at diffusion step t
C	Number of image channels (e.g., 3 for RGB)
$\mathbf{c}_i$	Vector of predicted class confidence scores for the i-th object
c_i	Class label assigned to the i-th object
D	Dimensionality of object-level latent vector $\mathbf{z}_i$
$D_{\mathrm{KL}}(p \parallel q)$	Kullback-Leibler divergence measuring difference between two distributions p and q
$\boldsymbol{\epsilon}^{(t)}$	True noise sampled from standard normal distribution at timestep t
$\hat{\boldsymbol{\epsilon}}^{(t)}$	Predicted noise at timestep t
ϵ	Sample from standard normal distribution
f	Downstream predictor operating on learned representations, $f : \mathcal{Z} \to \mathcal{Y}$ or $f : \mathcal{Z}^K \to \mathcal{Y}$
FN	False Negatives — incorrectly predicted negative instances
FP	False Positives — incorrectly predicted positive instances
g	Representation learning function mapping inputs to latent features, $g : \mathcal{X} \to \mathcal{Z}$ or $\mathcal{Z}^K$
H	Image height (number of pixels vertically)
h^*	Unknown ground-truth function mapping inputs to targets, $h^* : \mathcal{X} \to \mathcal{Y}$
HOTA_α	Higher Order Tracking Accuracy at localisation threshold α
$\mathrm{IoU}(\cdot, \cdot)$	Intersection over Union between two sets
K	Number of object instances in a scene
$\mathcal{L}$	Loss function comparing predicted and true outputs
λ	Weighting factor loss function components
$\mathbf{m}_i$	Binary segmentation mask for the i-th object, $\mathbf{m}_i \in \{0, 1\}^{H \times W}$

$\mathbf{o}^i_{\text{dec}}$	Decoded i-th object patch before spatial transformation
$\mathbf{o}^i_{\text{transformed}}$	Object patch after applying spatial transformation according to $\mathbf{z}^i_{\text{where}}$
$\mathbf{p}$	Object presence probabilities predicted by YOLOv4
p_{ij}	Object presence confidence at cell (i, j)
$p(\mathbf{x})$	Marginal likelihood of the input image $\mathbf{x}$ under the generative model
$p(\mathbf{x} \mid \mathbf{z})$	Likelihood of input given latent variables
$p_\theta(\mathbf{x}^{(t-1)} \mid \mathbf{x}^{(t)})$	Reverse diffusion distribution predicting denoised sample
$p(\mathbf{z})$	Prior distribution over latent variables in generative models
$\text{PE}(x_{ij}, y_{ij})$	2D sinusoidal positional encoding based on coordinates
$q(\mathbf{x}^{(t)} \mid \mathbf{x}^{(t-1)})$	Forward diffusion distribution at step t
$q(\mathbf{z} \mid \mathbf{x})$	Approximate posterior distribution in variational inference
$\mathbf{r}$	Latent representation grid produced by the encoder
$\mathbf{r}_{ij}$	Latent representation at cell (i, j) in the feature grid
$\mathbf{r}^{\text{PE}}_{ij}$	Positionally encoded latent vector at cell (i, j)
S	Set of confidence scores associated with predicted bounding boxes, $S = \{s_1, s_2, \ldots, s_n\}$
T	Number of frames in a video
τ	Threshold value
TN	True Negatives — correctly predicted negative instances
TP	True Positives — correctly predicted positive instances
W	Image width (number of pixels horizontally)
$\mathcal{X}$	Input space (e.g., space of images or videos)
$\mathbf{x}$	Input data sample (e.g., image or video tensor)
$\hat{\mathbf{x}}$	Reconstructed image or video produced from latent slots
$\mathbf{x}_i$	Frame i of a video sequence
$\mathbf{x}^{(0)}$	Original data sample before the diffusion process
$\mathbf{x}^{(t)}$	Noised version of $\mathbf{x}^{(0)}$ at diffusion step t
$x_{h,w,c}$	Pixel value at location (h, w) in channel c
$\mathcal{Y}$	Target/output space (e.g., class labels, regression targets)
$\mathbf{y}$	Target output associated with input $\mathbf{x}$
$\mathcal{Z}$	Latent feature space capturing semantic or object-level information
$\mathbf{z}$	Learned latent representation of input $\mathbf{x}$
$\mathbf{z}_i$	Latent representation of the i-th discovered object
$\mathbf{z}^{(0)}$	Latent encoding of the input image, used in LDMs
$\mathbf{z}^{(t)}$	Noisy latent representation at diffusion step t
$\mathbf{z}^i_{\text{depth}}$	Scalar latent variable representing the relative depth of the i-th object
$\mathbf{z}^i_{\text{present}}$	Binary latent variable indicating whether the i-th object is present in the corresponding grid cell
$\mathbf{z}^i_{\text{what}}$	Latent vector capturing the i-th object's visual appearance features
$\mathbf{z}^i_{\text{where}}$	Latent vector describing the i-th object's position and size

Abbreviations

2D	Two-dimensional
3D	Three-dimensional
A2C	Advantage Actor-Critic
A3C	Asynchronous Advantage Actor-Critic
AE	Autoencoder
AIR	Attend, Infer, Repeat
AP	Average Precision
ARI	Adjusted Rand Index
AUV	Autonomous Underwater Vehicle
BB	Bounding box
BiGRU	Bidirectional Gated Recurrent Unit
BN	Batch Normalisation
CLEVR	Compositional Language and Elementary Visual Reasoning Dataset
CLIP	Contrastive Language-Image Pre-training
CNN	Convolutional Neural Network
Conv	Convolutional
DDPG	Deep Deterministic Policy Gradient
DDPM	Denoising Diffusion Probabilistic Model
DetDiff	Detection-Guided Latent Diffusion
DETR	Detection Transformer
DM	Diffusion Model
DQN	Deep Q-Network
DRL	Deep Reinforcement Learning
dVAE	Discrete Variational Autoencoder
E2E	End-to-end
ELBO	Evidence Lower Bound
FC	Fully-connected
FFHQ	Flickr-Faces-HQ Dataset
FG-ARI	Foreground Adjusted Rand Index
FID	Fréchet Inception Distance
FN	False Negative
FNA	False Negative Association

FP	False Positive
FPA	False Positive Association
FVD	Fréchet Video Distance
GAN	Generative Adversarial Network
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
HOG	Histogram of Oriented Gradients
HOTA	Higher Order Tracking Accuracy
ICA	Independent Component Analysis
IDF1	Identification F1-score
IoU	Intersection over Union
KL	Kullback-Leibler
LDM	Latent Diffusion Model
LPIPS	Learned Perceptual Image Patch Similarity
LSD	Latent Slot Diffusion
LSTM	Long Short-Term Memory
MAE	Masked Autoencoder
mAP	Mean Average Precision
mBO	Mean Best Overlap
mIoU	Mean Intersection over Union
MLP	Multi-Layer Perceptron
MOT15	Multiple Object Tracking 2015 Dataset
MOTA	Multiple Object Tracking Accuracy
MOVi	Multi-Object Video Dataset
MS COCO	Microsoft Common Objects in Context Dataset
MSE	Mean Squared Error
NLP	Natural Language Processing
NMS	Non-Maximum Suppression
$\text{OBJ}_{\sim}$	DetDiff variant without object-aware loss
PCA	Principal Component Analysis
PE	Positional Encoding
$\text{PE}_{\sim}$	DetDiff variant without positional encoding
PPO	Proximal Policy Optimisation
R-CNN	Region-based Convolutional Neural Networks
RDIR	Recurrent Detect, Infer, Repeat
ReLU	Rectified Linear Unit
RGB	Red, Green, Blue
RGB-D	Red, Green, Blue and Depth
RI	Rand Index
RL	Reinforcement Learning
RNN	Recurrent Neural Network

ROV	Remotely Operated Vehicle
RPN	Region Proposal Network
SA	Slot Attention
SAVi	Slot Attention for Videos
SIFT	Scale-Invariant Feature Transform
SPAIR	Spatially Invariant Attend, Infer, Repeat
SQAIR	Sequential Attend, Infer, Repeat
SSD	Single Shot Multi-Box Detector
SSDIR	Single-Shot Detect, Infer, Repeat
SSDIR-YOLO	SSDIR model with a YOLOv4 backbone and detection head
STN	Spatial Transformer Network
SURF	Speeded-Up Robust Features
SVM	Support Vector Machine
TN	True Negative
TP	True Positive
TPA	True Positive Association
t-SNE	t-Distributed Stochastic Neighbour Embedding
VAE	Variational Autoencoder
ViT	Vision Transformer
VQA	Visual Question Answering
VQ-VAE	Vector Quantised Variational Autoencoder
YOLO	You Only Look Once

Contents

Chapter 1

Introduction

Understanding complex visual scenes composed of multiple entities is a long-standing challenge in computer vision and machine learning. Unlike pixel-level processing, human perception focuses on semantically meaningful objects naturally, inferring their attributes, tracking their dynamics and reasoning about their interactions over time [52]. Replicating this ability in computer systems requires learning structured, object-centric representations that reflect these characteristics of real-world scenes. This thesis investigates the problem of multi-object representation learning, which focuses on automatic discovery and encoding of objects from raw visual data, both in static images and video sequences. This research area extends beyond classical vision tasks such as object detection or instance segmentation, attempting not only to locate and classify objects, but also to provide a deep understanding of the scene by generating disentangled and reusable representations that can be applied in complex downstream reasoning.

This chapter presents an introduction to the research problem addressed in the thesis. Section 1.1 outlines the characteristics of the research domain and formally defines the problem. Section 1.2 presents a review of related work, covering object detection, global image and video representation learning, and prior approaches to multi-object modelling, together with the datasets and main evaluation metrics. Section 1.3 provides the motivation for this research, while Section 1.4 details the research goals, including the plan of investigation, hypotheses and research questions. The chapter concludes with an overview of the main contributions (Section 1.5) and a description of the structure of the thesis (Section 1.6).

1.1 Characteristics of the Research Domain

Extracting useful features from complex, high-dimensional data such as images, videos, graphs, or natural language has been a challenge for computer applications. Unlike low-dimensional numerical inputs, these data types are rich in structure, redundancy and ambiguity, making them difficult to process directly. Historically, computer systems have relied heavily on task-specific feature engineering, where domain experts design transformations to extract relevant information from raw data for a particular task. In more recent machine learning approaches, especially deep neural networks, feature extraction was automated, learned jointly with the model by optimising performance on the end task. However, these task-specific models tend to extract features that are coupled tightly to the original objective they were trained for, making it difficult to transfer them to a more general setup [8]. For example, a deep neural network trained for image classification will encode high-level object categories, but might ignore fine-grained spatial details required for accurate object localisation.

Representation Learning. Motivated by these limitations, representation learning is a machine learning paradigm which aims at learning general-purpose, abstract representations that capture underlying explanatory factors of the data. These representations are intended to be meaningful and reusable, enabling better generalisation, interpretability and sample efficiency across a wide range of downstream applications. Representation learning stems from the hypothesis that most real-world data is generated by a relatively small number of latent factors, which interact in a complex way to produce high-dimensional data. By trying to reverse this generative process, representation learning models uncover the underlying structure of the data, providing verbose representations that support generalisation across tasks and domains, reducing the need for large labelled datasets.

Definition 1. *Let $\mathcal{X}$ be the input space and $\mathcal{Y}$ the target space, which are related by an unknown complex function $h^* : \mathcal{X} \to \mathcal{Y}$. Representation learning is the problem of learning a mapping $g : \mathcal{X} \to \mathcal{Z}$, where $\mathcal{Z}$ is a meaningful latent feature space, such that the learned representations $\mathbf{z} = g(\mathbf{x})$ preserve the high-level factors of variation in the data and admit the existence of a function $f : \mathcal{Z} \to \mathcal{Y}$ for which $f \circ g \approx h^*$, typically in the sense of minimising a loss function $\mathcal{L}(f(g(\mathbf{x})), \mathbf{y})$ over a data distribution $(\mathbf{x}, \mathbf{y}) \sim \mathcal{D}$.*

Learned representations should satisfy several properties [8]:

- **disentanglement**: different dimensions in $\mathcal{Z}$ correspond to distinct characteristics of the data,
- **invariance**: representations are robust to transformations that do not change semantic context, but sensitive to meaningful variations,
- **compactness**: representations are low-dimensional and sparse, which promotes generalisation,
- **completeness**: learned representations retain sufficient information for applying in downstream tasks.

Among the challenges of representation learning, one fundamental issue results from the ambiguity in defining a clear learning objective. Determining the quality of representations is non-trivial, as it often depends on performance in downstream tasks, which cannot be used as the learning objective, making model training and evaluation complicated. Additionally, effectively disentangling underlying factors of variation within the data can become particularly difficult when objects' features are complex and interdependent. Another significant challenge involves balancing invariance and sensitivity, as well as managing scalability and computational complexity. Properly matching model complexity to the available data is also critical to prevent overfitting to the training data and ensure generalisation capability. Finally, the choice of training data is essential for generative models, as unfair representations of examples will lead to biases and a lack of fairness.

Visual Representation Learning. In the domain of visual data, representation learning tries to transform raw pixel inputs into latent encodings that capture the semantic content of the scene from images or videos. The latent space produced by these models may focus on various level of abstraction, from the global topic of the image or video (e.g. "a crowded street"), to the most salient object within the scene (e.g. "a person riding a bike"), or the entire structure of the scene (e.g. "a group of people walking on a crosswalk"). In this context, representation learning aims to encode these high-dimensional inputs into a compact latent space $\mathcal{Z}$, which retains essential visual information about the scene while discarding intentionally irrelevant variation such as lighting or background textures. When applied to videos, the temporal dimension introduces both a source of contextual information and an added layer of complexity, as representations must also account for scene dynamics over time. Learning such representations simplifies reasoning about complex scenes, providing a deep understanding of their contents.

Definition 2. *An image is a two-dimensional grid of pixel values represented as a tensor* $\mathbf{x} \in \mathbb{R}^{H \times W \times C}$*, where:*

$$\begin{aligned} &H \in \mathbb{N} \ \textit{is the height (number of rows)},\\ &W \in \mathbb{N} \ \textit{is the width (number of columns)},\\ &C \in \mathbb{N} \ \textit{is the number of channels (e.g. } C = 3 \textit{ for RGB images).} \end{aligned}$$

Each element $x_{h,w,c} \in \mathbb{R}$ *represents the intensity of the pixel at position* (h, w) *in channel* c*. An image captures the spatial structure of a visual scene and all objects within it at a fixed point in time.*

Definition 3. *A video is a temporal sequence of image frames, represented as a tensor* $\mathbf{x} \in \mathbb{R}^{T \times H \times W \times C}$*, where:*

$$\begin{aligned} &T \in \mathbb{N} \ \textit{is the number of frames},\\ &H, W, C \ \textit{are as in Definition 2}. \end{aligned}$$

Each frame $\mathbf{x}_i \in \mathbb{R}^{H \times W \times C}$*, for* $i \in \{1, ..., T\}$ *captures the visual content of the scene at time step* i*. Hence, video encodes both spatial and temporal information about the scene.*

Multi-Object Representation Learning. Global representations of visual inputs tend to interpret the scene as a whole; however, they lack the internal structure required to reason about individual entities effectively. Real-world scenes typically consist of multiple distinct objects of different characteristics and behaviours; to reflect this compositional nature of visual scenes, multi-object representation learning aims to encode them into object-centric components. Instead of mapping an image or video to a single embedding, these models represent scenes as structured latent vectors $\mathbf{z} = (\mathbf{z}_1, ..., \mathbf{z}_K) \in \mathcal{Z}^K$, where each vector $\mathbf{z}_i \in \mathbb{R}^D$ (with D being the size of the representation) focuses on a single discovered entity in the scene. This approach enables modular understanding of objects in the scene and their relations, supporting generalisation to complex reasoning tasks.

Definition 4. *Given the setup of Definition 1, multi-object representation learning is the problem of learning a mapping* $g : \mathcal{X} \to \mathcal{Z}^K$ *from an input* $\mathbf{x} \in \mathcal{X}$ *(an image or video) to a structured latent representation* $\mathbf{z} = (\mathbf{z}_1, ..., \mathbf{z}_K) \in \mathcal{Z}^K$*, where each* $\mathbf{z}_i \in \mathbb{R}^D$ *captures features corresponding to a distinct object or entity in the scene. The representation aims to capture object-specific high-level factors of variation, ensuring the existence of a downstream function* $f : \mathcal{Z}^K \to \mathcal{Y}$ *such that* $f \circ g \approx h^*$*, typically by minimising a loss function* $\mathcal{L}(f(g(\mathbf{x})), \mathbf{y})$ *over a data distribution* $(\mathbf{x}, \mathbf{y}) \sim \mathcal{D}$*.*

Ideally, object-centric representations should encode each object in the scene into distinct, decomposed latent variables that independently and compositionally capture the structured nature of visual scenes. These representations should be robust enough to handle varying object counts, sizes, occlusions, background clutter, and interactions, facilitating scalability to complex, real-world environments.

Object Discovery and Object-Centric Learning. Typically, multi-object representation learning models decompose the scene in a way that object-level entity representations capture the appearance, location, and potentially dynamics of each distinct object. Therefore, multi-object representation learning can be viewed as a generalisation of the object discovery problem, which focuses on identifying and localising object instances in raw, unannotated visual input. It goes one step further, learning feature-rich embeddings for each of the discovered entities, which are not limited to one particular computer vision task, but can be applied in a broad variety of applications.

The foundation for this research direction was laid by traditional computer vision tasks like object detection and instance segmentation. Designed to localise and label distinct objects in the scene, methods for solving these problems are typically trained under supervision on labelled datasets, providing each object's location as bounding boxes or pixel-level masks, respectively. Many of the multi-object representation learning models adopt similar inductive biases for object localisation (rectangular attention boxes or region-based masking), which provides explicit disentanglement of objects. However, instead of classifying each object, multi-object representation learning models encode their characteristics into a reusable embedding, operating with weaker supervision, or even without it.

Definition 5. *Object detection is the task of predicting a set of bounding boxes* $\{\mathbf{b}_i\}_{i=1}^K$ *and associated class labels* $\{c_i\}_{i=1}^K$ *from an input image* $\mathbf{x} \in \mathbb{R}^{H \times W \times C}$*, where each bounding box* $\mathbf{b}_i \in \mathbb{R}^4$ *localises an object of class* c_i*.*

Definition 6. *Instance segmentation is the task of predicting a set of binary masks* $\{\mathbf{m}_i\}_{i=1}^K$*, each of shape* $H \times W$*, and corresponding class labels* $\{c_i\}_{i=1}^K$ *from an input image* $\mathbf{x} \in \mathbb{R}^{H \times W \times C}$*, such that each mask* $\mathbf{m}_i$ *indicates pixel-wise support of a distinct object instance of class* c_i*.*

1.2 Related Work

This section provides an overview of literature relevant to the research directions explored in this thesis, specifically focusing on object detection (Subsection 1.2.1), image and video representation learning (Subsection 1.2.2), and multi-object representation learning (Subsection 1.2.3). Given the scope and objectives of this work, the survey emphasises deep learning-based approaches and reflects state-of-the-art methodologies published up to 2025. Consequently, this literature review extends beyond the knowledge available at the time of conducting work included in Chapters 2-5.

1.2.1 Object Detection

Early approaches to computer vision were based on hand-crafted features and statistical models, most notably the use of sliding window classifiers combined with descriptors such as Haar-like features [178], Histogram of Oriented Gradients (HOG) [34], or Deformable Part-based Model (DPM) [55], to mention a few. With the advent of deep learning, classical approaches were superseded by methods capable of learning hierarchical feature representations directly from raw data, leading to substantial improvements in accuracy and generalisation. Given convolutional neural networks' superior performance and scalability, this thesis focuses exclusively on deep learning-based object detection methods.

Two-Stage Detectors. The development of deep learning-based object detection methods is grounded in the success of convolutional neural networks (CNNs) in large-scale image classification tasks, initiated by the breakthrough performance of AlexNet [105] on the ImageNet dataset. The capability of learning hierarchical visual features outperforming traditional hand-crafted representations led towards further improvements introduced by deeper and more regularised models such as VGGNet [156], Inception [167] or ResNet [76].

These architectures formed the backbone of early object detection frameworks. The first influential deep learning-based object detector, Regions with CNN features (R-CNN) [61], utilised pre-trained classification networks to extract feature representations from proposed object regions. This method processes images in two stages, first generating object proposals via an external proposal algorithm (selective search [173]), and then applying a CNN to each region, followed by class-specific support vector machines (SVMs). While offering better performance than classical detectors, R-CNN was computationally inefficient due to the need to apply the CNN independently to thousands of overlapping regions per image.

This limitation was addressed by Fast R-CNN [60], where a single convolutional feature map computed over the entire image was used with a Region of Interest pooling operation to extract features for each proposed region. Fast R-CNN improved over R-CNN in terms of training and inference speed, but still relied on a separate, non-learnable region proposal step.

The introduction of Faster R-CNN [149] enabled a fully end-to-end trainable architecture by incorporating a learnable Region Proposal Network (RPN) into the detection pipeline. The RPN shared convolutional features with the classification CNN, generating object proposals directly from the feature map.

Two-stage detectors offer high accuracy and robustness, especially in scenarios with multiple overlapping objects and high variability in object scale and aspect ratio. Their modular architecture allowed for further enhancements, including the addition of segmentation branches (e.g. Mask R-CNN [75]). However, the computational complexity and inference latency of this class of methods have also driven the field toward designs improving detection speed, with the aim of reaching real-time object detection.

One-Stage Detectors. Despite the strong performance of two-stage detectors, their multi-step architecture introduces latency, which limits their applicability for time-sensitive tasks such as autonomous driving, robotics, or real-time video analysis. One of the most successful approaches explored to improve detection speed was unifying the detection pipeline into a single forward pass of the image through the network. These models, named one-stage detectors, directly predict object classes and bounding boxes over a dense sampling of locations in the image, typically using the spatial grid-based attention (Figure 1.1), eliminating the need for an explicit region proposal stage and significantly improving inference speed. In this approach, feature maps (tensors of shape $H^{(l)} \times W^{(l)} \times C^{(l)}$, where $H^{(l)}$, $W^{(l)}$ are the height and width of the l-th feature map, and $C^{(l)}$ denotes the number of channels), extracted using a convolutional backbone, are treated as a grid of cells, each containing a $C^{(l)}$-sized feature vector corresponding with a region in the input image. This method provides a spatially structured intermediate features for localised inference.

One of the earliest and most influential one-stage models is You Only Look Once (YOLO) [146]. This model reframed object detection as a single regression problem over a fixed grid of image regions; it predicts bounding box coordinates and class probabilities

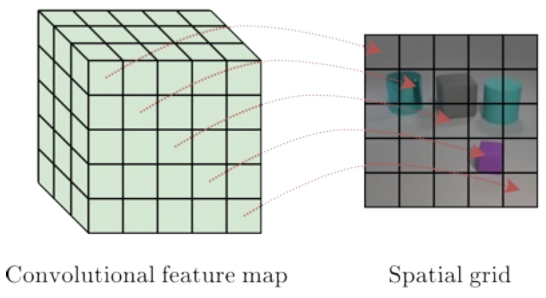


Figure 1.1: Spatial grid-based attention focusing on corresponding regions in the image

in a single forward pass through the network by applying prediction heads on each cell in the grid, assigning these predictions to a corresponding region in the original image. While the original design achieved real-time performance, the fixed, coarse spatial discretisation with a single grid made it struggle to detect small or overlapping objects.

To address this issue, Single Shot Multi-Box Detector (SSD) [122] extends the original YOLO design with two major improvements. It utilises anchor boxes, which simplify the task of bounding box regression: instead of predicting the absolute coordinates of bounding boxes, it uses a set of pre-defined boxes as reference points for the regression. Most notably, SSD leverages multiple feature maps as spatial grids, providing multi-scale intermediate features for predicting bounding box locations, aiming to overcome the issue of detecting objects of highly varying sizes (Figure 1.2). Subsequent iterations of YOLO [12, 147, 148, 180, 181] also introduced additional advancements, including anchor boxes, multi-scale feature maps, the use of batch normalisation, improved loss function, stronger convolutional backbones, etc. While they perform better than the original design, addressing its key limitations, only the most recent methods match the accuracy of state-of-the-art two-stage models. What is more, multi-scale feature maps tend to generate multiple predictions for the same object due to the overlap between the grids.

To address the problem of multiple overlapping bounding box predictions for the same object, object-detection models widely adopt Non-Maximum Suppression (NMS) as a post-processing technique, aiming at retaining only the most relevant bounding box for each detected object, therefore improving model precision and reducing redundancy. NMS is a greedy algorithm, which ensures that among a group of overlapping boxes (based on Intersection over Union threshold), only the one with the highest confidence is preserved (Algorithm 1). NMS was already used in early computer vision methods, filtering overlapping candidate detections produced by the sliding window or region proposals approaches. It is especially relevant in one-stage detectors, pruning redundant detections during inference and providing an accurate set of object predictions.

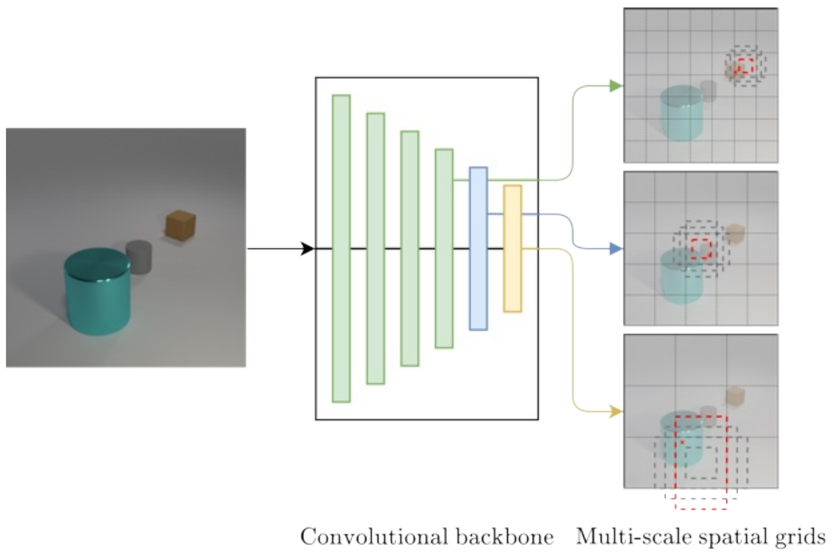


Figure 1.2: Multi-scale spatial grids extraction with 4 anchor boxes

In another attempt to address the imbalance between foreground and background region proposals generated by the grid-based approach, RetinaNet [118] introduced the focal loss, which weights well-classified examples down to focus learning on hard, misclassified instances. This model retains high inference speed, improving the accuracy to match the two-stage models. However, the addition of focal loss increases the complexity of hyperparameter tuning.

Transformer-Based Object Detection. Building on the success of natural language processing (NLP) transformers, the Vision Transformer (ViT) [43] demonstrated that pure self-attention can replace convolutions for image recognition. The capability of modelling long-range dependencies and the global context of transformers facilitates scaling beyond restricted receptive fields in CNN-based models.

Detection Transformer (DETR) model [16] was the first end-to-end object detector based on transformer architecture. It reframes object detection as a direct set prediction problem; in contrast to previous approaches, this eliminates the need for incorporating region proposals, anchor boxes and other heuristics. The model utilises a CNN-based backbone to extract image features, followed by a transformer encoder-decoder that predicts a fixed number of object queries. Each query outputs a class label and bounding box, and a bipartite matching loss (Hungarian loss) is used to align predictions with ground truth.

This design provides high interpretability of the process and removes the need for detection post-processing while achieving competitive accuracy. Subsequent models such as Deformable DETR [207], Co-DETR [211] and DINO [201] introduce various

Algorithm 1: Non-Maximum Suppression (NMS)

```
Input  : Set of predicted bounding boxes $\mathbf{B} = \{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n\}$ with associated
         confidence scores $\mathbf{S} = \{s_1, s_2, \ldots, s_n\}$, IoU threshold $\tau$
Output : Filtered set of bounding boxes $\mathbf{B}_F$
1  $\mathbf{B}_F \leftarrow \emptyset$
2  Sort $\mathbf{B}$ in descending order according to scores $S$
3  while $\mathbf{B} \neq \emptyset$ do
4      $\mathbf{b}_{max} \leftarrow$ first box in $\mathbf{B}$
5      $\mathbf{B}_F \leftarrow \mathbf{B}_F \cup \{\mathbf{b}_{max}\}$
6      $\mathbf{B} \leftarrow \mathbf{B} \setminus \{\mathbf{b}_{max}\}$
7      foreach $\mathbf{b} \in \mathbf{B}$ do
8          if $IoU(\mathbf{b}, \mathbf{b}_{max}) > \tau$ then
9              $\mathbf{B} \leftarrow \mathbf{B} \setminus \{\mathbf{b}\}$
10         end
11     end
12 end
13 return $B_F$
```

enhancements: multi-scale deformable attention, optimised encoder and decoder architectures and improved query initialisation, reducing the training time and reaching higher localisation accuracy, which makes transformer-based detectors the state-of-the-art in object detection. Despite the advantages over two- and one-stage detectors (modular architecture without engineered components and heuristics, improved accuracy), transformer-based detectors require larger model sizes, bigger datasets, and carefully tuned training protocols, and do not scale well during inference, which limits their usability in practical applications, especially for real-time object detection.

1.2.2 Image and Video Representation Learning

Extracting informative and compact feature representations from visual data is a prerequisite for modern computer vision tasks, including classification, object detection, instance segmentation and many more. Early solutions relied on hand-crafted feature descriptors that encode local structure in images; examples include Scale-Invariant Feature Transform (SIFT) [125], Speeded-Up Robust Features (SURF) [7], or Histogram of Oriented Gradients (HOG) [34] to mention a few. On the other hand, early data-driven feature extraction methods were grounded in statistical and signal processing techniques such as Principal Component Analysis (PCA), Independent Component Analysis (ICA) and sparse coding, which extracted low-dimensional representations of the input data. However, both groups of these methods lacked the capacity to adapt to complex, high-level visual semantics, their performance degrading with the increasing complexity of images, motivating the shift towards learned image representations.

Pre-trained CNNs (e.g. [76, 105, 156, 167], and others), particularly those trained on large-scale datasets, became widely adopted as generic feature extractors. The scalability and domain generalisation of these models were limited by the simple classification task; therefore, they were usually fine-tuned to downstream applications, following the transfer learning paradigm. This limitation, combined with a lack of key representation properties such as smooth latent space, equivariance to transformations and disentanglement of underlying factors of variation, inspired research towards explicit representation learning, where the goal is to learn structured, reusable and semantically meaningful feature spaces directly from data.

Self-Supervised Representation Learning. The challenges of collecting and annotating large datasets, as well as the limited scalability of domain-specific representations, motivated the research of methods which do not require human-provided labels. In image representation learning, one of the most prominent directions is applying self-supervised learning, based on designing pretext tasks, which are auxiliary objectives providing supervision from the data itself. Examples of pretext tasks include operating on image patches (predicting the relative position of two patches, solving a jigsaw puzzle, per-patch feature analysis, etc.) [40, 135, 136], or transformations of the image (rotation [58], inpainting missing regions [140], denoising [177], colourisation [86, 110, 202], or recognising distortion [44]).

A particularly influential class of self-supervised methods is contrastive learning [29, 153], which uses a learning objective that draws semantically similar sample pairs closer in the latent space and pushes dissimilar pairs apart. The core challenge is the effective construction of positive and negative pairs and formulating the contrastive loss. These approaches use heavy data augmentation to define positive and negative pairs [22], or maintain a memory queue with momentum-based negative sampling [24, 26, 74]. A prominent and commonly used image embedding models, CLIP and ALIGN, pair vision representation learning with text annotations in a vision-language contrastive learning approach, aligning image and text embeddings using a symmetric contrastive objective [88, 143]; later work extended these models with a ViT image encoder architecture [114, 115]. These approaches showcase the cross-modal alignment as foundational techniques in modern self-supervised representation learning, however, they also require providing or creating text captions for each of the dataset examples.

Another important class of methods removes the need for explicit positive and negative pair construction. Initially, methods achieved this objective by introducing asymmetry or noise in the prediction targets [67, 199]. Later approaches include clustering-based methods [18], redundancy-reduction techniques [19], and symmetric view prediction frameworks [25]. In parallel, masked token reconstruction emerged as a highly effective strategy for self-supervised learning, drawing inspiration from masked language modelling. Methods such as Masked Autoencoders (MAE) [73], SimMIM [191], BEiT [4], and their later extensions [54, 138] pre-train Vision Transformers by reconstructing missing parts of images, achieving state-of-the-art performance of inferred representations in downstream tasks.

In the video domain, these ideas are extended by exploiting temporal dependence between subsequent frames. Initially, these methods applied temporal pretext tasks such as frame-order verification [56, 130], arrow-of-time prediction [185], and temporal colourisation [179] to encourage motion-aware features. More recent methods extend contrastive learning to video clips, leveraging instance discrimination across space and time [139, 142]. Finally, self-supervision is also implemented via future frame prediction and reconstruction task [72, 170, 182], capturing long-range dependencies in videos.

Unsupervised Representation Learning. Self-supervised learning methods have proven highly effective at producing discriminative visual representations by leveraging pretext tasks or contrastive objectives to separate instances within a latent space. These methods are explicitly designed to optimise instance-level distinguishability rather than to model the full underlying data distribution. In contrast, unsupervised representation learning often adopts a generative perspective; here, models aim to capture the data distribution, reconstructing or predicting observations rather than merely distinguishing among them. By modelling the structure of the data, generative approaches produce more general and versatile representation spaces, capable of supporting a wide range of downstream tasks, including those requiring finer-grained understanding or compositional reasoning.

A foundational class of generative models for unsupervised representation learning is the variational autoencoder (VAE) [101, 150]. These models work under the assumption that observed data is generated from latent variables drawn from a simple prior distribution (typically Gaussian). VAEs introduce an encoder network to approximate the intractable posterior distribution, enabling training via variational inference. It is trained to maximise the variational lower bound on the data likelihood (evidence lower bound), balancing reconstruction fidelity with a regularisation term that encourages the latent distribution to remain close to the prior. This formulation should lead to a smooth and structured latent space, which satisfies the representation learning criteria. Further extensions, such as β-VAE [79] imposed a stronger constraint on the latent space capacity to promote disentanglement; subsequent work proposed increasing the expressiveness of the latent space and the fidelity of generated outputs, including hierarchical VAEs [27, 175] that uses multi-scale latent variables and vector-quantised VAEs (VQ-VAE) [137] that model discrete latent codes.

A different approach to generative modelling came with the introduction of Generative Adversarial Networks (GANs) [62]. In this setup, an adversarial objective is used to train two networks: a generator, which synthesises images from random latent codes and a discriminator that attempts to distinguish real pictures from generated ones. This leads to implicit modelling of the data distribution without estimating likelihood; GANs are capable of generating high-fidelity images, unlike VAEs, which tend to be blurred due to the probabilistic regularisation of the latent space. The original setup in GANs doesn't provide an image encoder for inference; it was introduced in later research [41, 42, 45], where the encoder is trained alongside the original two networks. Other improvements focused on providing photorealistic quality of generated images [13, 99, 100], or enhancing the disentanglement of latent factors with the GAN's latent space [23]. An alternative approach suggests joining the VAE and GAN idea in adversarial autoencoders [128], which replace the KL divergence term with the adversarial objective, or VAE-GAN hybrids [109], which apply the discriminator on VAE output to improve perceptual quality.

An alternative path in unsupervised representation learning is offered by flow-based models, which use invertible neural transformations for explicit probability modelling. Normalising flows [38] construct a series of bijective mappings that transform an image into a latent vector of the same dimensionality, enabling exact computation of data likelihood via the change-of-variable formula. Other models use affine coupling layers [39] and invertible 1×1 convolutions [102] to map images to latent representations. The latent space in flow models, unlike in VAEs or GANs, is of the same dimensionality as the input; since they are trained by maximising exact log-likelihood, this approach can preserve all information and provide complete coverage of the data distribution. However, the strict requirement of mapping invertibility can limit the architectural flexibility of these models.

A major breakthrough in unsupervised generative modelling was diffusion models [161]. The Denoising Diffusion Probabilistic Model (DDPM) [81] learns data structure through a gradual denoising process. A forward diffusion process incrementally corrupts an image with noise, while a neural network is trained to reverse this process step-by-step, recovering the original input. A key milestone was the introduction of Latent Diffusion Models (LDM) [152], where a DDPM is paired with a VAE to apply the diffusion process within the lower-dimensional latent space, achieving high fidelity with lower computational complexity. Additionally, the denoising process in LDM is conditioned on text encodings provided by captions of images. Unlike the previous categories of generative modelling, diffusion models do not explicitly encode images into a latent space, but instead capture the probability distribution by learning to remove noise; research shows that the denoising network produces valuable representations of the input image [169, 194]. Recent research further demonstrates that diffusion models naturally learn rich representations and can be adapted to improve their utility for downstream tasks. One approach introduces an auxiliary encoder into the denoising process, conditioned on the diffusion timestep, enabling the model to organise information hierarchically across noise scales and produce effective features for classification with simple linear probes [131]. Another line of work uses a bottleneck between the encoder and the denoising decoder, combined with a novel-view synthesis objective, to explicitly structure the latent space and encourage the capture of pose-invariant, semantically meaningful features [84].

Similar to self-supervised models, unsupervised representation learning from videos requires models to capture additional temporal dynamics. Early approaches focused on next-frame prediction in a probabilistic VAE framework [3, 36], by decomposing video generation into separate content and motion components [172], or jointly modelling frame and optical flow generation [116]. Subsequent methods model entire video sequences, aiming at coherent video generation [70, 108]. The field of generative modelling of videos accelerated with the advent of diffusion models, here extending the denoising network to model both spatial and temporal dependencies [82], later extending the existing LDM text-to-image models with temporal extensions [11, 69, 157].

1.2.3 Multi-Object Representation Learning

Multi-object representation learning is commonly tackled in an unsupervised or weakly supervised setting, where the goal is to autonomously discover and disentangle individual objects or entities from raw visual input, both in static images and in video sequences. These models aim to organise perceptual data into structured, object-centric representations without requiring dense supervision, enabling more generalisable, modular, and interpretable reasoning about visual scenes. Over the years, numerous architectural paradigms have been proposed to tackle this task, differing in how they extract, represent, and decode object-level information while balancing interpretability, flexibility, and scalability to complex real-world data. Most of them, however, share the training objective, which is learning to reconstruct the input through a structured neural network architecture, facilitating learning per-object representations.

Inference method. Initial research in this area focuses on spatial attention-based models, which use interpretable mechanisms to localise and process individual objects in an image. The foundational model, Attend-Infer-Repeat (AIR) [53], introduced a generative variational autoencoder framework that sequentially attends to and reconstructs individual objects via a recurrent inference mechanism, learning the number, location, and identity of objects without supervision. While AIR provided interpretable object-wise decomposition and strong generalisation in simple scenes, it suffered from limited scalability and difficulty handling occlusions or overlapping objects. To address AIR's inefficiency on complex scenes, SPAIR [31] replaced the recurrent inference with a convolutional architecture that processes a dense spatial grid in parallel in a single-shot manner. SPAIR improved scalability and spatial generalisation but retained weaknesses in representing objects with irregular shapes or in cluttered scenes, especially due to the limited variability of inferred object sizes caused by the fixed size and resolution of the spatial grid. Subsequent models introduced hybrid mechanisms to mitigate these limitations. For instance, SPACE [120] introduced a two-stream framework combining parallel spatial attention for foreground objects with a scene-mixture background model. This approach retained the geometric clarity of spatial attention while enabling background decomposition, resulting in improved segmentation and scalability across varied visual domains. Generative Scene Graph Networks (GSGN) [35] expanded the spatial attention paradigm by recursively composing part-whole hierarchies into scene graphs, enabling the model to capture structured relationships among object components. Meanwhile, Generative Neurosymbolic Machines (GNM) [89] fused symbolic and distributed object representations by leveraging a probabilistic generative model structured around spatial attention, improving interpretability and sample efficiency. Further, ROOTS [21] extended spatial attention to 3D-aware settings by learning object-centric representations conditioned on camera pose and viewpoint, enabling novel view synthesis. Spatial attention-based models offer high interpretability, explicit object factorisation, and the ability to impose geometric inductive biases beneficial in structured reasoning and downstream tasks. However, common limitations include difficulty modelling background explicitly, sensitivity to object scale and shape variance, and challenges in handling dense, real-world scenes without strong priors or supervision.

Scene-mixture models provide an alternative to spatial attention-based approaches for object-centric representation learning by decomposing scenes into a set of latent components through soft masking rather than discrete spatial selection. Instead of focusing on bounding boxes, these models interpret an image as a composition of overlapping reconstructions; the final image is produced by blending these reconstructions based on learned attention masks, allowing the model to capture complex object shapes and flexible spatial layouts without enforcing rigid geometries. A preliminary approach that led to scene-mixture models was Neural Expectation Maximisation (N-EM) [66], which introduced an iterative clustering framework for discovering entities, combining EM-style updates with neural inference networks to separate objects in an unsupervised fashion, though it is limited in generative capacity and scalability. Subsequent works such as MONet [14], IODINE [65], and GENESIS [49, 50] extended this line by integrating variational inference and compositional decoding, enabling per-slot reconstructions that are combined into full images via spatial masks. These models showed substantial improvements in handling occlusions and learning disentangled object representations, especially in structured synthetic environments. A major advance came with the introduction of Slot Attention [124], which states the object encoding process as an iterative attention mechanism applied to fixed-size latent slots. The model leverages dot-product attention between learned slots and input features to softly assign visual regions to slots, followed by slot-wise updates. This architecture allows flexible, differentiable object encoding without the need for explicitly defined object positions, achieving superior reconstruction quality, particularly in scenes with ambiguous boundaries or occlusion. The learned slots serve as compact, interpretable object representations that are disentangled from each other by design. However, Slot Attention remains computationally demanding due to the iterative attention and slot update steps. It often struggles to cleanly separate individual objects in cluttered or realistic scenes, particularly when visual cues are subtle or overlapping or in the case of a large number of objects in the scene. Moreover, its performance drops significantly when scaling beyond synthetic datasets.

Temporal Dynamics. Temporal extensions of multi-object representation learning models aim to exploit the sequential character of video data to improve object discovery and representation over time. Early spatial attention models like Sequential Attend-Infer-Repeat (SQAIR) [104] introduced object discovery and propagation steps within a recurrent generative framework, allowing objects to be tracked through time by maintaining latent states per entity. While SQAIR provided clear object-level dynamics, its reliance on nested RNNs and sequential inference limited its scalability. Follow-up models such as SCALOR [92] and SILOT [32] improved dynamics modelling using learned background transitions and proposal-rejection mechanisms, but retained high computational cost and complex training dynamics. Similarly, recurrent mechanisms were applied for scene-mixture models as well [33, 197].

More recently, video extensions of Slot Attention have combined iterative object binding with slot-wise temporal dynamics. Models such as SAVi, STEVE and Loci [103, 160, 171] propagate object slots across frames while learning motion-conditioned updates, and SlotSSMs [91] replace RNNs with structured state-space models, enabling long-range memory and parallel training. On the other hand, SIMONe [95] proposed a different approach, where a parallel, fully differentiable model factorises visual sequences into disentangled object latents and temporal latents, allowing it to model entire sequences simultaneously. These innovations aim to address the limitations of recurrent approaches, such as vanishing gradients and sequential bottlenecks, while retaining the modularity and expressivity of per-object representations.

Decoder design. Most of the recent unsupervised multi-object representation learning models adopt an autoencoder framework, in which the decoder plays a crucial role in reconstructing the input image and providing a training signal through reconstruction loss. The decoder must not only faithfully reconstruct visual details but also correctly attribute and spatially place the individual object representations within the scene. Its design thus directly impacts the model's ability to learn meaningful, modular, and disentangled object representations. Early scene-mixture models [14, 49, 50, 65, 124] employed per-slot convolutional decoders whose outputs were combined using alpha-based blending, with soft spatial masks determining how each object contributed to the final image. This approach allowed for smooth composition and overlapping entities, but imposed limitations on expressiveness and fidelity, especially in the presence of complex textures or lighting effects. Additionally, such decoders tended to generate blurry reconstructions of real-world data and were often unable to disentangle object appearance from spatial context fully.

In parallel, spatial attention-based models [32, 53] employed spatial transformer networks as differentiable renderers [87]. Each object was decoded in a canonical space and then placed into the scene via an affine transformation, enabling position and scale to be explicitly modelled. While interpretable and compositional, this mechanism struggled with objects of varying aspect ratios or non-rigid shapes and required careful balancing of decoder capacity and inductive bias. More recent hybrids like SPACE [120] combined the benefits of both paradigms by using spatial transformers for foreground objects and soft blending for complex backgrounds.

To improve reconstruction fidelity and scalability, a number of later models replaced these simple decoders with more powerful transformer-based architectures [158, 159, 160]. By using slot-conditioned transformers, where a pre-trained discrete VAE (dVAE) tokenises the image into visual codes, and a transformer decoder autoregressively reconstructs these tokens from object slots, these models are capable of learning object representations that align with high-level semantics and generating higher-quality images, particularly in natural scenes. However, the shift from pixel-space to token-space reconstruction introduced architectural complexity, increased reliance on pre-training, and often led to decreased interpretability due to entangled token-slot associations.

More recently, latent diffusion models have emerged as a powerful generative mechanism for object-centric learning [90, 188]. Here, models condition a diffusion process on structured slot representations, learning to denoise latent image features in a way that respects the underlying object decomposition. These approaches offer state-of-the-art image quality and robustness to complex textures while preserving slot modularity. Yet, diffusion models introduce substantial computational overhead, depend heavily on pre-trained models, and obscure the connection between slots and localised image regions due to their global conditioning and iterative nature.

Supervision in object-centric learning. The majority of the aforementioned research focuses on unsupervised learning from raw visual input; however, it has become increasingly clear that achieving robust performance on complex, real-world datasets requires some form of external supervision or inductive signal. One common strategy is to introduce conditioning signals in video models, such as motion cues derived from optical flow or spatiotemporal consistency constraints, which help maintain object identity across frames and improve segmentation under motion and occlusion [103]. Similarly, motion segmentation masks were used as a weak supervision signal, improving object discovery in challenging real-world datasets [5, 6]. Another approach suggests using depth information provided in RGB-D images as a means of supervision, offering strong geometric cues that guide object separation when appearance features alone are ambiguous [48].

One strategy to improve the performance of object-centric models is to incorporate pre-trained components into their architecture, particularly powerful feature extractors. This enables the use of powerful vision transformer backbones, which are either kept frozen [155] or fine-tuned [90, 96, 188] during object-centric training, showing improved results, especially on complex, real-world data. Similarly, latent diffusion-based models rely on pre-trained auto-encoder, trained either on large-scale image datasets [90] or on the target object-centric dataset [188], to provide a versatile latent space for the diffusion process. This approach effectively decouples low-level feature extraction from object-centric reasoning, enabling more scalable and data-efficient learning.

Self-supervised learning objectives have also been used as an effective means of regularising slot representations for learning object-centric representations in images. These include techniques such as enforcing consistency across augmentations [155], or leveraging teacher-student setups to reinforce instance segmentation [96]. In video settings, self-supervised approaches include learning to segment, transform, and inpaint objects in a scene [10], masked slot attention mechanism guided by fused semantic features and feature correspondence [141], incorporating patch-level temporal similarity derived from self-supervised vision transformer features to guide slot assignment across frames [198], and reconstructing high-level semantic features from masked inputs using spatial-temporal slot binding and dynamic slot merging [2].

Collectively, these approaches address the poor scalability of unsupervised models to real-world scenes with background clutter, intra-object variability, and ambiguous spatial cues. By incorporating some forms of supervision, ranging from motion and depth priors to feature-level self-supervision and minimal external hints, recent models have achieved substantial gains in segmentation quality and object decomposition, which promises a higher quality of the internal representations.

1.2.4 Object-Centric Datasets

Training and evaluating object-centric representation models requires datasets that depict scenes containing multiple objects of interest, ideally with variation in types, position and appearance. Because these models are typically generative and aim to model the underlying data distribution, one of the key requirements is a satisfactory large number of diverse samples to support robust generalisation. Early, foundational works focused on demonstrating a basic feasibility of automatic object discovery and encoding using simple synthetic data, where basic shapes or digits are placed randomly on plain, uniform backgrounds. These toy datasets enable testing in controlled conditions, allowing for detailed ablation studies to assess the influence of specific data properties, such as the number and sizes of objects in images. As the field progressed, more complex synthetic datasets were introduced, incorporating textured 3D objects, cluttered layouts of scenes, occlusion, complex background and camera motion in videos. In parallel, there is a growing interest in applying object-centric models to real-world images and videos, where annotations enable evaluation through downstream tasks such as object tracking, segmentation, or visual question answering.

Early Toy Datasets. Early methods, which pioneered the development of multi-object representation learning, relied on toy datasets, which offer controlled environments to test the capability of these models to discover and encode individual objects without supervision (Figure 1.3). Common examples include multi-object variants of MNIST digits [53] or 2D shapes (Multi-dSprites [14] and Tetrominoes [65]), randomly placed on a blank or monocolour background, or very simple 3D scenes (Objects Room [14]).

In case of videos, researchers used sequences of these images, where objects moved randomly within the frame. These datasets typically feature a small number of simple objects, minimal variation in appearance and no background clutter. The synthetic nature of these datasets allowed researchers to generate large datasets, validate the functioning of the methods and perform fine-grained evaluations, including detailed tests of objects' discovery, effects of changes in objects' count, size and overlap. Despite their simplicity, toy datasets remain a valuable diagnostic tool for this category of models.

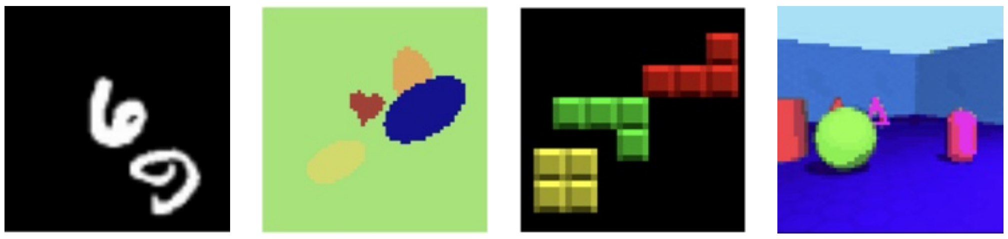


Figure 1.3: Examples of toy datasets (left to right): scattered MNIST, Multi-dSprites, Tetrominoes, and Objects Room

Synthetic Datasets. With progress in object-centric representation learning research, more sophisticated synthetic datasets were introduced to analyse how these models behave in more complex visual scenes (Figure 1.4). One of the key contributions is CLEVR [93], which includes 3D-rendered scenes with multiple geometric objects, varying in shape, colour and position. Together with its extension, CLEVRTex [98] (which adds textured materials and backgrounds), these datasets expand on the earlier setups by incorporating visual complexity and providing annotations, such as contextual questions and answers, spatial relations and detailed object poses' descriptions. Similar to toy datasets, video-based extensions such as CATER [59] and CLEVRER [195] build upon their image counterparts, focusing on object occlusions and temporal questions, respectively. One of the most commonly used synthetic benchmarks is MOVi [64]. It features several datasets with diverse 3D objects, realistic textures, dynamic lighting, camera motion and detailed annotations. These datasets attempt to approximate real-world challenges more closely, while retaining the benefits of synthetic data generation.

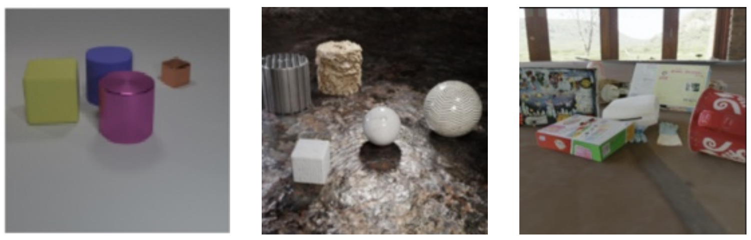


Figure 1.4: Examples of synthetic datasets (left to right): CLEVR, CLEVRTex, and MOVi-C

Real-World Datasets. The transition to real-world datasets is essential for assessing whether object-centric representations can be applied to practical problems. Real-world photos and videos are far more difficult for unsupervised models to comprehend, as they feature more complicated textures, uncontrolled lighting, and a high diversity of objects, making it difficult to attend to individual objects precisely (Figure 1.5). Object-centric models have begun to show potential on these data only in recent years, driven by advances in model scalability, decoder design and supervision strategies. However, the size of real-world datasets can limit a model's ability to learn a generalisable representation of the data without overfitting: unlike synthetic datasets, which can be generated at scale, real-world image and video collections often are not large enough given their diversity, which makes it challenging for generative models to capture their full complexity. The choice of dataset in research is usually guided by the intended downstream application of learned representations. For example, datasets like MS COCO [119] are commonly used to evaluate qualitatively the fidelity of reconstructed images, or assess scene understanding capabilities through visual question answering task [63]. For generative tasks such as image editing, face-centric datasets like CelebA [123] or FFHQ [100] were used, while the MOT dataset [111] can be utilised as a benchmark for object tracking in real-world scenarios. In many of these setups, one of the persisting challenges is the scarcity of annotations, limiting the evaluation of these models beyond reconstruction quality only to human-annotated samples.

Figure 1.5: Examples of real-world datasets (left to right): COCO, FFHQ, and MOT

1.2.5 Multi-Object Representation Learning Models Training

Most multi-object representation learning models are trained under an autoencoder framework, where the objective is to reconstruct the input image as accurately as possible from a compact, object-centric latent representation. This reconstruction-driven training encourages the model to extract meaningful features for each entity in the scene, with the latent bottleneck prioritising compactness and interpretability. A critical role is played by the decoder design, which explicitly utilises the defined structure of the latent space to create reconstructions. By requiring object representations to be composable into a coherent scene, these models are not only trained to compress visual information but also to organise it in a way that reflects the semantics of the underlying objects.

Autoencoders. The simplest conceptual approach to training multi-object representation learning models relies on plain reconstruction error, following the original autoencoder framework setup. Here, the model consists of an encoder e_ϕ that maps the image to a latent vector $\mathbf{z} = e_\phi(\mathbf{x})$, and a decoder d_θ that maps this intermediate representation to an image reconstruction $\hat{\mathbf{x}} = d_\theta(\mathbf{z})$. In the context of multi-object representation learning, the latent representation is structured, consisting of object-centric latent vectors $\mathbf{z} = (\mathbf{z}_1, ..., \mathbf{z}_K) \in \mathcal{Z}^K$

The model is trained to minimise a reconstruction loss between the original and reconstructed image. The most commonly used objective is the mean squared error (MSE) (Equation (1.1)).

$$\mathcal{L}_{\text{MSE}} = \frac{1}{N} \left\| \mathbf{x} - d_\theta \left(e_\phi(\mathbf{x}) \right) \right\|_2^2 \tag{1.1}$$

where N is the number of pixels in the image.

With the representation bottleneck of the encoder-decoder architecture, autoencoders compress the encoding of the input as a byproduct of learning to reconstruct. This setup is agnostic to the structure of the latent space, which means that object-wise decomposition emerges only from the network design, especially the decoder, which must meaningfully consider each object's appearance when creating the reconstruction.

Variational Autoencoders [101, 150]. Variational autoencoders (VAEs) extend the basic autoencoder framework by introducing a probabilistic generative model, defining a distribution over latent variables. It is assumed that the data $\mathbf{x}$ is generated by a two-step stochastic process: first, a latent variable $\mathbf{z}$ is sampled from a prior distribution $p(\mathbf{z})$ (typically standard normal distribution $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$); then the observed data is generated by sampling from a conditional likelihood $p_\theta(\mathbf{x} \mid \mathbf{z})$, parametrised by the decoder d_θ. In this setup, inferring $\mathbf{z}$ given $\mathbf{x}$ would require computing the true posterior $p(\mathbf{z} \mid \mathbf{x})$, which is intractable (it involves computing marginal likelihood, requiring integration over all latent configurations). VAE approximates the posterior using the encoder network e_ϕ (introducing a variational distribution $q_\phi(\mathbf{z} \mid \mathbf{x})$), which enables training by maximising the evidence lower bound (ELBO) (Equation (1.2)).

$$\mathcal{L}_{\text{ELBO}} = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log p_\theta(\mathbf{x} \mid \mathbf{z}) \right] - D_{\text{KL}} \left(q_\phi(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z}) \right) \tag{1.2}$$

where

> $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta\left(\mathbf{x} \mid \mathbf{z}\right)\right]$ is the expected log-likelihood of the data under the decoder $p_\theta\left(\mathbf{x} \mid \mathbf{z}\right)$, encouraging accurate reconstructions,
>
> $D_{\mathrm{KL}}\left(q_\phi(\mathbf{z} \mid \mathbf{x}) \parallel p\left(\mathbf{z}\right)\right)$ is the Kullback-Leibler divergence between the approximate posterior and the prior, constraining the expressiveness of the latent space and penalising divergence from the prior.

To allow gradient-based optimisation through the stochastic sampling of latent variables, VAE's employ the reparametrisation trick, where instead of sampling $\mathbf{z} \sim \mathcal{N}\left(\boldsymbol{\mu}_\phi, \boldsymbol{\sigma}_\phi^2\right)$ directly, the latent variable is reparametrised as in Equation (1.3), allowing gradients to propagate during training.

$$\mathbf{z} = \boldsymbol{\mu}_\phi + \boldsymbol{\sigma}_\phi \odot \epsilon, \quad \epsilon \sim \mathcal{N}\left(\mathbf{0}, \mathbf{I}\right) \tag{1.3}$$

where both $\boldsymbol{\mu}_\phi$ and $\boldsymbol{\sigma}_\phi$ are the outputs of the encoder network e_ϕ.

Latent Diffusion Models [152]. Motivated by the improved generative capabilities, recent research in the area of object-centric learning explores the idea of using latent diffusion models conditioned on object representations. Denoising Diffusion Probabilistic Models (DDPMs) [161] learn to synthesise data by reversing a gradual noising process. They rely on applying a forward diffusion process to transform a complex data distribution into a simple prior (usually standard Gaussian); then, a neural network is trained to iteratively map the noise back to the data distribution by approximating the reverse process. Formally, given a data sample $\mathbf{x}^{(0)}$, the forward process is defined as a Markov chain that corrupts it with Gaussian noise over T steps (Equation (1.4)). The reverse generative process models $p_\theta\left(\mathbf{x}^{(t-1)} \mid \mathbf{x}^{(t)}\right)$, parametrised with a noise prediction network $\epsilon_\theta\left(\mathbf{x}^{(t)}, t\right)$, estimates the added noise at each step t. In this setup, sampling data involves iteratively denoising from random Gaussian noise $\mathbf{x}^{(T)}$ to $\mathbf{x}^{(0)}$.

$$q\left(\mathbf{x}^{(t)} \mid \mathbf{x}^{(t-1)}\right) = \mathcal{N}\left(\mathbf{x}^{(t)}; \sqrt{1-\beta_t}\mathbf{x}^{(t-1)}, \beta_t\mathbf{I}\right) \tag{1.4}$$

where β_t is a variance schedule at timestep t.

The high dimensionality of the image space makes training DDPMs costly and slow, which is why Latent Diffusion Models (LDMs) [152] utilise a lower-dimensional latent space to perform the diffusion process. They incorporate a pre-trained autoencoder capable of encoding the data $\mathbf{x}$ to a learned latent space $\mathbf{z}^{(0)} = e\left(\mathbf{x}\right)$, as well as reconstructing the image from this latent encoding $\hat{\mathbf{x}} = d\left(\mathbf{z}^{(0)}\right)$. Then, the diffusion process is applied to the latent representation $\mathbf{z}^{(0)}$, producing its noisy version $\mathbf{z}^{(t)}$, whereas the reverse process utilises a conditional model to denoise the noisy encoding (Equation (1.5)).

$$p_\theta\left(\mathbf{z}^{(t-1)} \mid \mathbf{z}^{(t)}\right) = \mathcal{N}\left(\mathbf{z}^{(t-1)}; \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{z}^{(t)} - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\hat{\boldsymbol{\epsilon}}^{(t)}\right), \beta_t\mathbf{I}\right)$$
$$\hat{\boldsymbol{\epsilon}}^{(t)} = \epsilon_\theta\left(\mathbf{z}^{(t)}, t, e_\phi^{\text{aux}}(y)\right) \tag{1.5}$$

where:

$\hat{\boldsymbol{\epsilon}}^{(t)}$ is the noise predicted at timestep t,
$\epsilon_\theta\left(\mathbf{z}^{(t)}, t, e_\phi^{\text{aux}}\right)$ is the output of the noise prediction network,
$e_\phi^{\text{aux}}(y)$ denotes an encoder of auxiliary input y, used for conditional image generation,
β_t is a variance schedule value at timestep t,
$\alpha_t = 1 - \beta_t$,
$\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$.

During training, a random timestep t is sampled uniformly, and the latent $\mathbf{z}^{(0)}$ is noised as in Equation (1.6).

$$\mathbf{z}^{(t)} = \sqrt{\bar{\alpha}}\mathbf{z}^{(0)} + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}^{(t)}, \quad \boldsymbol{\epsilon}^{(t)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \tag{1.6}$$

The network ϵ_θ is trained to predict the noise $\boldsymbol{\epsilon}$ from $\mathbf{z}^{(t)}$ and t, conditioned on the input from the encoder $e_\phi^{\text{aux}}(y)$, by minimising the mean squared error (Equation (1.7)). The encoder e_ϕ^{aux} can be optimised jointly with the noise prediction network ϵ_θ, or frozen when using a pre-trained domain-specific expert network (e.g. text encoder for text-based conditioning).

$$\mathcal{L}_{\text{LDM}} = \mathbb{E}_{\mathbf{x},\epsilon\sim\mathcal{N}(\mathbf{0},\mathbf{1}),t,y}\left[\left\|\epsilon - \epsilon_\theta\left(\mathbf{z}^{(t)}, t, e_\phi^{\text{aux}}(y)\right)\right\|_2^2\right] \tag{1.7}$$

Conditioning is implemented via cross-attention layers embedded in the noise prediction network ϵ_θ. The auxiliary input is mapped to a conditioning embedding $\mathbf{r}^{\text{aux}} = e_\phi^{\text{aux}}(y)$, which is then used to produce key-value pairs for attention (Equation (1.8)), allowing the model to modulate denoising based on the conditioning signal.

$$\text{Attn}\left(Q, K\left(\mathbf{r}^{\text{aux}}\right), V\left(\mathbf{r}^{\text{aux}}\right)\right) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{1.8}$$

where:

Q are the internal feature queries,
$K(\mathbf{r}^{\text{aux}}), V(\mathbf{r}^{\text{aux}})$ are the keys and the values – learned projections of $\mathbf{r}^{\text{aux}}$,
d_k is the dimensionality of the key/query vectors.

1.2.6 Evaluation of Multi-Object Representations

Evaluation of the quality of object-centric representations is a critical part of multi-object representation learning research. In the context of object-centric modelling, it often relies on downstream tasks; these models are typically trained without supervision, generally using the reconstruction error objective, which makes evaluation more challenging than in the case of supervised learning. Because the goal is to learn structured, object-centric representations of input visual data, evaluation generally involves assessing the alignment between these representations and semantically meaningful properties of the objects (appearance, location, dynamics, etc.) relevant to established benchmark problems. Among the most commonly used downstream tasks for evaluating multi-object representations are:

- **object counting** and **classification**, estimating the number of distinct objects and assigning semantic labels based solely on learned representations,
- **object detection** and **segmentation**, which emerge as a part of object representation in spatial attention and scene-mixture models, respectively,
- **object tracking**, focusing on maintaining object identity over time in video sequences, based on objects' representations in consecutive frames,
- **visual question answering (VQA)**, which involves answering queries about objects and their relations using the learned representations,
- **visual reasoning**, utilising scene representation to infer properties or outcomes such as the location of occluded objects,
- **novel scene synthesis**, which applies the model's decoder to reconstruct the image or video based on provided object encoding, testing how modifying the representation alters the synthesised scene.

1.2.6.1 Reconstructions Quality

Reconstruction error is often used as the objective during training multi-object representation learning models, which is why the quality of reconstructions is one of the most critical means of evaluating these models. Besides assessing the similarity of the generated output to the input image or video, reconstruction metrics provide a quantitative measure of reconstruction fidelity and realism.

- **Mean Squared Error (MSE)** is a standard metric used to measure the difference between predicted values and actual values in the dataset (Equation (1.9)).

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{x}_i)^2 \tag{1.9}$$

  where:

  N is the number of observations,
  x_i refers to the true value of the i-th observation,
  $\hat{x}_i$ is the predicted value of the same observation.

  In the case of images and videos, observations refer to pixels and their intensities. A lower MSE value means that the prediction is closer to the actual values, indicating high accuracy of input reconstruction. While commonly used in multi-object representation learning, MSE doesn't explicitly evaluate the quality of representations. Models can achieve low MSE focusing on pixel-level accuracy without inferring structured and meaningful representations.

- **Fréchet Inception Distance (FID) [78] and Fréchet Video Distance (FVD) [174]** are metrics used to evaluate the generative capability of models such as Generative Adversarial Networks. Unlike pixel-wise metrics, FID and FVD assess the similarity between the distribution of real and generated images (Equation (1.10)). Specifically, they compare the mean and covariance of feature representations extracted from a pre-trained Inception v3 network [168] (in the case of FVD, Inflated 3D ConvNet [20] is used as the feature extractor).

$$d_{\text{FID,FVD}}(P_r, P_g) = \|\mu_r - \mu_g\|^2 + \text{Tr}\left(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{\frac{1}{2}}\right) \tag{1.10}$$

  where:

  μ_r, Σ_r are the mean and covariance of real input features distribution P_r,
  μ_g, Σ_g are the same parameters of the generated output features distribution P_g.

A lower FID score indicates that the feature distribution of the generated images or videos is more similar to that of the real data, suggesting that the model has captured the underlying data distribution. However, similar to MSE, this metric does not explicitly evaluate the latent space of the model.

- **Learned Perceptual Image Patch Similarity (LPIPS)** [203] addresses the challenge of perceptually similar images, which have high pixel-wise error due to slight shifts, colour variations, or texture differences. It measures the perceptual similarity between two images, using a pre-trained feature extractor (such as AlexNet [105] or VGG [156]) and computes the ℓ_2 distance between normalised features at multiple layers (Equation (1.11)).

$$d_{\text{LPIPS}}(\hat{\mathbf{x}}, \mathbf{x}) = \sum_l \frac{1}{H^{(l)} W^{(l)}} \sum_{h,w} \left\| \mathbf{w}_l \odot \left(\mathbf{y}^l_{hw} - \hat{\mathbf{y}}^l_{hw} \right) \right\|_2^2 \tag{1.11}$$

where:

$\mathbf{y}^l, \hat{\mathbf{y}}^l$ are unit-normalised features extracted using the pre-trained network for layer l from the real input $\mathbf{x}$ and the generated output $\hat{\mathbf{x}}$, respectively,
$H^{(l)}, W^{(l)}$ denote the dimensions of l-th layer's output,
$\mathbf{w}_l$ is a trainable weights vector used to scale the activations for l-th layer data, reflecting perceptual importance of each layer.

A lower LPIPS score indicates higher perceptual similarity. Unlike FID/FVD, which assesses distribution-level similarity across a dataset, LPIPS is a per-image metric.

1.2.6.2 Entity Classification

One of the most fundamental downstream tasks used to assess learned object-centric representations is classification. In the most straightforward scenario, each object within the scene is assigned a class label from a predefined set, based on ground-truth annotations. These labels can correspond to various attributes of objects, including object category, colour, size, material, or spatial properties. They may also encode relational properties, such as whether the object is interacting with another one in a particular way. If the model is capable of learning semantically meaningful representations, even a simple classifier should be able to accurately map the latent representation to the correct labels.

Beyond basic object-level classification, more complex reasoning tasks, which assess scene-level understanding, can also be framed as classification tasks. For example, in visual question answering with a closed set of answers, the model has to classify which answer applies given a query and scene representation. Similarly, in scene-level reasoning tasks, the model may assign high-level labels to the entire scene. These scenarios can be used to validate if, beyond capturing isolated object properties, the model is also capable of understanding the global scene structure and object interactions.

- **Accuracy** measures the proportion of correct predictions over all predictions (Equation (1.12)).

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{1.12}$$

  where:

  TP are true positive examples,
  TN are true negative examples,
  FP are false positive examples,
  FN are false negative examples.

- **Precision** measures the proportion of correctly predicted positive instances among all predicted positives (Equation (1.13)).

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{1.13}$$

- **Recall** measures the proportion of actual positive instances that were correctly predicted (Equation (1.14)).

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{1.14}$$

- **F1-Score** is the harmonic mean of precision and recall, which balances these measures especially for skewed datasets (Equation (1.15)).

$$\text{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{1.15}$$

In a multi-class setting, these metrics need to be aggregated across classes. The most common strategies involve:

- micro-averaging, which aggregates the contributions of all classes by considering total count of TP, TN, FP and FN,

- macro-averaging, which computes metrics for each label, and returns the average value,

- weighted-averaging, which considers the proportion for each label in the dataset when averaging per-label metrics.

1.2.6.3 Semantic Segmentation

Scene-mixture models use a structured latent space to encode both objects' appearance, as well as their location, using segmentation masks to indicate the spatial support of each entity in the scene. These masks emerge from trained attention mechanisms, assigning pixels to distinct object-centric slots. When ground-truth segmentation masks are available, the encoded attention masks can be evaluated using semantic segmentation metrics. This is particularly useful from the perspective of models' capability to identify and separate individual entities within the scene. A strong alignment between encoded masks and ground truth segmentation indicates that the model learned to discover object boundaries and group pixels meaningfully. This spatial disentanglement is one of the key properties of multi-object representation learning.

- **Mean Intersection over Union (mIoU)** is one of the most widely used metrics for evaluating spatial segmentation equality. Intersection over Union measures the overlap between two sets (Equation (1.16)).

$$\mathrm{IoU}\left(P, G\right) = \frac{|P \cap G|}{|P \cup G|} \tag{1.16}$$

  where:

  $P \subset \Omega$ is the predicted mask,
  $G \subset \Omega$ is the ground-truth mask,
  Ω is the set of all pixels in the image.

  The metric takes values between 0 (no overlap) and 1 (perfect alignment of masks). To evaluate the entire image segmentation, IoU is calculated between each predicted mask and the ground truth in order to get the optimal one-to-one matching, maximising the total IoU. Then, the Mean Intersection over Union is calculated as the mean over matched pairs.

- **Mean Best Overlap (mBO)** follows a similar approach to mIoU, but instead of getting an optimal one-to-one matching, it assigns each ground-truth object mask to the max-overlapping predicted mask (Equation (1.17)).

$$\mathrm{BO}\left(P, G_i\right) = \max_j \mathrm{IoU}\left(P_j, G_i\right) \tag{1.17}$$

  where:

G_i is the i-th ground-truth mask,
P_j is the j-th predicted mask.

The Mean Best Overlap is computed by averaging the BO scores across all ground-truth objects. This approach allows each ground-truth object to be matched to its best-overlapping predicted mask, properly handling scenarios where the number of predicted objects does not match the ground truth.

- **Adjusted Rand Index (ARI)** is another metric used for evaluating segmentation performance. Unlike mIoU, which measures pixel-level overlap between matched masks, ARI evaluates overall agreement between two clusterings of the image, making it particularly useful for object-centric masks, which lack class assignments. It extends the Rand Index [145], which quantifies between two clusterings by considering all pairs of pixels and counting how many pairs are consistently grouped in both true and predicted segmentations. Given all combinations of n pixels in an image ($\binom{n}{2}$ combinations), the Rand Index is calculated as in Equation (1.18).

$$\mathrm{RI} = \frac{a+b}{\binom{n}{2}} \tag{1.18}$$

where:

a is the number of pairs of pixels covered by the same mask both in ground-truth and predicted segmentation (true positives),
b is the number of pairs of pixels covered by different masks in both segmentations (true negatives).

Adjusted Rand Index includes correction for chance, subtracting its expected value under random labelling and normalising the results (Equation (1.19)).

$$\mathrm{ARI} = \frac{\mathrm{RI} - \mathbb{E}[\mathrm{RI}]}{1 - \mathbb{E}[\mathrm{RI}]} \tag{1.19}$$

where:

$\mathbb{E}[\mathrm{RI}]$ is the expected Rand Index (average similarity between two random clusterings).

ARI ranges from -1 to 1, where 1 indicates a perfect clustering match, 0 corresponds to chance-level agreement, and values lower than 0 are worse than random matchings. Importantly, ARI is invariant to permutation, making it especially useful in evaluating object-centric models, where representation order is arbitrary. In this context, researchers usually utilise FG-ARI, a variant which only considers foreground objects and ignores false segmentation of the background.

1.2.6.4 Object Detection

Spatial attention-based multi-object representation learning models use a different approach for representing object location. Instead of segmentation masks, their structured encodings typically include bounding boxes that indicate the spatial extent of each discovered entity. Similar to scene-mixture models, evaluating how well these spatial components align with ground-truth object locations provides insight into the model's ability to attend to and disentangle meaningful entities within the scene. In this setting, the quality of object localisation is assessed using object detection metrics.

- **Average Precision** is a widely used metric for measuring the accuracy of predicted object locations. To evaluate the performance in object detection, it determines for each predicted bounding box if it qualifies as a true or false positive based on its Intersection over Union with any unmatched ground-truth box. By using various thresholds of IoU, it builds a precision-recall curve for all predicted bounding boxes. In this setup, Average Precision is defined as the area under the precision-recall curve. In practice, it is approximated by using discrete samples of recall levels, AP computed as a weighted sum of precisions at each threshold with an increase in recall as weight (Equation (1.20)).

$$\text{AP} = \sum_{k=0}^{N-1} \Big(\text{Recall}_{\tau_k} - \text{Recall}_{\tau_{k+1}}\Big) \cdot \text{Precision}_{\tau_k} \tag{1.20}$$

  where:

  N is the number of thresholds,
  τ_k is the k-th threshold, at which Recall_{τ_k} and $\text{Precision}_{\tau_k}$ is computed.

  Typically, in an object detection task, Average Precision is aggregated over all available classes, forming Mean Average Precision. However, in object-centric models, spatial attention only provides boxes' locations, without classifying them, hence, AP can be applied in a class-agnostic fashion.

1.2.6.5 Multi-Object Tracking

One of the downstream tasks utilised to evaluate the temporal stability and consistency of learned object-centric representations in videos is object tracking. In this setting, the model is expected to maintain coherent representations of individual objects across frames as they move or undergo minor appearance changes. A common evaluation approach involves integrating these representations into a tracking-by-detection framework similar to Deep SORT [186]. In this framework, object-centric representations are used as objects' embeddings, and later associated across frames based on an appearance similarity metric. This setup enables evaluating whether the learned objects' representations remain stable and temporally consistent, providing insight into the model's ability to support dynamic scene understanding.

- **Multiple Object Tracking Accuracy (MOTA)** [9] is one of the popular metrics for evaluating multi-object tracking performance. MOTA relies on qualifying tracked objects as false negatives (missed detections), false positives (spurious detections), and identity switches to determine tracking accuracy (Equation (1.21)).

  $$\text{MOTA} = 1 - \frac{\sum_t (\text{FN}_t + \text{FP}_t + \text{MME}_t)}{\sum_t \text{GT}_t} \tag{1.21}$$

  where:

  t is the timestep,
  MME is the number of mismatches (identity switches),
  GT is the number of ground-truth objects.

  MOTA can be less than 0, indicating poor tracking performance. Values above 0.5 are considered acceptable, while 1 means perfect tracking.

- **Identification F1-score (IDF1)** [151] is a metric focusing on object identity preservation in multi-object tracking. It measures the consistency of predicted identities with respect to ground truth by computing the F1-score in the identity matching problem. Here, the metric is computed by creating a one-to-one matching between ground-truth and predicted object trajectories, which minimises mismatched frames. Then, true positive, false positive and false negative examples are counted to calculate the F1-score (Equation (1.15)). In the multi-object representation learning setup, IDF1 allows focusing on identity consistency instead of raw detection accuracy, providing more insight into association quality across time.

- **Higher Order Tracking Accuracy (HOTA)** [127] is a more recent metric that aims to unify detection, association and localisation performance in a single interpretable score, addressing the limitation of MOTA and IDF1, which overemphasise detection and association accuracy, respectively. For a given localisation threshold α, it is computed as the geometric mean of detection and association accuracy (Equation (1.25)). Detection accuracy DetA_α is calculated by determining if a prediction is a true or false positive, similarly to Average Precision

(Equation (1.22)), whereas Association accuracy score (AssA_α) involves counting true positive associations (TPA), false negative associations (FNA) and false positive associations (FPA) (Equations (1.23) and (1.24)).

$$\mathrm{DetA}_\alpha = \frac{\mathrm{TP}_\alpha}{\mathrm{TP}_\alpha + \mathrm{FN}_\alpha + \mathrm{FP}_\alpha} \tag{1.22}$$

$$\mathrm{AssA}_\alpha = \frac{1}{\mathrm{TP}_\alpha} \sum_{c \in \{TP_\alpha\}} \mathcal{A}_\alpha(c) \tag{1.23}$$

$$\mathcal{A}_\alpha(c) = \frac{\mathrm{TPA}_\alpha(c)}{\mathrm{TPA}_\alpha(c) + \mathrm{FNA}_\alpha(c) + \mathrm{FPA}_\alpha(c)} \tag{1.24}$$

$$\mathrm{HOTA}_\alpha = \sqrt{\mathrm{DetA}_\alpha \cdot \mathrm{AssA}_\alpha} \tag{1.25}$$

The final HOTA score is the average of the HOTA_α scores calculated at a number of different localisation thresholds α.

1.3 Motivation

As detailed in Section 1.2, effective feature extraction is fundamental for numerous computer vision tasks such as classification, detection, or visual reasoning. Conventional approaches commonly rely on global feature extraction, where the entire image is processed into a single holistic embedding. Such global representations often lack the resolution and structure required to analyse complex scenarios involving multiple interacting objects. By merging object-specific details into entangled feature vectors, these methods limit the ability to distinguish individual object properties, which is essential for human-level understanding of visual scenes. Multi-object representation learning emerges as a response to this limitation, offering a more structured, human-like understanding of visual input. These models have numerous practical applications, including their use as robust feature extractors for downstream tasks such as visual question answering, object tracking or scene understanding, which is necessary for robotics applications. Their inherent generative capabilities enable utilising them for creating new scenes based on learned representations, as well as modifying input images through object manipulation or scene editing.

Modern representation learning predominantly relies on unsupervised or self-supervised training methodologies. The use of supervised models as feature extractors involves optimising for a specific objective based on human-provided annotations. This can lead to highly specialised representations which are difficult to use in more general applications, different from the original task. Furthermore, the scarcity of large-scale annotated datasets and the risk of obtaining inaccurate annotations motivated the shift towards unsupervised approaches, capable of leveraging vast amounts of unlabeled visual data. Within this context, generative modelling emerges as particularly powerful, explicitly targeting the learning of the underlying data distribution. Unlike discriminative models, which aim at differentiating objects and their characteristics, generative models inherently capture the full complexity of the input data, facilitating richer and more flexible representations suitable for diverse downstream tasks.

During the initial literature review and preliminary research, several key research gaps were identified:

1. **Scalability of early object-centric methods.** Initial influential approaches, such as AIR [53], MONet [14], or IODINE [65], significantly advanced the idea of unsupervised multi-object representation learning. However, their reliance on sequential inference with recurrent neural network architectures resulted in high computational complexity, caused by processing one object at a time, or iteratively refining the inferred representations. Research demonstrates their performance mainly on synthetic datasets or relatively simplistic scenes, containing a small, fixed number of objects. These methods fail to generalise to more realistic, visually complex settings, which remains a key challenge in the area of unsupervised multi-object representation learning.

2. **Limitations of convolutional grid-based architectures.** Subsequent methods like SPAIR [31] and SPACE [120] aimed to address scalability through convolutional, grid-based approaches inspired by single-shot object detectors. In this setting, the model can infer objects' locations simultaneously across all image regions, instead of recurrent analysis of a single image. The inferred objects' positions are used to extract glimpses, which are processed by a shared encoder. Although this approach allows scalable processing of multiple objects in an image, the convolutional encoder design introduces a significant rigidity due to reliance on a fixed-size grid and predefined spatial discretisation. Objects substantially deviating from the most common dimensions (larger or smaller than the chosen grid resolution) could not be encoded effectively. Similarly, with each cell corresponding to a region on the image, densely clustered objects with overlaps could not be separated into distinct object representations, as multiple entities could fall within the same grid cell, leading to ambiguous object representations.

Consequently, convolutional grid-based methods struggled with realistic scenarios where object scales, shapes, positions, and densities vary, negatively impacting their practical utility.

3. **Inadequate object disentanglement in fully unsupervised setups.** Purely unsupervised multi-object representation learning methods frequently struggled to reliably disentangle scenes into distinct object representations, especially when used on real-life images. Without explicit supervision or targeted inductive biases, these models tended to collapse multiple distinct objects into a single, overly coarse latent representation rather than clearly disentangling each entity. In the case of scene mixture models, this behaviour was demonstrated by creating large masks adhering to regions in images or colours rather than individual objects; in the case of spatial attention models, they tend to reconstruct the input image by splitting it into rectangular patches instead of focusing on individual objects. This behaviour was particularly prominent in complex, cluttered, and realistic datasets where individual object boundaries are more difficult to differentiate visually, limiting the versatility of the resulting representations. Thus, later multi-object representation learning models introduced additional inductive biases or minimal supervision to achieve meaningful decomposition, addressing a critical gap between fully unsupervised representation objectives and the practical demand for precise, interpretable object-centric representations.

4. **Temporal inconsistency in video models.** Temporal extensions of object-centric models, designed to handle video data and dynamic visual contexts, inherited and amplified the computational complexity and architectural limitations observed in static-image counterparts. Video-based methods often rely heavily on sequential recurrent inference mechanisms to track objects over multiple frames, making them computationally expensive and challenging to scale beyond short temporal sequences. Additionally, these models typically lacked explicit mechanisms for enforcing long-term temporal consistency or stable object identity preservation across frames. Consequently, these approaches frequently exhibited undesirable behaviours, such as drift in object representations over time, and inconsistent segmentation across consecutive frames, limiting their applicability in realistic dynamic scenarios. Addressing these temporal limitations requires efficient temporal modelling strategies based on a robust image representation learning model, capable of reliably maintaining coherent and stable object-centric representations throughout extended video sequences.

Given how the original grid-based approaches were inspired by advancements in object detection frameworks, particularly single-shot detection methods [146, 122], it is natural to hypothesise that subsequent improvements from modern object detection methods could enhance multi-object representation learning models. Multi-scale feature maps and anchor boxes, which allowed these object detection models to better attend to objects of varying sizes, densely clustered and overlapping, could lead to better disentanglement of object representations in the object-centric learning setting, improving the quality of their embeddings. Similarly, more powerful convolutional backbones could be used to enhance the effectiveness of initial feature extraction in these models. Finally, the original glimpse-based object encoding could be replaced by extending the convolutional grid-based encoding to comprehensive object attribute encoding, without the need for an additional glimpse encoding step. Such integrated, parallel encoding strategies inspired by state-of-the-art detection models could result in more efficient, scalable, and robust multi-object representation learning approaches capable of handling realistic and complex visual scenes.

1.4 Research Goals

The general aim of this research is to advance the multi-object representation learning methods in complex visual environments, improving the scalability of unsupervised generative methods. This section outlines the detailed plan for the research, formulates the central hypothesis and specifies the research questions that guide the investigation.

1.4.1 Research Plan

The research conducted in this thesis, through a series of stages, presents the development and evaluation of novel approaches to object-centric representation learning across both image and video domains. The initial stage of the research (Section 2) involves investigating the internal feature maps of pre-trained object detection neural network, particularly a simplified YOLO [146], for their usability as inputs to deep reinforcement learning agents. This includes a comparison of different levels of visual abstraction, including two early exits from the convolutional backbone, and final detection outputs, with the aim to determine their utility for facilitating navigation tasks in autonomous underwater vehicles. In this setup, the experimental scenario includes training and evaluation of separate models based on each level of visual features.

Building upon this approach, the first core phase of the research (Section 3) investigates the main research gap identified in the area of spatial attention, multi-object representation learning models utilising convolutional grid-based object encoding. The challenge of learning structured representations of diversely-sized objects without relying on sequential or recurrent mechanisms is addressed by integrating a multi-scale encoding architecture, inspired by the Single-Shot Multi-Box Detector (SSD) [122]. On the other hand, this phase explores the capability of extending existing unsupervised approaches to more complex datasets by integrating a pre-trained object detection model for localising encoded objects and improving feature extraction. Evaluating these capabilities is conducted through training models on specifically selected datasets, including objects of highly varying sizes, ranging from synthetic images to realistic photos, analyzing the quality of learned representations in downstream tasks.

The second phase (Section 4) addresses the challenge of applying these improvements in a video-based multi-object representation learning model. In an attempt to mitigate the inflexibility of strict recurrent mechanisms for discovery and propagation of objects, this stage involves reviewing an implicit approach, where instead intermediate features are propagated over time using a recurrent neural network (RNN). Following the previous phase, here a staged training procedure is used, which includes three consecutive steps: first, a supervised object detection model is trained to localise selected objects of interests; then, an image-based representation learning model is trained, incorporating encoding heads to extract object-centric embeddings; finally, a temporal extension is introduced, enabling the model to learn consistent object representations from video sequences. The experimental scenario includes preparing dedicated synthetic sequences of images, as well as complex real-life videos, to review the quality of representations and the improved temporal coherence of the objects' representations.

The final stage of the research (Section 5) explores the generative modelling techniques to improve the quality of reconstructions produced by the multi-object representation learning model. Inspired by recent advances in diffusion-based generative architectures [90, 152, 188], it explores the idea of using multi-scale object representations as conditioning in a denosing network. This phase involves utilising detection-informed object encodings, integrating the objects' location information through positional encodings and a complex loss function. The experimental setup, aside from evaluating the influence of this training setup on the resulting representations of objects, includes a review of reconstruction quality on simple and complex datasets, addressing one of the main limitations of existing methods.

A critical aspect is the comparison with selected baseline methods on the same dataset, ensuring a fair and detailed comparison. The reproducibility of the experiments is guaranteed by providing the source code for all experiments, as well as the datasets.

1.4.2 Research Hypothesis

The central hypothesis of this research is that utilising multi-scale intermediate features from single-shot object detection networks as a means for spatial grid-based attention enables the development of object-centric representations that are disentangled, scale-invariant and generalisable across images and videos. It is further hypothesised that incorporating a pre-trained object discovery mechanism enables the shift from synthetic images to real-life datasets, allowing these models to focus on important objects discovered within the scenes, and providing generalisable representations that can be effectively reused in downstream computer vision tasks.

1.4.3 Research Questions

In this thesis, the following research questions were considered.

RQ 1: *To what extent can internal representations extracted from intermediate layers of an object detection neural network serve as effective visual inputs for deep reinforcement learning agents, especially in 3D robotic navigation?* See Section 2.3.

RQ 2: *To what extent does fully convolutional spatial grid-based attention, which replaces sequential encoding or iterative refinement, enable the learning of high-quality object-centric representations in multi-object visual scenes?* See Section 3.4.

RQ 3: *How does utilising multi-scale feature maps improve the performance of object-centric representations in downstream tasks, particularly for objects of varying sizes in complex visual scenes?* See Section 3.4.

RQ 4: *How can an unsupervised learning framework that incorporates knowledge from a pre-trained object detector be used to learn structure latent variables, such as appearance and depth?* See Section 3.4.

RQ 5: *How can implicit representation-based temporal mechanisms be used to provide consistent representations for tracking object identities across video frames?* See Section 4.3.

RQ 6: *How does staged training procedure (translating from image-based to video-based representation learning) impact the ability to learn objects' representations in a video setting?* See Section 4.4.

RQ 7: *How can detection-guided object representations be used to condition diffusion-based generative models for controllable image synthesis?* See Section 5.3.

RQ 8: *How can positional information from detection models be incorporated in a diffusion-based object-centric model to improve object-to-representation matching in generative applications and complex downstream tasks?* See Section 5.3.

1.5 Thesis Contributions

The following contributions of the thesis can be enumerated:

1. Identification of the research problem: learning modular, transferable object representations from images and videos using internal feature maps of pre-trained single-shot detection networks as a structured visual prior – see Section 1.4;

2. Experimental evaluation of the applicability of visual features extracted from a pre-trained object detection network for use in deep reinforcement learning-based 3D navigation, demonstrating the feasibility of structured and abstract visual inputs as alternatives to explicit object detections [210] – see Chapter 2;

3. Design and implementation of a scale-invariant, fully parallel encoding approach for multi-object representation learning in images, which integrates spatial grid-based attention with multi-resolution features and structured latent variables [208] – see Chapter 3;

4. Introduction of SSDIR, an object-centric generative model that leverages detection-based spatial supervision while learning appearance and depth latents in an unsupervised manner, supporting compositionality and generalisation [208] – see Chapter 3;

5. Development of a temporally consistent multi-object representation learning model RDIR for videos by extending the static model with a recurrent feature encoder (Sequence encoder) and a multi-scale spatial fusion module (Mixer module), enabling consistent object representations across time and robust inference in dynamic environments [209] – see Chapter 4;

6. Formulation of an approach that uses a staged training procedure for the video-based object-centric model to improve optimisation stability [209] – see Chapter 4;
7. Design of a detection-guided object encoding pipeline for generative modelling, which includes positional disentanglement, cross-resolution fusion, and object-conditioned latent diffusion – see Chapter 5;
8. Integration of learned object-centric representations with latent diffusion models and detection-guided loss function in DetDiff to enable controllable object manipulation in image synthesis, while maintaining compositional structure – see Chapter 5;
9. Experimental validation of the proposed methods on synthetic and real-world datasets, demonstrating their effectiveness in object disentanglement, scalability, temporal coherence, and applicability in downstream tasks – see Chapters 3, 4, and 5;
10. Validation of the thesis outcomes through publications in peer-reviewed conferences and journals [210, 208, 209] – see Appendix A.

1.6 Thesis Outline

This thesis is organised into four main research chapters, each addressing a distinct phase of the research on learning object-centric representations from images and videos using detection-guided and unsupervised learning methods. Chapter 2 presents the initial stage of the research, which investigates the use of intermediate feature representations from pre-trained object detection networks as inputs to deep reinforcement learning agents for vision-based navigation tasks. This chapter describes the design of the experimental pipeline, the motivation behind using mid-level features, and the evaluation of their utility in spatial reasoning. Chapter 3 introduces a method for learning structured, multi-object representations in static images using a fully convolutional, parallel encoding approach. It focuses on the integration of multi-scale feature pyramids, structured latent variables, and an unsupervised training strategy. The chapter includes detailed experiments on object reconstruction, disentanglement, and ablation studies. Chapter 4 extends the previous method to video data by incorporating an implicit temporal modelling component. It presents a recurrent architecture that maintains encoded feature maps over time and fuses spatially structured features at multiple scales. This chapter provides an analysis of representation quality, object tracking, and the effects of object dynamics, supported by a series of ablation studies and visualisations. Finally, Chapter 5 explores the generative aspect of detection-enhanced object-centric representations. It presents a detection-guided diffusion-based approach that conditions generation on structured object latents. This chapter details the integration of positional encoding, cross-scale feature fusion, and object-aware losses and evaluates the model's effectiveness in reconstruction and downstream tasks. Chapter 6 concludes the thesis by summarising the research findings, answering the research questions, and outlining directions for future work.

Additionally, Appendix A lists selected scientific achievements, including publications, conference talks, and project participation. The *List of Figures*, *List of Tables* and *Bibliography* follow at the end of the thesis.

Chapter 2

Exploring Visual Representations for Vision-Based Deep Reinforcement Learning Navigation

This chapter presents the initial phase of the PhD research, which explored the use of deep reinforcement learning (DRL) for 3D robotic navigation of autonomous underwater vehicles (AUVs), with a particular focus on integrating visual perception into the agent's decision-making process. While the original goal was to demonstrate the feasibility of vision-based control using deep reinforcement learning, a central contribution of this work lies in investigating how various levels of visual feature embeddings, extracted at different stages of an object detection neural network, can inform navigation. Rather than limiting the input to final, human-interpretable outputs (i.e. bounding box coordinates), the study examined internal feature maps from early exits placed within the network as alternative visual representations, comparing them against an end-to-end trained convolutional feature extractor. This insight provided a foundation for later research directions and an in-depth literature review, focused on refining object representations and learning to encode multiple object instances simultaneously in the multi-object representation learning setup.

This work was originally published as [210]; I was the main author of the publication, responsible for the core concept, research methodology, experimental setup, and the preparation of the manuscript, including all figures and tables. The results and insights from this study serve as a starting point for the broader investigation of representation learning for application in downstream tasks, which is further developed in subsequent chapters.

It is important to note that the related work and methodological choices presented in this chapter reflect the state of the art at the time of preparing the original publication [210]. The field of vision-based deep reinforcement learning for robotic navigation is evolving rapidly; since then new approaches and architectures have emerged. However, the findings described in this chapter remain relevant, as they became the foundation for the continuation of this research.

In this study, four distinct types of visual embeddings were evaluated: the final object detection outputs, a separate trainable convolutional encoder, and two internal feature maps from a pre-trained object detection model. These representations were used as input to a DRL agent tasked with navigating an AUV toward a visual target. The agent's performance was evaluated in terms of average reward, episode length, inference speed, and detailed success/failure episode statistics. Section 2.1 outlines the motivation for this study and summarises related work. The proposed architecture and the methodology of visual feature extraction are described in Section 2.2. Section 2.3 presents the experimental setup, detailed results and analysis. Finally, Section 2.4 summarises and discusses the contributions.

2.1 Motivation

Autonomous Underwater Vehicles (AUVs) are a class of unmanned systems capable of operating independently in complex underwater environments. Together with Remotely Operated Vehicles (ROVs), they belong to a broader category of Unmanned Underwater Vehicles (UUVs), widely adopted in scientific exploration, commercial inspection and military operations. The ability of these systems to execute missions in hazardous or inaccessible locations makes them valuable tools in modern robotics. The foundational aspect of AUV control systems is autonomous navigation, the ability to travel reliably from one operational point to another while avoiding obstacles, reacting to environmental cues, and making real-time decisions under uncertainty [184].

Traditionally, path-planning for this class of systems utilises methods such as Dijkstra's [97] or A* [71] algorithms, or artificial potential field method [30], which offer a reliable solution for known or low-complexity environments, with a static map and limited action space, allowing for tractable optimisation. However, this methods typically fail to generalise in dynamic or high-dimensional spaces. Alternative approaches using evolutionary algorithms, including multi-objective optimisation [113], game theory-inspired models [46] and hybrid bio-inspired techniques [1], offer more flexibility but often suffer from challenging hyperparameter tuning and lack of sample efficiency. In contrast, reinforcement learning (RL) provides a model-free paradigm for learning behaviour directly from interaction with environment, and its deep learning extension (deep reinforcement learning) has demonstrated success in high-dimensional perception tasks such as robotic locomotion and control. Among many of the methods, the most common ones are using Deep Deterministic Policy Gradients (DDPG [117]): [17, 85, 187], variants of Deep Q-Networks (DQN [107, 133]): [165, 192, 205], Asynchronous Advantage Actor-Critic (A3C [132]): [15], and Proximal Policy Optimisation (PPO [154]): [162, 164]. Later advances extended DRL applications with vision-based models, where raw images or feature representations serve as inputs to policy models [106, 190, 196, 200]. However, at the time of this research, these papers were focused only on two-dimensional problems.

The aim of this research is to extend DRL-based navigation approaches from 2D to 3D robotic control tasks by leveraging structured visual information from a pre-trained object detection model. Instead of relying on an end-to-end vision pipeline that requires training a feature extractor from scratch with the agent model, a compact detection network was used to supply visual inputs that are already meaningful and easier for the DRL model to interpret. This can include both detection's bounding box, and internal feature maps that reflect higher-level abstractions of the scene. These feature maps are extracted from various layers of the detection network using an early-exit strategy, enabling analysis of how implicit object-level context can be used as a valid substitute for explicit target localisation.

The study was inspired by RoboSub, an international AUV competition that reflects real-world robotic challenges under constrained and partially observable conditions. In particular, the research involves a task of navigating toward a gate structure, simulating a waypoint transition in a mission. This scenario provides a benchmark for evaluating visual-based navigation strategies.

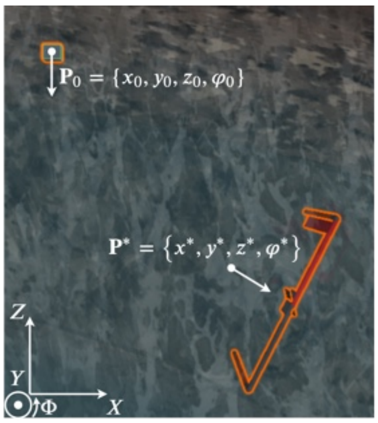
2.2 Proposed Method

The research focuses on one of the tasks from the RoboSub 2018 competition, which focuses on navigating an autonomous underwater vehicle (AUV) from the starting area (dock) towards a gate that marks the beginning of the competition task sequence. As shown in Figure 2.1, the navigation begins at point $\boldsymbol{P}_0$ (the starting location) and ends at the target point $\boldsymbol{P}^*$, situated in front of the gate. It is assumed that the gate is initially outside the robot's field of view, located several metres away from $\boldsymbol{P}_0$. The navigation objective is to reach a position where the robot is centred and facing the gate, such that it can pass through it by moving forward. The controller's goal is to steer the robot from $\boldsymbol{P}_0$ to $\boldsymbol{P}^*$ in the shortest possible time. Both points are given as four-element vectors (Equation (2.1)), relative to the centre of the competition arena.

$$\boldsymbol{P} = \{x, y, z, \varphi\} : x, y, z, \varphi \in \mathbb{R} \tag{2.1}$$

where:

x, y, z are position on axes X, Y, Z,
φ is heading angle (rotation about the vertical axis).



$\mathbf{P}_0 = \{x_0, y_0, z_0, \varphi_0\}$
$\mathbf{P}^* = \{x^*, y^*, z^*, \varphi^*\}$
Z
Y
Φ
X

Figure 2.1: Plan view of an example configuration of a starting point $\boldsymbol{P}_0$ and a destination point $\boldsymbol{P}^*$ for the examined task

To solve this navigation problem, the robot's controller has to find the best move to the target using the information delivered from sensors. The AUV is equipped with three high-definition colour cameras. Each one generates an image tensor $\mathbf{x}_{\mathrm{img}}$ of shape $1280 \times 720 \times 3$. The model utilises one of the cameras, mounted on the front of the robot and facing forward.

Apart from the cameras, the AUV utilises an AHRS (Attitude and Heading Reference System), an integrated sensor composed of a gyroscope, an accelerometer and a compass, which allows orientation and both linear and angular acceleration measurement. The controller analyses the following three-element vectors: linear acceleration vector $\mathbf{x}_a$ (Equation (2.2)), angular velocity vector $\mathbf{x}_\omega$ (Equation (2.3)) and rotation vector $\mathbf{x}_\varphi$ (Equation (2.4)) with values in the range $(-180, 180]$.

$$\mathbf{x}_a = \{a_x, a_y, a_z\}, \quad a_x, a_y, a_z \in \mathbb{R}\left[\frac{\mathrm{m}}{\mathrm{s}^2}\right] \tag{2.2}$$

$$\mathbf{x}_\omega = \{\omega_x, \omega_y, \omega_z\}, \quad \omega_x, \omega_y, \omega_z \in \mathbb{R}\left[\frac{\mathrm{rad}}{\mathrm{s}^2}\right] \tag{2.3}$$

$$\mathbf{x}_\varphi = \{\varphi_x, \varphi_y, \varphi_z\}, \quad \varphi_x, \varphi_y, \varphi_z \in \mathbb{R}\,[^\circ] \tag{2.4}$$

Finally, the model utilises the current depth value $\mathbf{x}_d$, measured in metres (Equation (2.5)).

$$\mathbf{x}_d = d, \quad d \in \mathbb{R}, d \geq 0\,[\mathrm{m}] \tag{2.5}$$

The control settings vector $\mathbf{y}$ estimated by the model consists of four real values in the range $[-1, 1]$ (according to Equation (2.6)). All these values are illustrated in Figure 2.2:

- **longitudinal velocity setting** v_z ($v_z = 1$ stands for maximal forward velocity),
- **lateral velocity setting** v_x ($v_x = 1$ stands for maximal velocity to the right),
- **vertical velocity setting** v_y ($v_y = 1$ stands for maximal emergence speed),
- **yaw velocity setting** ω_y ($\omega_y = 1$ stands for maximum clockwise angular velocity).

$$\mathbf{y} = \{v_z, v_x, v_y, \omega_y\}, \quad v_z, v_x, v_y, \omega_y \in \mathbb{R} \tag{2.6}$$

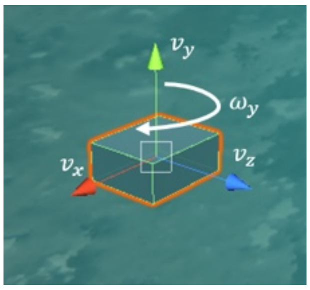


Figure 2.2: Control settings vector components: longitudinal velocity v_z, lateral velocity v_x, vertical velocity v_y and yaw velocity ω_y

2.2.1 Agent Controller

The high-level architecture of the agent controller is presented in Figure 2.3. It is implemented as a deep reinforcement learning model, which maps an input vector $\mathbf{x}$ that contains information from all sensors to an output control settings vector $\mathbf{y}$.

The image from the front camera is first processed by the Vision Module, which extracts visual features and reduces dimensionality. Its output is concatenated with readings from other sensors (i.e. linear acceleration, angular velocity, current rotation and depth) in the Data Processing Module. To capture temporal relationship between streams of data, a recurrent network is applied as the final module of the controller (the Time-Series Analysis Module). The output of this module is the control vector $\mathbf{y}$, determining the robot's movement at each step.

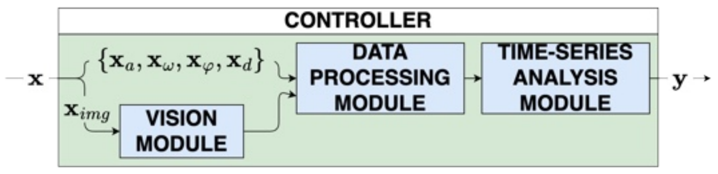

CONTROLLER
$\mathbf{x}$
$\{\mathbf{x}_a, \mathbf{x}_\omega, \mathbf{x}_\varphi, \mathbf{x}_d\}$
$\mathbf{x}_{img}$
VISION MODULE
DATA PROCESSING MODULE
TIME-SERIES ANALYSIS MODULE
$\mathbf{y}$

Figure 2.3: Overview of the controller architecture used by the agent. Sensor inputs $\mathbf{x}$ are processed through three main modules: the Vision Module (visual feature extraction), the Data Processing Module (sensor fusion and feature transformation) and the Time-Series Analysis Module (modelling temporal dependencies). The controller outputs the control vector $\mathbf{y}$.

The controller's model is trained using an actor-critic strategy [166], specifically the A2C (Advantage Actor-Critic) method [132]. This approach employs two models; the actor (policy gradient method) is used to control the agent behaviour, while the critic evaluates those actions using an action-value function. Both components are trained in parallel: the actor policy $\pi_\theta(\mathbf{s}, \mathbf{a})$ is optimised to maximise the value of reward for actions taken by the agent, while the critic function $Q^*_\omega(\boldsymbol{s}, \boldsymbol{a})$ is trained to minimise the error in reward function approximation. Here, $\mathbf{s}$ refers to the state vector and $\mathbf{a}$ is the action vector. With the A2C, training is distributed across multiple workers, which interact with their own environments, maintaining local versions of the global network. These local networks are updated independently, and their gradients are synchronised periodically to update the global model.

The controller uses PPO (Proximal Policy Optimisation) [154] as the policy gradient method. PPO extends the classic gradient ascent method by introducing a trust region mechanism, limiting the maximal update step size, and thereby preventing destabilising shifts in the learned policy. The size of the trust region is set using the clipped surrogate objective function, penalising excessive updates. This approach efficiently restricts policy weight changes, balancing exploration and stability of the training process.

Vision Module. As illustrated in Figure 2.4, the Vision Module supports two methods for image processing, which can be used independently or in combination. The first option is a custom convolutional neural network (CNN) based on the architecture from [133], trained end-to-end with the rest of the controller. The second option utilises the YOLO object detection architecture [146], specifically the lightweight TinyYOLO variant (see Figure 2.5). This network, modified to detect only a single object, processes images in a single forward pass and outputs either the coordinates of a detected object $\mathbf{z}_{\text{bbox}}$, or intermediate features extracted via early exits ($\mathbf{z}_{\text{basic}}$ and $\mathbf{z}_{\text{raw}}$). Optionally, the selected representation is passed through flattening and additional, trainable fully-connected layers. The Vision Module returns the visual feature vector $\mathbf{z}_{\text{img}}$.

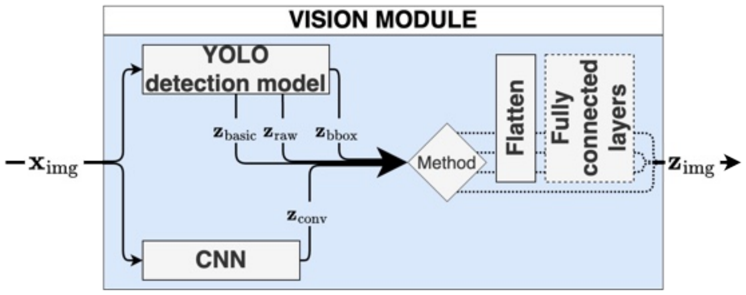

VISION MODULE
YOLO detection model
$\mathbf{z}_{\text{basic}}$
$\mathbf{z}_{\text{raw}}$
$\mathbf{z}_{\text{bbox}}$
$\mathbf{x}_{\text{img}}$
Method
Flatten
Fully connected layers
$\mathbf{z}_{\text{img}}$
$\mathbf{z}_{\text{conv}}$
CNN

Figure 2.4: Structure of the Vision Module used for visual feature extraction. The input image tensor $\mathbf{x}_{\text{img}}$ is processed either by a custom CNN or a pre-trained YOLO detection model. From the YOLO model, three levels of visual features can be selected: bounding box predictions ($\mathbf{z}_{\text{bbox}}$), and *basic* or *raw* intermediate features ($\mathbf{z}_{\text{basic}}$ and $\mathbf{z}_{\text{raw}}$). The selection of the processing path is denoted by the *Method* block. The chosen visual representation is optionally flattened and passed through a set of trainable, fully-connected layers. The final visual embedding $\mathbf{z}_{\text{img}}$ is forwarded to subsequent modules in the controller.

CNN. The custom CNN's architecture used in the controller follows [133] and consists of the following layers:

- **convolutional layer** (32 filters 8×8 with stride 4×4),
- **convolutional layer** (64 filters 4×4 with stride 2×2),
- **convolutional layer** (64 filters 3×3 with stride 1×1),
- **flattening layer**,
- **fully-connected layer** (512 neurons).

The output of the CNN is a tensor of shape $48 \times 48 \times 64$. It is then flattened and can be processed using fully connected layers. In a configuration which uses the CNN, the whole controller model is trained end-to-end.

YOLO detection model. The architecture of the YOLO models is shown in Figure 2.5. It consists of a series of convolutional layers, with the first six layers each followed by a max-pooling layer. Batch normalisation is applied after every convolutional layer, and ReLU is used as the activation function. The final output is a bounding box $\mathbf{z}_{\text{bbox}}$, which localises the detected object in the input image. To explore the usability of internal visual representations, two early-exit points were added to the network (marked as $\mathbf{z}_{\text{basic}}$ and $\mathbf{z}_{\text{raw}}$, corresponding to different levels of feature complexity). The motivation for using internal features instead of final bounding box output is to retain richer semantic information that may improve decision-making by the controller. In this framework, the YOLO model is pre-trained independently of the controller and used as a fixed, non-trainable feature extractor within the Vision Module.

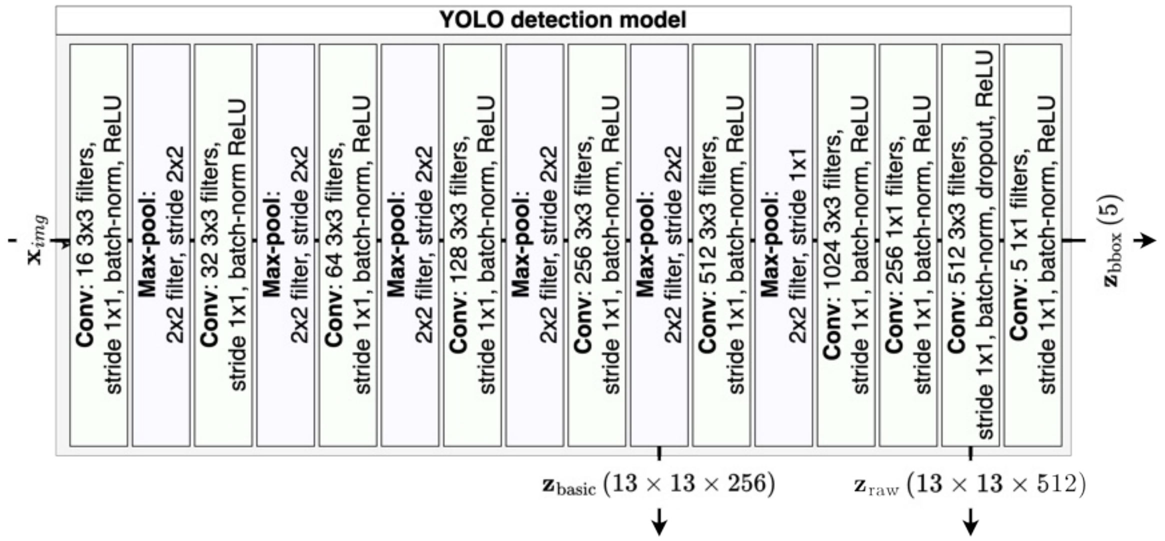

YOLO detection model
$\mathbf{x}_{img}$
Conv: 16 3x3 filters, stride 1x1, batch-norm, ReLU
Max-pool: 2x2 filter, stride 2x2
Conv: 32 3x3 filters, stride 1x1, batch-norm ReLU
Max-pool: 2x2 filter, stride 2x2
Conv: 64 3x3 filters, stride 1x1, batch-norm, ReLU
Max-pool: 2x2 filter, stride 2x2
Conv: 128 3x3 filters, stride 1x1, batch-norm, ReLU
Max-pool: 2x2 filter, stride 2x2
Conv: 256 3x3 filters, stride 1x1, batch-norm, ReLU
Max-pool: 2x2 filter, stride 2x2
Conv: 512 3x3 filters, stride 1x1, batch-norm, ReLU
Max-pool: 2x2 filter, stride 1x1
Conv: 1024 3x3 filters, stride 1x1, batch-norm, ReLU
Conv: 256 1x1 filters, stride 1x1, batch-norm, ReLU
Conv: 512 3x3 filters, stride 1x1, batch-norm, dropout, ReLU
Conv: 5 1x1 filters, stride 1x1, batch-norm, ReLU
$\mathbf{z}_{\text{bbox}}$ (5)
$\mathbf{z}_{\text{basic}}$ $(13 \times 13 \times 256)$
$\mathbf{z}_{\text{raw}}$ $(13 \times 13 \times 512)$

Figure 2.5: Architecture of the YOLO detection model used in the Vision Module. The input is the image tensor $\mathbf{x}_{\text{img}}$ (scaled to 416×416). The model consists of a series of convolutional layers (*Conv*) with batch normalisation (*batch-norm*), interleaved with max-pooling layers (*Max-pool*). Two early-exit points are defined for intermediate feature extraction: $\mathbf{z}_{\text{basic}}$ and $\mathbf{z}_{\text{raw}}$. The final output $\mathbf{z}_{\text{bbox}}$ is the detected object's bounding box prediction.

The YOLO network is an object detection model that estimates the position and size of each object on the image. It uses a 13×13 grid to divide the input image into 169 cells, where each predicts one bounding box. To match this grid structure, the input image is resized to 416×416 pixels. Each of the convolutional layers performs feature extraction, finally returning a full bounding boxes tensor $\mathbf{Z}_{\text{bbox}}$ of shape $13 \times 13 \times 5$. Each five-element vector $[c, x, y, w, h]$ represents a bounding box prediction for one cell, where c is the detection confidence (Equation (2.8)), x and y are the normalised coordinates of the bounding box centre and w and h are its normalised width and height. To produce the final output $\mathbf{z}_{\text{bbox}}$, only the bounding box with the highest confidence score is selected, according to Equation (2.7).

$$\mathbf{z}_{\text{bbox}} = \arg\max_{C}(\mathbf{Z}_{\text{bbox}}) = \left[\hat{c}, \hat{x}, \hat{y}, \hat{w}, \hat{h}\right] \tag{2.7}$$

$$c = p_{\text{obj}} \cdot \text{IoU}\left(\mathbf{z}^{*}_{\text{bbox}}, \mathbf{z}_{\text{bbox}}\right) \tag{2.8}$$

where:

p_{obj} is the probability, that the bounding box contains the object,

$\mathrm{IoU}\left(\mathbf{z}^*_{\mathrm{bbox}}, \mathbf{z}_{\mathrm{bbox}}\right)$ is the Intersection over Union (Equation (1.16)) between ground truth $\mathbf{z}^*_{\mathrm{bbox}}$ and predicted bounding box $\mathbf{z}_{\mathrm{bbox}}$.

The YOLO model utilised in this research was modified to detect a single type of objects, therefore classification error component was removed from the original loss function. The assumed loss function is described in Equation (2.9):

$$L = L_{\mathrm{pos}} + L_{\mathrm{size}} + L_{\mathrm{obj}} + L_{\mathrm{noobj}} \tag{2.9}$$

where:

L_{pos} is the **detected object position error** (based on the difference between the target and predicted bounding box position):

$$L_{\mathrm{pos}} = \lambda_{\mathrm{coord}} \sum_{i=0}^{S^2} \mathbb{1}_i^{\mathrm{obj}} \left[\left(x_i^* - x_i\right)^2 + \left(y_i^* - y_i\right)^2 \right] \tag{2.10}$$

λ_{coord} – position component relevance coefficient,
S – number of rows and columns in YOLO grid,
$\mathbb{1}_i^{\mathrm{obj}}$ – step function ($\mathbb{1}_i^{\mathrm{obj}} = 1$ if cell i contains the detected object; $\mathbb{1}_i^{\mathrm{obj}} = 0$ if it does not),
(x_i^*, y_i^*) – target bounding box position,
(x_i, y_i) – predicted bounding box coordinates,

L_{size} is the **bounding box size error** (comparing predicted and true width and height of the bounding box):

$$L_{\mathrm{size}} = \lambda_{\mathrm{coord}} \sum_{i=0}^{S^2} \mathbb{1}_i^{\mathrm{obj}} \left[\left(\sqrt{w_i^*} - \sqrt{w_i}\right)^2 + \left(\sqrt{h_i^*} - \sqrt{h_i}\right)^2 \right] \tag{2.11}$$

(w_i^*, h_i^*) – real width and height of the bounding box,
(w_i, h_i) – predicted size of the detection,

$L_{\mathrm{obj}}, L_{\mathrm{noobj}}$ are the **positive and negative detection error**, based on comparing the confidence of the detection C_i with the ground truth C_i^* ($C_i^* = 0$ for negative detection; $C_i^* = 1$ for positive):

$$L_{\mathrm{obj}} = \sum_{i=0}^{S^2} \mathbb{1}_i^{\mathrm{obj}} \left(C_i^* - C_i\right)^2 \tag{2.12}$$

$$L_{\mathrm{noobj}} = \lambda_{\mathrm{noobj}} \sum_{i=0}^{S^2} \mathbb{1}_i^{\mathrm{noobj}} \left(C_i^* - C_i\right)^2 \tag{2.13}$$

λ_{noobj} – negative detection loss relevance factor,

$\mathbb{1}_i^{\text{noobj}}$ – step function ($\mathbb{1}_i^{\text{noobj}} = 1$ if cell i does not contain the detected object; $\mathbb{1}_i^{\text{noobj}} = 0$ if it does).

Data Processing Module. The data collected from the robot's sensors (AHRS and depth sensor) are concatenated with the Vision Module's output, creating the input for the Data Processing Module. A sequence of trainable, fully-connected layers is used to process this input vector.

Time-Series Analysis Module. The navigation task exhibits strong temporal dependencies, requiring the controller to condition its action on the past environmental states as well as the current observation. To capture these temporal dynamics, a recurrent network (LSTM [83]) is used and trained jointly with the rest of the controller. This module processes the output of the Data Processing Module, propagating the temporal dynamics via a hidden state (of size 256). The final output of the module $\mathbf{y}$ is the control setting vector, specifying the signals used to steer the robot.

2.2.2 Reward Function

The reward function is a critical component of reinforcement learning, which guides the training process and shapes the agent's behaviour [166]. An effective reward function should accurately evaluate agent actions, promoting those leading to success and discouraging suboptimal or incorrect decisions. In the context of the navigation task researched here, the goal is for the agent to reach the target efficiently, following a smooth and direct path, while maintaining appropriate speed and natural orientation. To support this, four reward components were proposed to influence specific aspects of the agent's behaviour.

- **Position reward** R_p, based on the agent's distance from the target point (e.g. the centre of the gate), computed separately for each dimension $D \in \{X, Y, Z\}$ according to Equation (2.14) (see Figure 2.6a); this component encourages the agent to minimise the distance to the target, with the reward increasing as the agent gets closer.

$$R_{p,D} = \exp\left(|d_{\text{target}} - d_{\text{agent}}| \frac{\ln k_{\min,D}}{l_{\min,D}} \right) \tag{2.14}$$

where:

$|d_{\text{target}} - d_{\text{agent}}|$ is the absolute distance between agent and target position,

$k_{\min,D}$ is the expected function value achieved at distance $l_{\min,D}$; during research, $k_{\min,D}$ and $l_{\min,D}$ were set empirically.

- **Rotation reward** R_r, which evaluates the agent's yaw orientation relative to the normal vector of the target object's front plane (illustrated in Figure 2.6b and defined in Equation (2.15)). This component encourages the agent to align its heading with the target, so that it can approach the object by moving forward.

$$R_r = \cos\left(\gamma_{\text{target}} - \gamma_{\text{agent}}\right) \tag{2.15}$$

where:

γ is the normal angular position around yaw axis.

- **Velocity reward** R_v, evaluating agent's movement direction and speed, defined in Equation (2.16). This component encourages the agent to move toward the gate as quickly and directly as possible.

$$R_v = \cos\left[\sphericalangle\left(\boldsymbol{v}_{\text{agent}}, \boldsymbol{T}_{\text{target}}\right)\right] \frac{|\boldsymbol{v}_{\text{agent}}|}{v_{\max}} \tag{2.16}$$

where:

$\sphericalangle\left(\boldsymbol{v}_{\text{agent}}, \boldsymbol{T}_{\text{target}}\right)$ is the angle between agent velocity vector and a vector connecting agent's centre of mass and target point (see Figure 2.6c),
$|\boldsymbol{v}_{\text{agent}}|$ is the length of the velocity vector, normalised to maximal agent's speed $v_{\max}$;

- **Angular velocity reward** R_{av}, which evaluates whether the agent's current rotational motion is aligned with the desired yaw orientation, as defined in Equation (2.17). It encourages angular movement towards the target's orientation (i.e. parallel to target object's normal) and penalises turning away from it.

$$R_{av} = \cos\left[\pi\left(\frac{\tanh\left(\Delta\gamma\right) - \tanh\omega_Y}{\tanh 1}\right)\right] \tag{2.17}$$

where:

$\Delta\gamma = \gamma_{\text{target}} - \gamma_{\text{agent}}$ is the difference between agent and target normals' angular positions around the yaw axis (same as in Equation (2.15)), normalised to the range $[-1, 1]$,
ω_Y is the normalised yaw angular velocity.

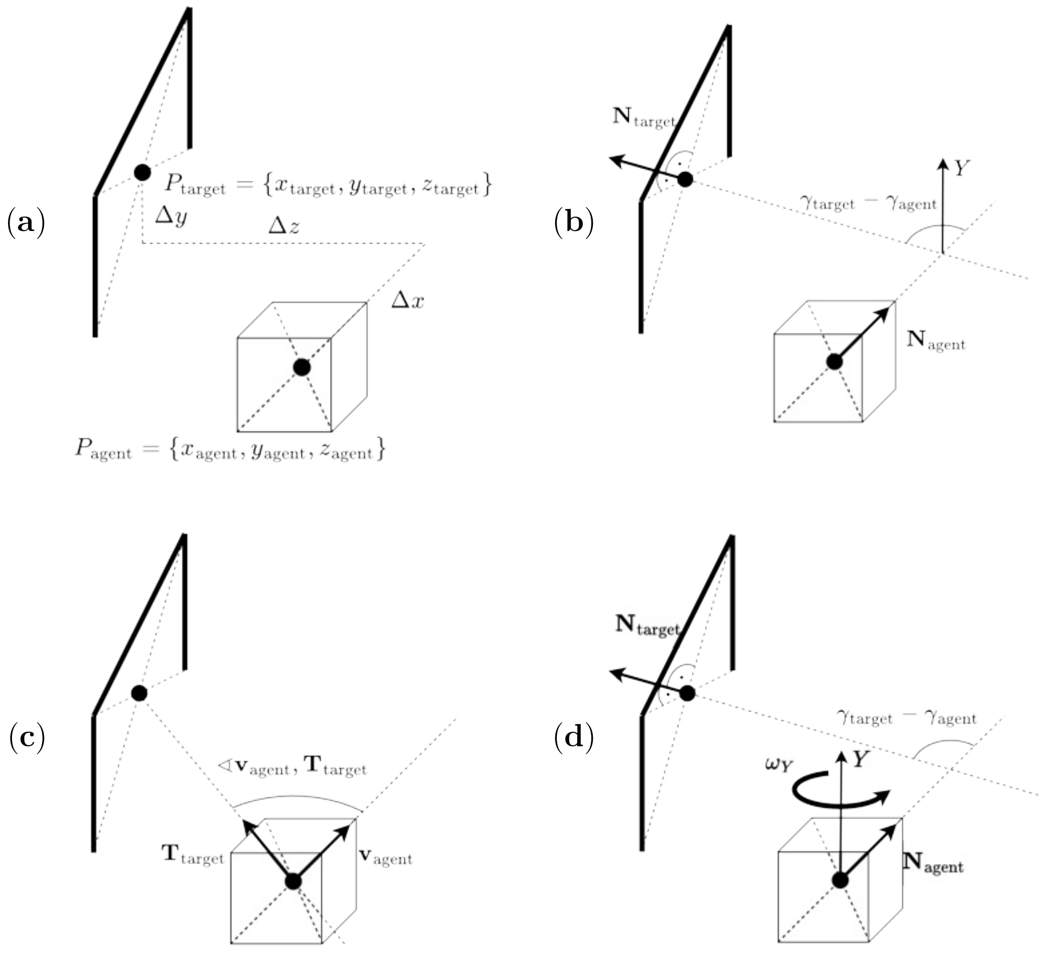

(a)
$P_{\text{target}} = \{x_{\text{target}}, y_{\text{target}}, z_{\text{target}}\}$
Δy
Δz
Δx
$P_{\text{agent}} = \{x_{\text{agent}}, y_{\text{agent}}, z_{\text{agent}}\}$
(b)
$\mathbf{N}_{\text{target}}$
Y
$\gamma_{\text{target}} - \gamma_{\text{agent}}$
$\mathbf{N}_{\text{agent}}$
(c)
$\sphericalangle \mathbf{v}_{\text{agent}}, \mathbf{T}_{\text{target}}$
$\mathbf{T}_{\text{target}}$
$\mathbf{v}_{\text{agent}}$
(d)
$\mathbf{N}_{\text{target}}$
$\gamma_{\text{target}} - \gamma_{\text{agent}}$
ω_Y
Y
$\mathbf{N}_{\text{agent}}$

Figure 2.6: Parameters used to compute each reward component. (**a**) Position reward: based on the agent's distance from the target along each axis (Δx, Δy and Δz). (**b**) Rotation reward: computed from the difference in yaw orientation between the agent's and the target's surface normals $\boldsymbol{N}_{\text{agent}}$ and $\boldsymbol{N}_{\text{target}}$ (γ_{agent} and γ_{target}). (**c**) Velocity reward: assesses the alignment and magnitude of the agent's linear velocity vector $\boldsymbol{v}_{\text{agent}}$ with respect to the target direction vector $\boldsymbol{T}_{\text{target}}$. (**d**) Angular velocity reward: evaluates the agent's yaw angular velocity ω_Y in relation to the desired orientation difference ($\gamma_{\text{target}} - \gamma_{\text{agent}}$).

To discourage undesirable behaviour in which the agent prolongs episodes by accumulating small rewards over time, two constraints were introduced. First, the duration of each training episode is limited by a maximum number of steps L_{episode}. Second, an exponential discount factor D_{exp} (defined in Equation (2.18)) is applied to positive rewards; this factor decreases continuously from the start of the episode, reaching a value of 0.1 at the halfway point, thereby encouraging the agent to complete the task efficiently.

$$D_{\text{exp}} = \exp\left(\text{step} \cdot \frac{\ln 0.1}{\frac{1}{2}L_{\text{episode}}}\right) \tag{2.18}$$

The agent receives several distinct rewards and penalties. Upon successfully reaching the target, it is granted a high reward, as defined in Equation (2.19). Task completion is determined by checking whether the agent enters a predefined box surrounding the target object with an added tolerance margin. Crossing into this zone is treated as a successful episode termination, triggering an environment reset and the start of a new training episode.

$$R_{success} = 50 \cdot D_{\text{lin}} \left(R_r w_r + R_{p,\text{avg}} w_p \right) \tag{2.19}$$

$$R_{p,\text{avg}} = \frac{1}{3} \left(R_{p,x} + R_{p,y} + R_{p,z} \right) \tag{2.20}$$

$$D_{\text{lin}} = \frac{k_{\text{min,lin}} - 1}{l_{\text{min,lin}}} \cdot \text{step} + 1 \tag{2.21}$$

where:

w_r, w_p are the weights for rotation and position components (parameters),
$R_{p,\text{avg}}$ is the average position reward for each axis,
D_{lin} is the linear discount factor, taking the value $k_{\text{min,lin}}$ at step $= l_{\text{min,lin}}$ (promotes reaching the target in a low number of steps).

The agent is punished when hitting an obstacle ($R_{\text{penalty}} = -1$) or when making a fatal mistake, such as untimely emergence or missing the target ($R_{\text{fatal}} = -10$). A fatal mistake causes an immediate reset of the environment and starts a new training episode.

2.3 Experiments

The controller model was evaluated to assess its effectiveness in solving the navigation task and to analyse how various hyperparameters affect training dynamics, preformance efficiency and achieved results. This section describes the experimental platform, the evaluation procedure and presents the experiments conducted to validate the proposed approach.

2.3.1 Simulation Environment

To evaluate the proposed solution, a simulation environment was developed to closely replicate real-world conditions at the RoboSub test facility. The facility mirrors the structure of the competition arena, which is divided into four quarters, each containing an identical set of objects. The quarter in which a team performs its run is selected at random, and the robot's initial orientation is randomly assigned via a coin flip (see

Figure 2.7). To ensure robustness to varying conditions, the environment includes randomisation of various parameters, such as water hue and opacity, sun position and lighting conditions. The simulation is implemented in Unity Software, based on the model provided by the Coleman University team. Integration with the environment is provided by the Unity ML-Agents toolkit [94], supporting interaction with externally controlled models.

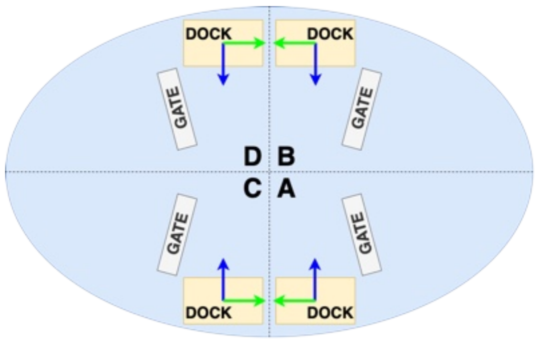


Figure 2.7: Test facility layout. Each region is marked with letters A-D: the robot's starting orientations are indicated with green and blue arrows.

2.3.2 Object Detection Model

The detection model was trained to detect the gate, used as the target object in the first competition task. The model was evaluated using Intersection over Union (Equation (1.16)) and the modified loss function [146] (Equation (2.9)).

Dataset. To train the object detection model, a dataset of images was generated using the simulation environment. Each image depicts the target gate from the perspective of the robot's front camera. For every sample, a random facility quarter was selected, along with randomised positions for other scene elements. The agent was positioned and oriented such that the gate appeared in the camera's field of view, with slight perturbations applied to simulate natural variation. During image generation, the conditions within the environment (such as water hue and opacity, sun position and brightness, etc.) were randomised dynamically. Additionally, various distractor objects from other competition tasks were placed around the gate to introduce visual noise.

The bounding box annotations for each image were generated automatically by computing the gate's position relative to the agent's camera. Each annotation consists of a four-element vector (x, y, w, h), representing the centre coordinates, width and height of the bounding box, normalised to the image dimensions.

Using this procedure, a dataset of 4000 training examples was collected, evenly split between positive and negative samples (determined by the visibility of the gate). Half of the images included added distractor objects. A separate test set of 400 images was generated using the same procedure.

Training and evaluation. The object detection model was trained using online image augmentation to improve generalisation. Each training batch was modified by randomised transformations (horizontal flipping, random crop, Gaussian blur and noise, contrast normalisation and random affine transformation). To form mini-batches, a randomly selected buffer of 1000 images was drawn from the full dataset each time the model was saved. The key training hyperparameters are listed in Table 2.1.

Table 2.1: Object detection model hyperparameters

hyperparameter	**value**
mini-batch size	30
learning rate	1×10^{-4}
model saving frequency	10000 steps
position loss coefficient (λ_{coord})	5
no-object loss coefficient (λ_{noobj})	0.5
YOLO grid size (S_{grid})	13

The training strategy followed an iterative approach involving alternating cycles of learning steps, model evaluation, weight checkpointing and re-sampling of the training buffer. The learning curve, presented in Figure 2.8, shows that the best performance (measured by the Intersection over Union (IoU) on both training and validation datasets) was achieved between the sixth and seventh saved checkpoints. An empirical comparison of saved models' performance confirmed that the 70k-step model offered the highest performance in a test video generated with the simulation. As a result, this checkpoint was selected for integration into the Vision Module in subsequent experiments.

2.3.3 Navigation Controller

The controller was implemented using a deep reinforcement learning model from the Stable Baselines library [80], a widely adopted fork of OpenAI's Baselines [37]. Both training and evaluation were conducted within the simulation environment prepared for the project. The performance was measured using four metrics:

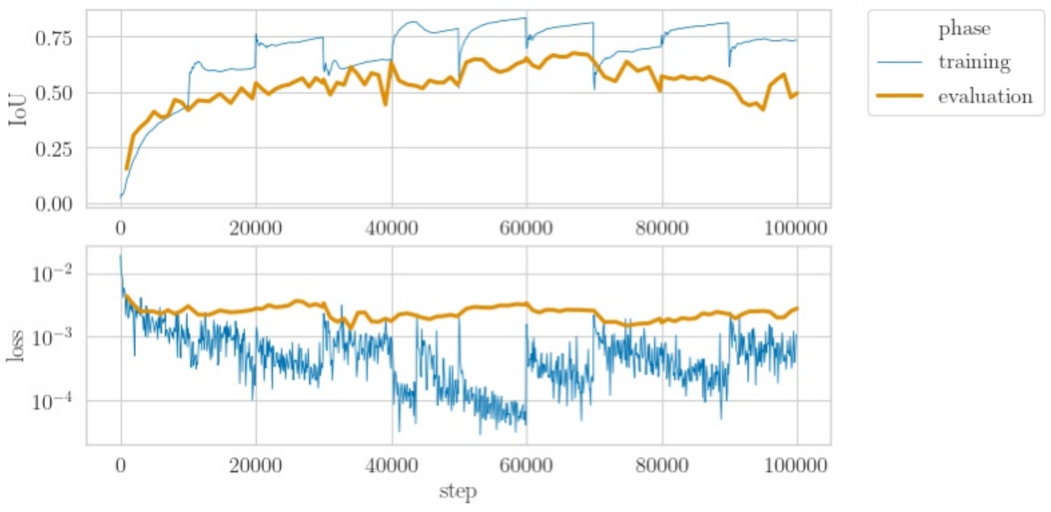


Figure 2.8: Learning curves for the object detection model during training. The top plot shows the Intersection over Union (IoU) metric, while the bottom shows corresponding loss values, for both the training and evaluation phases.

- **Mean reward** $\bar{R}$, the average cumulative reward collected by the agent across all environment instances:

$$\bar{R} = \frac{1}{N_e} \sum_{i}^{N_e} \sum_{t=0}^{L_{\text{episode}}} r_i(t) \tag{2.22}$$

where:

N_e – total number of agents,
L_{episode} – episode length,
$r_i(t)$ – value of a reward received by the agent in i-th environment at time step t.

- **Average episode length** $\bar{L}$, i.e. the number of steps taken by the agent before reaching the target or terminating the episode. High reward with short episodes indicates effective behaviour, while low reward with short episodes may reflect failure cases (e.g. untimely emergence or missing the target). Conversely, long episodes may suggest indecisive or wandering behaviour.

- **Inference speed** $\bar{\text{it}}_s$, calculated as the average number of agent steps per second during evaluation:

$$\bar{\text{it}}_s = \frac{N_{\text{steps}}}{T_{\text{eval}}} \tag{2.23}$$

where:

N_{steps} – total number of steps taken,
T_{eval} – total evaluation time in seconds.

- **Episode statistics**, i.e. proportions of different episode outcomes:

$$q_S = \frac{|S|}{|T|}; \quad q_E = \frac{|E|}{|T|}; \quad q_M = \frac{|M|}{|T|}; \quad q_C = \frac{|C|}{|T|} \tag{2.24}$$

  where:

  $|S|$ – number of successful episodes (agent reaches the target),
  $|E|$ – number of episodes terminated by untimely emergence,
  $|M|$ – number of episodes where the target was missed,
  $|C|$ – number of episodes in which a collision occurred,
  $|T|$ – total number of evaluated episodes; $|T| = |S| + |E| + |M|$.

During training, 12 agents were run in parallel using the actor-critic architecture with the A2C method, over a fixed period of 24 hours (using the test platform specified in Table 2.2). The training procedure assumed cyclic model evaluation every 50000 training steps: each evaluation episode involved performing 200 steps according to the model's prediction in each of the 12 environments and registering average reward and episode length. By repeating this process five times, evaluation metrics were averaged, while episode statistics summed accordingly, allowing for identifying the best-performing model for each run (i.e. the one achieving the highest average reward per test episode during training).

Table 2.2: Test platform configuration

CPU	AMD Ryzen 7 2700
RAM	32 GB
GPU	Nvidia RTX 2070
OS	Ubuntu Linux 18.04

The experiments focus on analysing how the choice of visual feature embeddings and the form of the reward function affect the controller's performance, training efficiency and inference speed. In all experiments, models were trained using the default parameters from Table 2.3. The specific hyperparameters explored in the study, along with their possible configurations, are summarised in Table 2.4.

2.3.4 Preliminary Architecture Selection

The configuration of the Data Processing Module (Data Processing FC layers) was selected through a small-scale hyperparameter sweep. The search involved several configurations of the Data Processing Module, varying the size of fully-connected (FC)

Table 2.3: Default parameters used in experiments

hyperparameter	value	description
T_{training}	24	model training period (hours)
n_{envs}	12	number of agents
n_{train}	50000	number of steps between each evaluation and model saving
n_{eval}	200	number of evaluation steps
i_{dec}	10	interval between each model's decision
L_{episode}	4000	maximum number of steps per episode
C_t	256	LSTM cell state vector length
$L_{\text{minibatch}}$	30	number of steps in one mini-batch
$n_{\text{minibatch}}$	4	number of mini-batches used for model training
lr	0.00025	constant learning rate value
ε_{PPO}	0.2	PPO clipping threshold
$k_{\text{min},X}$	0.3	
$k_{\text{min},Y}$	0.3	
$k_{\text{min},Z}$	0.3	expected position reward value $k_{\text{min},D}$ achieved at
$l_{\text{min},X}$	0.8	distance $l_{\text{min},D}$ for each dimension (Eq. (2.14))
$l_{\text{min},Y}$	0.5	
$l_{\text{min},Z}$	2.0	
w_r	0.2	rotation component weight in success reward (Eq. (2.19))
w_p	0.1	position component weight in success reward (Eq. (2.19))
$k_{\text{min,lin}}$	0.5	linear discount factor parameters (Eq. (2.21))
$l_{\text{min,lin}}$	L_{episode}	

Table 2.4: Hyperparameters examined in the research

hyperparameter	values	description
visual features	$\{\mathbf{z}_{\text{bbox}}, \mathbf{z}_{\text{raw}}, \mathbf{z}_{\text{basic}}, \mathbf{z}_{\text{conv}}\}$	Visual embedding method used in the Vision Module (Fig. 2.4)
Vision FC layers	$[l_1, l_2, l_3, ...]$	The number of neurons in fully-connected (FC) layers in the Vision Module (None refers to no FC layers)
Data Processing FC layers	$[l_1, l_2, l_3, ...]$	The number of neurons in FC layers in the Data Processing Module (Fig. 2.4)

layers, as well as their number. In this experiment, all models were trained using the bounding box $\mathbf{z}_{\text{bbox}}$ as the visual representation and the complex reward function (Equation (2.29)).

The results, summarised in Table 2.5, show that the Data Processing Module configuration does not cause major performance changes: all tested configurations demonstrated the ability to learn the task, with moderate to high rewards and success rates. Among the two-layer variants, the 2×64 model performed best, outperforming both narrower and wider alternatives. Increasing the network depth yielded only minor performance gains; the 4×64 model achieved the highest success rate (51.8%) and competitive reward, without affecting the inference time or stability.

Table 2.5: Evaluation results for different FC layers configurations in the Data Processing Module. Top: varying the number of neurons per layer (fixed depth = 2). Bottom: varying the number of layers (fixed width = 64).

configuration	$\bar{R}$	$\bar{L}$	$\bar{it}_s$	q_S [%]	q_E [%]	q_M [%]	q_C [%]
Varying layer size (2 layers)							
2×16	10.33	70.5	2.05	41.5	9.4	49.1	19.5
2×64	**11.80**	65.5	1.99	**42.4**	4.0	53.6	18.3
2×256	8.38	61.2	2.02	19.0	3.7	77.3	15.3
Varying number of layers (width = 64)							
2×64	11.80	65.5	1.99	42.4	4.0	53.6	18.3
3×64	11.58	114.8	2.00	39.3	1.6	59.0	37.7
4×64	**11.81**	69.4	1.99	**51.8**	1.8	46.3	9.9

These results indicate that the performance of the controller is not highly sensitive to the precise configuration of the Data Processing Module, provided the model has sufficient, but not excessive, capacity. Consequently, further experiments adopt the 4×64 configuration.

2.3.5 Experiment 1: Influence of Reward Function Components on the Model Performance

The main aim of this experiment is to examine the importance of the reward function components. To do that, four reward functions definitions were researched:

- **analyzing only agent's position** R_{pos}:

$$R_{\text{pos}} = \frac{1}{4} D_{\text{exp}} \left(R_{p,X} + R_{p,Y} + R_{p,Z} + R_r \right) \tag{2.25}$$

- **assessing only agent's velocity** R_{vel}:,

$$R_{\text{vel}} = \frac{1}{2} D_{\text{exp}} \left(R_v + R_{av} \right) \tag{2.26}$$

- **complex form without angular velocity** R_{noangvel}:

$$R_{\text{noangvel}} = \frac{D_{\text{exp}} \left(R_v + b \right)}{1 + b} * \frac{w_v + w_p R_{p,\text{agg}} + w_r R_r}{w_v + w_p + w_r} \tag{2.27}$$

$$R_{p,\text{agg}} = R_{p,Z} \left(R_{p,X} + R_{p,Y} \right) \tag{2.28}$$

- **complex form, using all components** R_{complex}:

$$R_{\text{complex}} = \frac{D_{\text{exp}}\left(R_v + b\right)}{1+b} \frac{w_v + w_{av}R_{av} + w_p R_{p,\text{agg}} + w_r R_r}{w_v + w_{av} + w_p + w_r} \tag{2.29}$$

The complex form of the reward function is a weighted sum of elementary reward functions. The weights were selected empirically, based on preliminary behavioural analysis (details not included here). The specific values used in this experiment are listed in Table 2.6.

Table 2.6: Parameters of the complex reward function

symbol	**value**	**description**
w_v	0.8	velocity reward weight
w_{av}	0.2	angular velocity reward weight
w_p	0.1	position reward weight
w_r	0.2	rotation reward weight
b	0.2	velocity reward bias

The complex reward function defined in Equation (2.29) was designed based on several practical observations to provide an informative training signal. Conditioning the velocity component R_v should promote consistent motion toward the target; a bias term b ensures the reward remains informative even when the agent is stationary (which occurs particularly during early training stages, where lack of movements could otherwise lead to random exploration and unproductive reward accumulation). The longitudinal position component $R_{p,Z}$ is the most critical for aligning with the gate, which is why it is used as a conditioning factor in the aggregated position reward (Equation (2.28)). Finally, the entire expression is scaled to the range $[-1, 1]$ to improve training stability.

The experiment was conducted in two phases. First, models were trained using each of the reward function variants, while keeping the model architecture fixed (same as concluded in the preliminary hyperparameter search). In the second phase, the best-performing model from each training run was evaluated using an averaged reward function defined in Equation (2.30), to enable consistent comparison across configurations.

$$R_{\text{eval}} = \frac{1}{6}\left(R_{p,X} + R_{p,Y} + R_{p,Z} + R_r + R_v + R_{av}\right) \tag{2.30}$$

The results of the training are shown in Figure 2.9; *position-only* refers to reward function from Equation (2.25), *velocity-only* is the form defined in Equation (2.26), *noangvel* refers to complex form without angular velocity component from Equation (2.27) and *complex* is the full form from Equation (2.29). Here, the rewards were calculated according to the form used for training the model (e.g. position-only form for the *position-only* model).

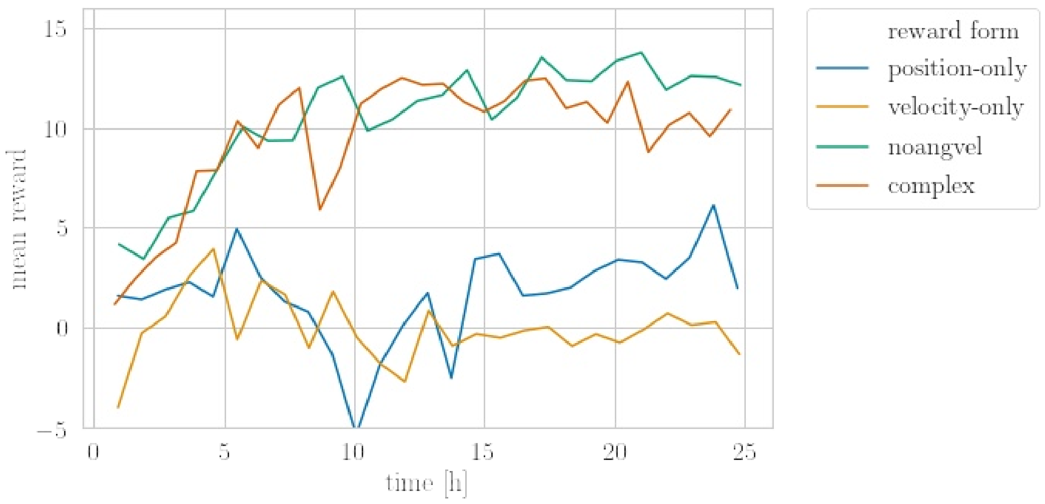


Figure 2.9: Learning curves comparing reward function forms. The plot shows the average reward obtained during periodic testing for each of the evaluated reward function formulations.

Following training, the model with the highest value of average reward achieved in evaluation sequences was subjected to a detailed evaluation procedure. The results are shown in Figure 2.10: (a) shows average evaluation reward (Equation (2.30)) achieved by the model, (b) presents average episode length during evaluation and (c) presents episode statistics. Since all models use the same model architecture, the model's speed was not analysed.

The performance differences between models trained using each reward function variant are clearly reflected in the evaluation results. Models trained with the simple formulations: *position-only* and *velocity-only* (Equations (2.25) and (2.26)) failed to learn effective navigation behaviour, achieving low mean rewards during training and evaluation. These models frequently terminated episodes through untimely emergence or by missing the target, indicating an inability to consistently complete the task.

In contrast, the two complex reward functions (Equations (2.27) and (2.29)) resulted in substantially better performance. The model trained with the full *complex* formulation achieved the highest evaluation reward and success rate. The *noangvel* variant also showed a higher proportion of successful runs than the simple reward functions, but the agent exhibited problems with aligning to the target normal, leading to more frequent misses. This behaviour can be attributed to the lack of the angular velocity component, which is essential for achieving the best performance in the full reward formulation.

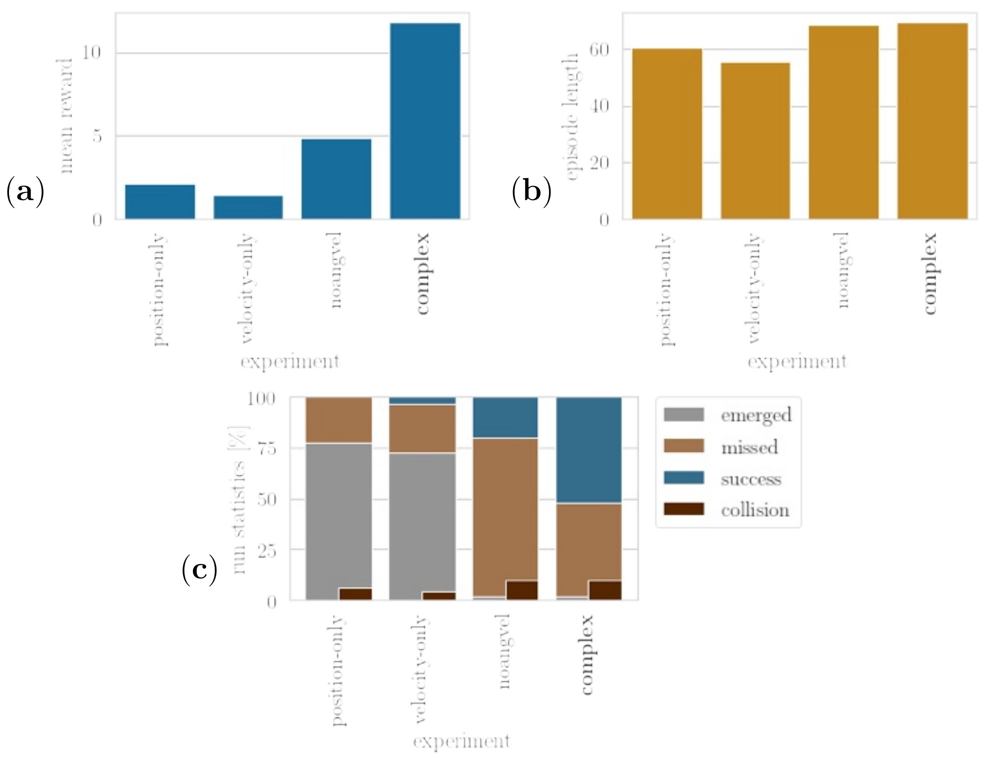


Figure 2.10: Evaluation results for each reward function form. (**a**) Mean reward achieved during evaluation, computed using the averaged reward function (Eq. (2.30)). (**b**) Average episode length. (**c**) Episode outcome statistics, showing the percentage of runs that ended in success, missing the target, untimely emergence, or collision.

One drawback observed in the best-performing model was a non-negligible collision rate ($\sim 10\%$), likely due to misaligned final approaches resulting in collisions with the gate structure.

Episode length results are consistent with these conclusions. The best models achieved high rewards within relatively short episodes, indicating efficient behaviour. In contrast, models trained with simple reward forms completed shorter episodes with poor performance, reflecting incorrect navigation.

2.3.6 Experiment 2: Influence of Visual Representations on the Model Performance

This experiment evaluates how different visual features extraction methods used in the Vision Module affect both the training process and the resulting performance of the controller. The objective was to analyse whether features of various complexity can effectively support the controller in solving the navigation task. All model configurations were trained using the complex reward function (Equation (2.29)), with default model parameters from Table 2.3 and researched variants listed in Table 2.7.

Each model configuration is identified by a structured denomination. The first term refers to the source of visual representations:

- ***default***: uses only the bounding box prediction from the YOLO network ($\mathbf{z}_{\text{bbox}}$),
- ***conv***: uses visual features extracted from a custom CNN trained end-to-end with the controller ($\mathbf{z}_{\text{conv}}$),
- ***raw*** and ***basic***: use feature maps extracted from early exits in a pre-trained YOLO network, with *raw* referring to penultimate convolutional layer features ($\mathbf{z}_{\text{raw}}$) and *basic* to mid-level features ($\mathbf{z}_{\text{basic}}$).

For each base embedding type, a model without additional processing in the Vision Module is denoted by the base name alone. These variants use deeper Data Processing Modules to handle the higher dimensionality of uncompressed visual inputs. Two additional suffixes are used to indicate architectural extensions:

- ***-fc***: indicates that the visual representation is processed by additional Vision FC layers,
- ***-bb***: indicates that the YOLO bounding box vector is concatenated with the visual representation before fusion.

These naming conventions are summarised in Table 2.7.

Figure 2.11 shows mean-reward training curves for *conv*, *raw* and *basic* representations-based models. Results of the evaluation of the best-performing checkpoints appear in Figure 2.12.

Among models without Vision Module fully connected layers, only the configuration using *basic* early-exit features from the YOLO network achieved meaningful performance, surpassing 40% success rate. In contrast, models using the *raw* early-exit features or trainable CNN embeddings failed to learn effective navigation policies, consistently oscillating around 0.0 reward throughout the entire process of training. The poor performance of *raw* configuration suggests that features extracted from the penultimate layer of the detection model are too specialised for object localisation, making them unsuitable for downstream navigation. On the other hand, the trainable CNN increases the controller's complexity, which makes its convergence more difficult. Adding the bounding box vector directly to high-dimensional visual embeddings (i.e., *bb* variants without Vision FC layers) did not improve performance. This is likely caused by the scale imbalance, where the small bounding box vector is concatenated with large feature tensors, leading to the controller ignoring the box location.

Table 2.7: Model naming conventions used in visual representation experiments. Each configuration name reflects the type of visual representation, and the architecture of the Vision and Data Processing Modules fully-connected layers.

name	**visual representation**	**Vision FC layers**	**Data Processing FC layers**
default	*bbox*	None	$[64, 64, 64, 64]$
conv	*conv*	None	$[256, 64, 64, 64]$
conv-fc	*conv*	[256]	$[64, 64, 64, 64]$
conv-bb	*conv + bbox*	None	$[256, 64, 64, 64]$
conv-bb-fc	*conv + bbox*	[256]	$[64, 64, 64, 64]$
raw	*raw*	None	$[1024, 256, 64, 64]$
raw-fc	*raw*	[1024, 256]	$[64, 64, 64, 64]$
raw-bb	*raw + bbox*	None	$[1024, 256, 64, 64]$
raw-bb-fc	*raw + bbox*	[1024, 256]	$[64, 64, 64, 64]$
basic	*basic*	None	$[1024, 256, 64, 64]$
basic-fc	*basic*	[1024, 256]	$[64, 64, 64, 64]$
basic-bb	*basic + bbox*	None	$[1024, 256, 64, 64]$
basic-bb-fc	*basic + bbox*	[1024, 256]	$[64, 64, 64, 64]$

In configurations where Vision FC layers were introduced (*fc* and *bb-fc*), performance generally improved for both CNN- and YOLO-based controllers. Notably, the *conv-bb-fc* model reached 25% success rate, with an average reward close to 10.00. However, the inclusion of Vision FC layers slightly degraded the performance of the *basic* features-based models.

What draws attention is the difference in prediction speed between models. YOLO-based feature extraction methods (*raw* and *basic*) resulted in more than 35% lower inference speed compared to end-to-end CNN models and bounding box prediction, primarily due to the overhead of extracting and transferring large feature maps from the detection network at every timestep. The use of bounding box prediction in feature maps-based models resulted in a modest computational overhead ($5-10\%$ slowdown), whereas adding Vision Module FC layers contributed minimally to inference time, with slowdowns below 5%.

2.3.7 Controller Performance Discussion

To illustrate the behaviour of the best-performing controller model, three representative examples of navigation attempts are presented. Figures below include plots of the robot's position during each run together with linear velocity settings output by the controller (Equation (2.6)), averaged over intervals between plotted points. Among

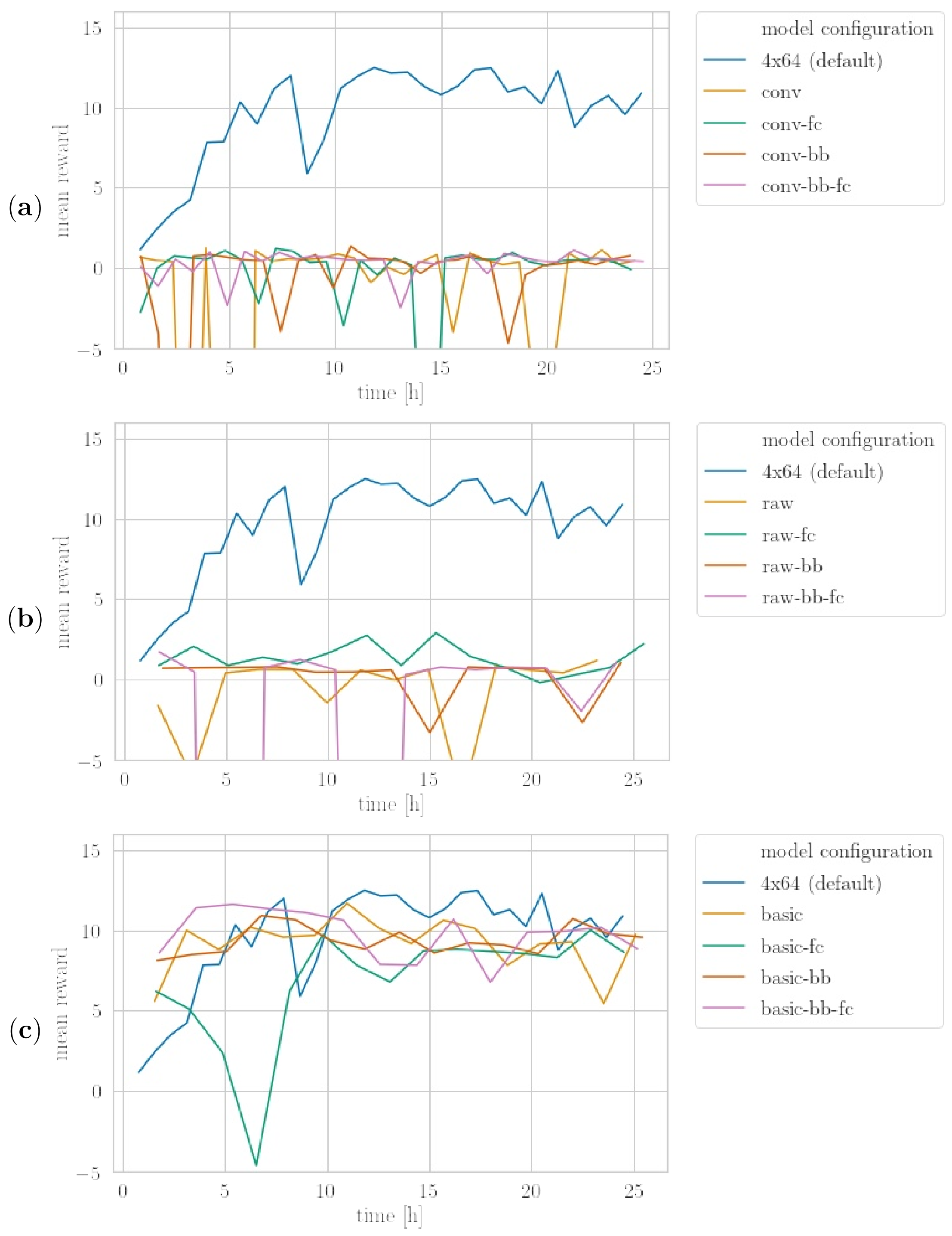


Figure 2.11: Learning curves comparing visual feature embedding methods. Each plot shows the average reward achieved during periodic testing over the 24-hour training period. The reference model (4×64) uses only the YOLO bounding box. (**a**) Models using visual features from a trainable CNN. (**b**) Models using *raw* early-exit features from the YOLO network. (**c**) Models using *basic* mid-level YOLO features.

all the runs, three categories of acquired paths were distinguished: successful runs with smooth path (Figure 2.13a), successful runs with a distorted trajectory (Figure 2.13b) and failed attempts (Figure 2.13c).

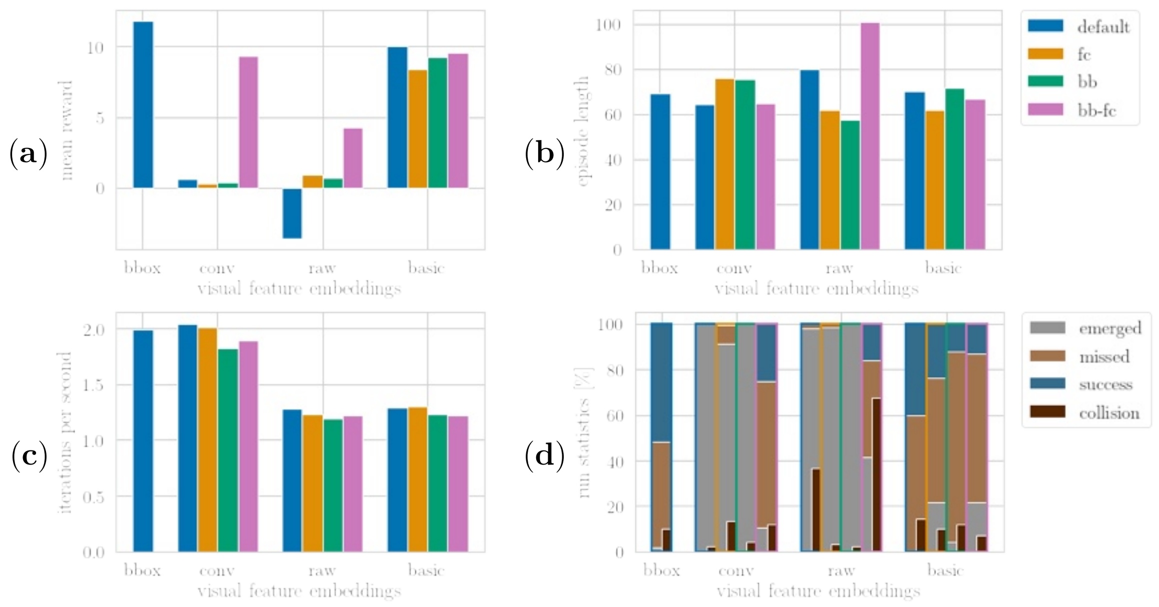


Figure 2.12: Evaluation results for various visual representations used in the model. Each group of bars corresponds to a different visual input type: *bbox* (YOLO bounding box), *conv* (features from a trainable CNN), *raw* and *basic* (early-exit feature maps from the YOLO network). Visual embeddings were tested in four configurations: *default* (no Vision FC layers), *fc* (with Vision FC layers), *bb* (concatenated with bounding box vector), and *bb-fc* (combined embedding with Vision FC layers). (**a**) Mean reward. (**b**) Average episode length. (**c**) Inference speed measured in iterations per second. (**d**) Episode statistics: percentage of runs ending in success, emergence, missed target, or collision.

Subjectively satisfactory paths, which constitute approximately half of all successful attempts, were typically observed when the robot was able to detect the target object early in the episode. This allowed it to maintain visual contact with the target throughout the run, resulting in a smooth and direct trajectory and stable velocity settings.

In the second group, the robot was often unable to detect the target object at the beginning of the episode, leading to chaotic or exploratory movements in the initial phase, likely reflecting searching for the target. Once the target was located, the controller typically succeeded in directing the robot toward it; however, the resulting trajectory was less stable compared to the satisfactory runs.

Unsuccessful runs were attributed to two failure modes. In the first one, the controller continued to search for the target object without success, ultimately exceeding the fixed episode length without reaching the goal. In the second case, the robot did detect the target but lacked information about its orientation (since the detection model

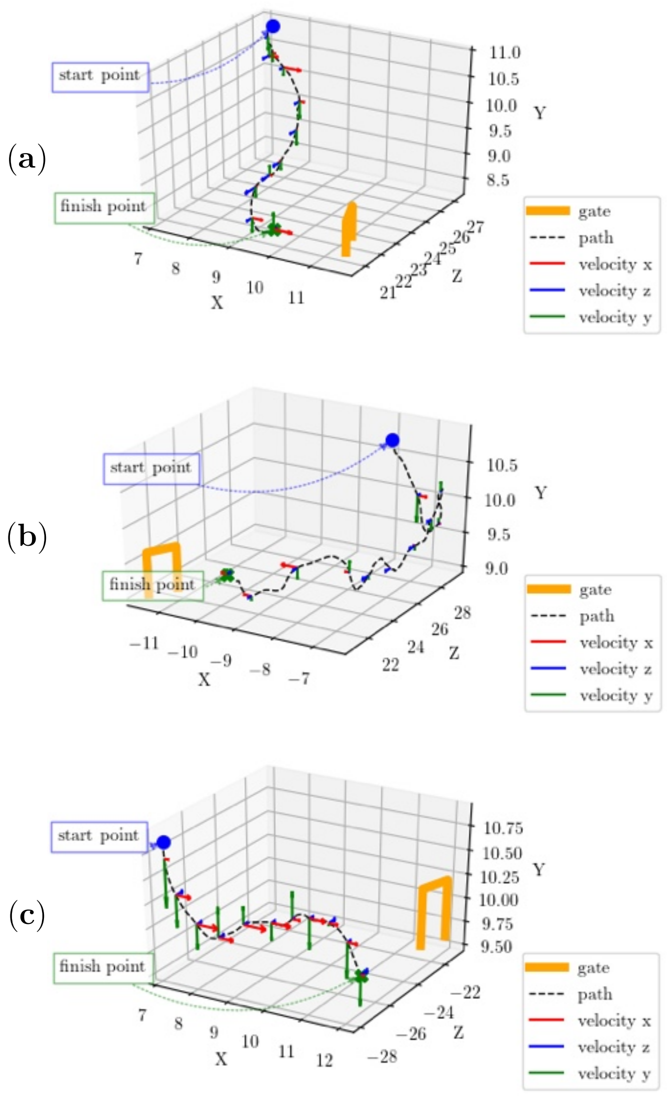


Figure 2.13: Visualisation of robot trajectories and control velocities using a trained *default* model (bounding box-based input). The plots show 3D paths along with velocity control settings in the X, Y, and Z directions. (**a**) A successful run with a smooth trajectory. (**b**) A successful run with a distorted path. (**c**) A failed run. All position axes are given in metres.

only provides a bounding box of the detection). As a result, the controller sometimes approached the target from an incorrect angle (e.g. from the side) and failed to align properly, causing the robot to pass by the gate.

2.4 Summary

This chapter presents research on a deep reinforcement learning-based controller for vision-guided 3D navigation of an autonomous underwater vehicle. While the primary goal was to demonstrate the feasibility of using DRL for underwater robotic control, the key focus of the study was the analysis of how different types of visual input, particularly those derived from a pre-trained object detection model, influence the performance of the navigation policy.

Four types of visual embeddings were investigated, ranging from a trainable convolutional feature extractor, through explicit detection outputs (bounding boxes) to internal feature maps extracted at varying depths of the detection network. This setup allowed examining how the level of abstraction in visual features affects learning dynamics, decision-making, and general navigation behaviour. The results showed that supplying the DRL agent with structured, semantically meaningful visual cues can lead to successful policy learning without the need for end-to-end training of a visual encoder.

Among the tested configurations, the '*basic*' feature maps, extracted from intermediate layers of the object detection network, demonstrated strong performance when incorporated into the controller, surpassing models that relied on heavier-processed '*raw*' features, or a simple convolutional feature extractor. These findings highlight the potential of using task-agnostic, abstract representations instead of narrow, task-specific outputs. This early investigation was the foundation for further research in the area of utilising representations in other downstream tasks in this thesis.

Chapter 3

Single-Shot Variational Autoencoder for Learning Representations of Multiple Objects in Images

This chapter presents the part of the PhD research dedicated to the advancement of methods for learning representations of multiple objects within images. The central contribution of this work is the development of a novel variational autoencoder architecture that integrates a pre-trained object detection model as a robust feature extractor in object-centric scenarios. Specifically, by extending the intermediate representations extracted by the convolutional backbone with additional spatial grid-based attention encoders, the proposed model enables encoding object representations in an efficient, single-shot manner. By doing so, this research addresses key limitations of its contemporary reference object-centric representation learning methods, particularly their difficulty in handling scenes with multiple objects of varying sizes and their constrained scalability due to sequential glimpse encoding and single-grid inference. The proposed approach utilises multi-scale feature pyramid to achieve scale-invariant, parallel encoding of object representations, resulting in improved performance and robustness of representations.

The motivation for working on this approach was grounded in the limitations of existing methods at the time of the study's conception (2021). Consequently, the methods selected as baselines for comparison reflect the the state-of-the-art available at that time; it is worth noting, however, that subsequent studies have since proposed alternative approaches, addressing the challenge of varying sizes and scalability through different methodologies.

The findings and methodology detailed here were published in 2022 as the associated publication [208]. As the principal author, I was responsible for developing the foundational concept, designing and executing the research methodology, managing the experimental evaluations, and preparing the manuscript along with all figures and tables.

The structure of this chapter is organised as follows: Section 3.1 explains the idea behind this research, its motivation, and Section 3.2 presents an overview of related works in the field at the time of working on this area. Section 3.3 provides a detailed description of the proposed method, emphasising the extraction of features via the pre-trained object detection model and the specific setup for representation learning. Section 3.4 outlines the experimental procedures, reports the results and provides their thorough analysis. Finally, Section 3.5 provides conclusions and highlights the contributions of this research.

3.1 Motivation

The ability to discriminate and reason about individual objects in an image is one of the important tasks of computer vision, which is why object detection and instance segmentation tasks have drawn vast attention from researchers throughout the years. The latest advances in artificial intelligence require a more insightful analysis of the image to provide more profound reasoning about its contents. It can be achieved through representation learning, which facilitates extracting useful information about objects, allowing transferring more general knowledge to other tasks [8]. One can see multi-object representation learning as a natural extension of the aforementioned computer vision tasks. Here, the objective is to produce a valuable abstract feature vector of each of the inferred objects and hence produce a structured representation of the image, allowing for its more insightful understanding.

Recently, the most successful methods are based on the variational autoencoder (VAE) framework [101, 150], with structured latent space, which includes individual objects' representations. The original approach consists in extracting object latent vectors with a recurrent network [14, 49, 50, 53, 65]. Alternatively, each object's representation can be produced with a single forward pass through the network by employing a convolution-based single-shot approach [31, 120]. However, these methods are limited by a single feature map utilised to create objects' latent vectors and hence cannot be used when object sizes vary.

This chapter presents a single-shot method for learning multiple objects' representations, called Single-Shot Detect, Infer, Repeat (SSDIR). It is a convolutional generative model applying the single-shot approach with a feature pyramid for learning valuable, scale-invariant object representations. By processing multi-scale feature maps, SSDIR can attend to objects of varying sizes and produce high-quality latent representations directly, without the need to extract objects' glimpses and process them with an additional encoder network. The ability to focus on individual objects in the image is improved by leveraging knowledge learned in an SSD [122] object detection model. Through experiments, the SSDIR model's performance is evaluated on multi-scale scattered MNIST digits, CLEVR [93] and WIDER FACE [193] datasets with other single-shot approaches, proving the ability to focus on individual objects of varying sizes in complicated scenes, as well as the improved quality of objects' latent representations, which can be successfully used in other downstream problems, despite the use of an uncomplicated convolutional backbone.

The contributions of this chapter are as follows. We present a model that enhances multi-object representation learning with a single-shot, feature pyramid-based approach, retaining probabilistic modelling of objects. We provide a framework for generating object representations directly from feature maps without extracting and processing glimpses, allowing easier scaling to larger images. We compare the method with other single-shot multi-object representation learning models and show its ability to attend to objects, the improved latent space quality, and applicability in various benchmark problems.

3.2 Related Works

Multi-object representation learning has recently been tackled using unsupervised, VAE-based models. Two main approaches include sequential models, attending to a single object or part of the image at a time, and single-shot methods, which generate all representations in a single forward pass through the network.

The original approach to this problem was presented by Ali Eslami *et al.* in [53]. The *Attend, Infer, Repeat* (AIR) model assumes a scene to consist of objects, represented with *what* vector, describing the object's appearance, *where* vector indicating its position on the image and *present* vector, describing if it is present in the image, controlling termination of the recurrent image processing. The model attends to a single object at a time, generating representations sequentially with a recurrent network until a non-present object is processed. Other studies, including [66] and [163] proposed a different approach, where object representations are learned using Neural Expectation-

Maximisation, without structuring the latent representations explicitly. These methods suffer from scaling issues, not being able to deal with complex scenes with multiple objects.

Alternatively, an image might be described with a scene-mixture approach, as in MONet [14], IODINE [65] and GENESIS [49, 50]. Here, the model does not explicitly divide the image into objects but instead generates masks, splitting the scene into components, which the model encodes. In the case of MONet and GENESIS, each component is attended to and encoded sequentially, while IODINE uses amortised iterative refinement of the output image. However, these methods are not a good fit for learning object representations in an image, as scene components usually consist of multiple objects. Furthermore, masks that indicate particular objects limit the model's scalability due to this representation requiring more memory than bounding box coordinates.

GENESIS belongs to a group of methods, which focus on the ability to generate novel, coherent and realistic scenes. Among them, one should notice recent advances with methods leveraging generative adversarial networks (GANs), such as RELATE [47] or GIRAFFE [134]. Compared to VAE-based methods, they can produce sharp and natural images, which are more similar to original datasets. However, these models do not include an explicit image encoder, and therefore cannot be applied to multi-object representation learning directly. What is more, the process of training GANs tends to be longer and more complicated than in the case of VAEs.

Recently, methods such as GMAIR [206] postulate that acquiring valuable *what* object representations is crucial for the ability to use object encodings in other tasks, such as clustering. Here, researchers enhanced the original *what* encoder with Gaussian Mixture Model-based prior, inspired by the GMVAE framework [68]. In this chapter, the importance of the *what* object representation is emphasised as well, evaluating its applicability in downstream tasks.

One of the promising methods of improving scalability of VAE-based multi-object representation learning models was presented in SPAIR [31], where the recurrent attention of the original AIR was replaced with a local feature maps-based approach. In analogy to single-shot object detection models like SSD [122], the SPAIR first processes image with a convolutional backbone, which returns a feature map with dimensions corresponding to a fixed-sized grid. Each cell in the grid is then used to generate the locations of objects. Object representations' are inferred by processing these cells sequentially, generating *what*, *depth* and *present* latent variables, describing its appearance, depth in the scene, and the fact of presence. This approach has recently been extended in SPACE [120], which fixes still existing scalability issues in SPAIR by employing parallel latent components inference. Additionally, the authors used the scene-mixture approach to model the image background, proving to be applicable for learning objects' representations in more complex scenes. However, both methods rely on a single grid of fixed size, which makes it difficult for this class of models to attend to objects of highly varying sizes. What is more, both of them employ glimpse extraction: each attended object is cut out of the input image and processed by an additional encoder network to generate objects' latent representations; this increases the computational expense of these methods.

The latest advances in the field of multi-object representation learning try to apply the aforementioned approaches for inferring representations of objects in videos. SQAIR [104] extends the recurrent approach proposed in AIR for sequences of images by proposing a propagation mechanism, which allows reusing representations in subsequent steps. A similar approach was applied to single-shot methods by extending them with a recurrent network in SILOT [32] and SCALOR [92]; here, the representations were used in the object tracking task. An interesting approach was proposed by Henderson and Lambert [77]. Authors choose to treat each instance within the scene as a 3D object; the image is then generated by rendering each object and merging their 2D views into an image. This allows for a better understanding of objects' representations, at the cost of significantly higher computational complexity.

3.3 Single-Shot Detect, Infer, Repeat

SSDIR (Single-Shot Detect, Infer, Repeat) is a neural network model based on a variational autoencoder architecture [101, 150] as shown in Figure 3.1; its latent space consists of structured objects' representations $\mathbf{z}$, enhanced by leveraging inductive priors learned in a single-shot object detection model SSD [122], both sharing the same convolutional backbone.

The model extends the idea of single-shot object detection by incorporating them into a probabilistic generative framework. Let $\mathbf{x}$ denote an input image containing all relevant (i.e. detected by a pre-trained SSD detector) objects present in the image. SSDIR assumes that this image is generated from a structured latent representation $\mathbf{z}$, which captures object-specific attributes across multiple spatial locations. These latent variables are associated with each grid cell in the feature pyramid produced by the SSD convolutional backbone and are sampled from a prior distribution $p(\mathbf{z})$. SSDIR parametrises the conditional distribution $p_\theta(\mathbf{x} \mid \mathbf{z})$ using the decoder network θ. Then, the generative model can be formulated as a standard VAE decoder (Equation (3.1)).

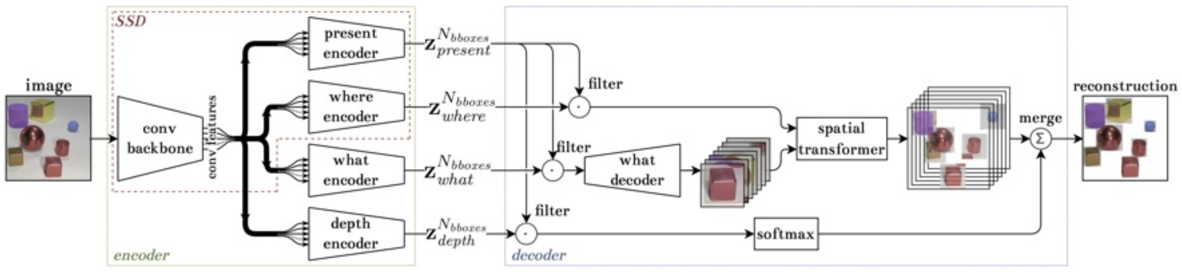


Figure 3.1: Overview of the SSDIR model architecture. The model consists of two fully convolutional neural networks: an *encoder* and a *decoder*. The encoder uses a convolutional backbone as a feature extractor (shared with a pre-trained SD object detector), which builds a pyramid of multi-scale features processed by each latent component encoder. Each object's position $\mathbf{z}_{\text{where}}$ and presence $\mathbf{z}_{\text{present}}$ latent vectors are computed using the SSD detection heads, indicating grid cells, which refer to detected objects. In parallel, additional convolutional encoders generate the appearance $\mathbf{z}_{\text{what}}$ and depth $\mathbf{z}_{\text{depth}}$ latents from the same feature maps. In the decoder, latents corresponding to detected objects are filtered based on presence indicators. The appearance vectors are decoded into object reconstructions, which are spatially transformed back into their respective positions using the affine parameters from $\mathbf{z}_{\text{where}}$ in the *spatial transformer* module. The final image is composed by merging all transformed reconstructions, weighted by depth-based softmax to handle occlusions.

$$p(\mathbf{x}) = \int p_\theta(\mathbf{x} \mid \mathbf{z})\, p(\mathbf{z})\, d\mathbf{z} \tag{3.1}$$

To perform inference with this generative framework, SSDIR employs the variational inference technique. It approximates the intractable true posterior with a function $q_\phi(\mathbf{z} \mid \mathbf{x}) \approx p(\mathbf{z} \mid \mathbf{x})$, parametrised by ϕ (the encoder network's parameters). This enables training via maximisation of evidence lower bound (ELBO) as the model's loss function (Equation (1.2)).

3.3.1 Object Representation

SSDIR extends the grid-based approach with a feature pyramid inspired by the SSD object detector to enable multi-scale object encoding. Each spatial location in multi-scale grids is associated with a candidate object, which is described using a structured latent representation. For each i-th object, this representation consists of the following four latent components:

- $\mathbf{z}^i_{\text{where}} \in \mathbb{R}^4$ – the object's bounding box position and size, specifying the object's spatial attributes,

- $\mathbf{z}^i_{\text{present}} \in \{0, 1\}$ – a binary variable indicating whether a valid object is present at the location corresponding with the given cell,
- $\mathbf{z}^i_{\text{what}} \in \mathbb{R}^D$ – D-sized vector capturing the object visual features,
- $\mathbf{z}^i_{\text{depth}} \in \mathbb{R}$ – a scalar representing the object's relative depth in the scene; higher values correspond to objects appearing closer in the final rendering.

To facilitate object discovery and reduce the complexity of model training, SSDIR reuses a pre-trained SSD object detection model to obtain bounding box position and size, as well as the detected object confidence score. These outputs are directly used to produce $\mathbf{z}_{\text{where}}$ and $\mathbf{z}_{\text{present}}$ as defined in Equations (3.2) and (3.3).

$$\mathbf{z}^i_{\text{where}} = \begin{bmatrix} x_i & y_i & w_i & h_i \end{bmatrix} \tag{3.2}$$

$$\mathbf{z}^i_{\text{present}} \sim \text{Bernoulli}\left(\beta^i\right) \tag{3.3}$$

where:

i refers to the index of a cell in the feature pyramid,
x_i, y_i are the bounding box' centre coordinates,
w_i, h_i are the bounding box' width and height dimensions,
$$\beta^i = \begin{cases} \arg\max_k \mathbf{c}_i & \text{if an object detected in the cell } i, \\ 0 & \text{otherwise,} \end{cases}$$
$\mathbf{c}$ is the vector of the object's predicted class confidences.

The two remaining latent components: $\mathbf{z}_{\text{what}}$ and $\mathbf{z}_{\text{depth}}$ are modelled with Gaussian distributions, as shown in Equations (3.4) and (3.5).

$$\mathbf{z}^i_{\text{what}} \sim \mathcal{N}\left(\boldsymbol{\mu}^i_{\text{what}}, \boldsymbol{\sigma}^i_{\text{what}}\right) \tag{3.4}$$

$$\mathbf{z}^i_{\text{depth}} \sim \mathcal{N}\left(\boldsymbol{\mu}^i_{\text{depth}}, \boldsymbol{\sigma}^i_{\text{depth}}\right) \tag{3.5}$$

where:

$\boldsymbol{\mu}^i_{\text{what}}, \boldsymbol{\mu}^i_{\text{depth}}$ are the mean vectors, encoded with *what* and *depth* encoders,
$\boldsymbol{\sigma}^i_{\text{what}}, \boldsymbol{\sigma}^i_{\text{depth}}$ are the corresponding standard deviations, which are treated as the model's fixed hyperparameters.

3.3.2 SSDIR Encoder Network

The encoder network, denoted $q_\phi(\mathbf{z} \mid \mathbf{x})$, is responsible for inferring the structured latent representation $\mathbf{z}$ from an input image $\mathbf{x}$. It is implemented using a fully convolutional backbone (VGG11), which accepts images of resolution $300 \times 300 \times 3$, extended with a feature pyramid, and processed by additional convolutional encoders, as shown in Figure 3.1.

Given an input image $\mathbf{x} \in \mathbb{R}^{H \times W \times 3}$, a grid-based spatial attention mechanism divides the image into a regular grid of cells. Each cell corresponds to a region of the original image and is responsible for attending to at most one object within its local spatial context. This grid is created from feature maps encoded by a convolutional backbone and leverages the spatial invariance property of deep convolutional neural networks [112]. Each cell is encoded with a D-sized vector ($b(\mathbf{x}) \in \mathbb{R}^{H_b, W_b, D}$).

Each cell encoding $b_{ij} \in \mathbb{R}^D$ contains a representation of the visual features corresponding to the area in the original image. They are then used to infer object properties via learned transformations. In single-shot object detection models [12, 122], prediction heads transform the encodings into bounding box locations and class logits. In SSDIR, spatial grid-based attention is used to infer object representations by applying additional convolutional sub-encoders on cell features:

- the *where* encoder applies a single convolutional layer with 3×3 kernel size and four output channels (corresponding to bounding box parameters),
- the *present* and *depth* encoders each use a single convolutional layer with 3×3 kernel size and one output channel per feature map (for single presence probability and depth, respectively),
- the *what* encoder involves multiple 3×3 convolutional layer with ReLU activations, ultimately returning D-sized vector for each cell in each feature pyramid grid.

All latent variables are produced in parallel for each feature pyramid cell, without requiring sequential attention or glimpse extraction.

The backbone's, as well as *where* and *present* encoders' weights are transferred from a pre-trained SSD model, detecting objects of interest in a given task. These weights remain frozen during training; *what* and *depth* encoders, which share the same pre-trained backbone, are trained jointly with the decoder. This approach enables highly parallel inference across all candidate object locations.

3.3.3 SSDIR Decoder Network

The decoder network in SSDIR reconstructs the input image from inferred object-centric latent variables, forwarded from the encoder network. Here, the reconstruction is restricted to regions corresponding to objects deemed present by the encoder, focusing on salient scene content. The decoding process starts with filtering the latent variables using $\mathbf{z}_{\text{present}}$, retaining only those entries for which an object has been detected. Next, for each present object $i = 1, ..., K$, the corresponding $\mathbf{z}^i_{\text{what}}$ vector is passed through a shared convolutional *what* decoder, producing K localised image patches of size $64 \times 64 \times 3$, representing each detected object's appearance.

These object patches are then spatially transformed to their original location according to the bounding box parameters in $\mathbf{z}^i_{\text{where}}$, implemented via a Spatial Transformer Module [87]. The resulting set of K $300 \times 300 \times 3$ images of object patches placed at their inferred positions is merged using a weighted sum, using softmax-normalised, filtered depth values derived from $\mathbf{z}_{\text{depth}}$. The output of the model might then be normalised with respect to the maximum intensity of pixels in the reconstruction to improve the fidelity of the reconstruction.

SSDIR does not require special preprocessing of the image, apart from the standard normalisation used widely in convolutional neural networks. Originally, the model does not model the scene background explicitly, as its focus lies in the object-centric representations, which encourages the model to extract and reconstruct only the most informative parts of the image. However, an optional background encoder could be incorporated; by treating the background as an extra object, which is transformed to fill the entire image and put behind all other objects, the model could provide higher fidelity of reconstructions.

Crucially, the parallel nature of the model is preserved in the decoder. The entire pipeline, including filtering, decoding, transforming, and depth-based merging is implemented using fully parallelisable matrix operations, maintaining computational efficiency and scalability.

3.3.4 Training

The SSDIR model[1] is trained with a modified ELBO loss function. Building upon the standard VAE loss (Equation (1.2)), which intuitively includes reconstruction error of an entire image and KL divergence for latent and prior distributions, SSDIR

[1]Code available at: `https://github.com/piotlinski/ssdir`

extends it with an additional term, specifically a normalised sum of each detected object's reconstruction error. This design encourages the model to learn expressive object appearance and proper depth ordering while preserving transformation function continuity thanks to KL divergence-based regularisation. The complete form of the loss function is shown in Equation (3.6).

$$\begin{aligned}\mathcal{L}(\mathbf{x},\theta,\phi) &= \lambda_{\text{obj}}\mathbb{E}_{\mathbf{z}}\left[\log p_\theta(\mathbf{x}\mid\mathbf{z})\right] + \lambda_{\text{rec}}\frac{1}{K}\sum_i^K \mathbb{E}_{\mathbf{z}_i}\left[\log p_\theta\left(\mathbf{x}_i^{\text{obj}}\mid\mathbf{z}_i\right)\right] \\ &\quad - \lambda_{\text{what}} D_{\text{KL}}\left(q_\phi(\mathbf{z}_{\text{what}}\mid\mathbf{x}) \parallel p(\mathbf{z}_{\text{what}})\right) \\ &\quad - \lambda_{\text{depth}} D_{\text{KL}}\left(q_\phi(\mathbf{z}_{\text{depth}}\mid\mathbf{x}) \parallel p(\mathbf{z}_{\text{depth}})\right)\end{aligned} \tag{3.6}$$

where:

$\mathbb{E}_{\mathbf{z}}\left[\log p_\theta(\mathbf{x}\mid\mathbf{z})\right]$ is the log-likelihood of the full image reconstruction,
$\mathbb{E}_{\mathbf{z}_i}\left[\log p_\theta\left(\mathbf{x}_i^{\text{obj}}\mid\mathbf{z}_i\right)\right]$ is the expected reconstruction log-likelihood of the i-th detected object,
$\lambda_{\text{obj}}, \lambda_{\text{rec}}, \lambda_{\text{what}}, \lambda_{\text{depth}}$ are scalar weights that control the relative contribution of each loss component,
K is the number of objects detected by the SSD network in a given image.

For both $\mathbf{z}_{\text{what}}$ and $\mathbf{z}_{\text{depth}}$, standard Gaussian priors are assumed, i.e. $\mathcal{N}(\mathbf{0},\boldsymbol{I})$. The training objective is described by Equation (3.7) for each image $\mathbf{x}_i$ in the training dataset. The model is trained jointly with gradient ascent using Adam as the optimiser, utilising the reparametrisation trick for backpropagating gradients through the sampling process. The training procedure is unsupervised with respect to the appearance and depth components, however the SSD backbone's as well as *where* and *present* encoders' weights are initialized from a pre-trained SSD model and kept frozen during training.

$$\theta^*,\phi^* = \arg\max_{\theta,\phi}\sum_i \mathcal{L}(\mathbf{x}_i,\theta,\phi) \tag{3.7}$$

3.4 Experiments

This section presents an empirical evaluation of the SSDIR model. During experimentation, SSDIR was compared against two representative baseline methods: SPAIR [31] and SPACE [120], both following the single-shot, grid-based approach for multi-object representation learning. The evaluation was designed to assess the model's capacity to handle objects of varying sizes, learning interpretable and reusable object representations. This research includes an analysis of the quality of reconstructions produced by each model, as well as the utility of representations in a downstream classification task. Additionally, this section shows an ablation study on the influence of the dataset characteristics on the performance of SSDIR.

For SPAIR, the original architecture (based on a fully connected encoder) was modified to use a convolutional encoder to improve its performance on more complex datasets. As the focus of this evaluation is on learning object-centric representation, background modelling was omitted across all compared methods. SPAIR does not explicitly include a background component, whereas in SPACE only the outputs of the foreground module were analyzed, as this module is responsible for reconstructing individual objects within the scene.

For reference, Table 3.1 presents a comparative summary of the evaluated methods, including an additional pipeline for theoretical reference: an object detector followed by a spatial transformer network (STN) for extracting glimpses and a variational autoencoder (VAE) for encoding object appearance. This reference is denoted as *basic VAE*.

Table 3.1: Summary of the key differences between SSDIR and baseline models. A method is considered semi-supervised if it incorporates a pre-trained object detector. "Glimpse-based" refers to models that extract and encode object subimages, while "single-shot" methods infer all object representations directly from feature maps.

	basic VAE	**SPAIR** [31]	**SPACE** [120]	**SSDIR**
supervision level	semi-supervised	unsupervised	unsupervised	semi-supervised
object encoding strategy	glimpse-based	glimpse-based	glimpse-based	single-shot
handles varying object sizes	✓	✗	✗	✓
selective object type attention	✓	✗	✗	✓
parallel object encoding	✗	✗	✗	✓

Datasets. The experimental evaluation utilises three datasets reflecting varying levels of visual complexity, aiming at validating the model in simple problems and proving its performance on more realistic data.

1. **Multi-scale scattered MNIST**[2] is a synthetic dataset, a variant of the widely used scattered MNIST dataset where digit sizes are sampled from a configurable range to simulate scale variation; this dataset served as a controlled environment for evaluating the model's robustness to dataset parameters, especially object overlapping and high variability in size,

2. **CLEVR** [93] is a synthetic benchmark dataset designed originally for visual reasoning tasks; featuring scenes composed of multiple 3D-rendered objects, which vary in shape, material and size, CLEVR is widely used for evaluating object-centric models,

3. **WIDER FACE** [193] is a real-world benchmark for face detection, containing highly cluttered scenes with multiple people, varying conditions and face sizes; this dataset was selected to test the model's ability to selectively focus on objects of specific semantic type based on detection priors.

3.4.1 Experiment 1: Qualitative Analysis of Per-Object Reconstructions

This experiment reviews the quality of images' and objects' reconstructions of SSDIR and baseline methods. The quality of object-centric reconstructions serves here as an indicator of representation quality. Figure 3.2 shows a qualitative comparison between SSDIR and the baseline methods: SPAIR [31] and SPACE [120], across the three evaluation datasets. For each model, samples from the held-out test set are shown, together with the corresponding reconstruction and a subset of reconstructed individual objects to illustrate the quality of objects' representations. Due to variability in object count and a high number of utilised objects, only a selection of reconstructions is visualised.

Both SPAIR and SPACE can reconstruct full-scene images from the scattered MNIST dataset correctly. However, inspection of the inferred *where* boxes reveals that these models struggle to assign a single object representation to individual objects. Due to a fixed-size grid in both of these models, their limited capacity for scale variation leads to fragmenting larger digits across multiple latent variables. In SPAIR all objects' reconstructions tend to consist of disjoint digit parts merged together, while SPACE performs well only on digits of sizes similar to its preset grid configuration, and splits larger digits into multiple components. SSDIR is able to detect and reconstruct the

[2] https://github.com/piotlinski/multiscalemnist

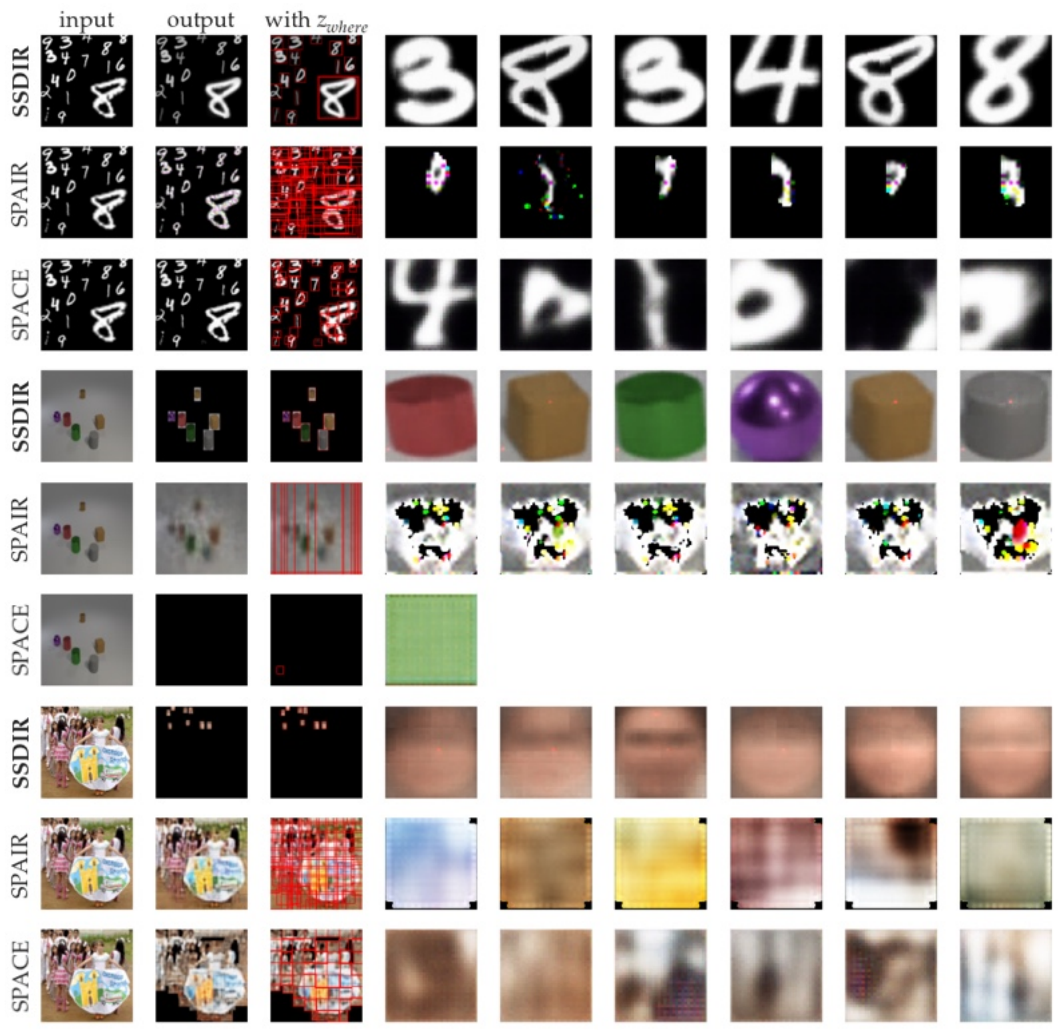


Figure 3.2: Qualitative comparison of inference results for SSDIR, SPAIR [31] and SPACE [120] across three datasets: multi-scale MNIST (top), CLEVR (middle) and WIDER FACE (bottom). For each model and image, the first column shows the input image, the second and third present the full-scene reconstruction (without and with inferred bounding boxes $\mathbf{z}_{\text{where}}$, respectively). The remaining columns display individual object patches decoded from the object-centric latents. SSDIR produces coherent, well-localised object reconstructions even in cluttered and complex scenes. In contrast, SPAIR and SPACE often tend to fragment larger objects, allocate latents to background regions, or fail to isolate full objects in more complex scenes, leading to redundant representations.

MNIST image accurately: the use of a multi-scale feature pyramid allows for attending to entire objects across a wide range of scales, resulting in scale-invariant reconstructions, which are then mapped to the output image according to tight bounding boxes.

On more complex datasets such as CLEVR and WIDER SPACE, SPAIR fails to learn meaningful object-level representations. Instead of capturing discrete entities, it tends to segment the image into regular rectangular patches aligned with the grid structure, often encompassing a bigger part of an image. In CLEVR, this behaviour leads to blurry and geometrically distorted reconstructions, further degraded by a transparency mask applied in this model during reconstruction. The tendency to divide the image into rectangles is exhibited in the WIDER FACE dataset. Despite low reconstruction error and overall fair quality of image reconstructions, the representations are semantically uninformative.

SPACE exhibits comparable limitations. Despite an extensive hyperparameter search, particularly for the object size parameter in the foreground module, the model fails to learn valid object decompositions on the CLEVR dataset. Instead, it defers most of the reconstruction to the background module, which aggregates multiple objects into a single representation. This behaviour, consistent with issues reported by other users[3], leads to noisy and unstructured object representations. When applied to the WIDER FACE dataset, SPACE exhibits similar behaviour to SPAIR, dividing the image into rectangular partitions rather than meaningful entities, resulting in an acceptable reconstruction quality but poor object-level semantics of representations.

SSDIR demonstrates strong performance on both CLEVR and WIDER FACE. On CLEVR, the model accurately localizes and reconstructs individual objects with high fidelity, reflecting successful appearance encoding. On WIDER FACE, SSDIR is capable of identifying and reconstructing individual faces. Although the visual quality of reconstructed patches is limited by the relatively shallow convolutional backbone and simple decoder architecture, the model produces distinct, semantically consistent object representations. Importantly, due to the use of a multi-scale feature pyramid, featuring multiple grids, SSDIR may produce redundant reconstructions for individual objects, which could motivate the use of post-processing methods such as non-maximum suppression.

3.4.2 Experiment 2: Evaluating the Structure and Utility of the Latent Space

This section presents the analysis of the SSDIR model's latent space and compares it with the latent space of SPAIR and SPACE. Figure 3.3 visualises latent spaces for the scattered multi-scale MNIST dataset. For each model, the test subset was processed to generate latent vectors of each image. Then, individual objects' $\mathbf{z}_{\text{where}}$ vectors were compared with ground truth bounding boxes, and labels were assigned to latent representations by choosing the maximum intersection over union between predicted and true boxes. Each $\mathbf{z}_{\text{what}}$ vector was then embedded into two-dimensional space using t-SNE.

[3]`https://github.com/zhixuan-lin/SPACE/issues/1`

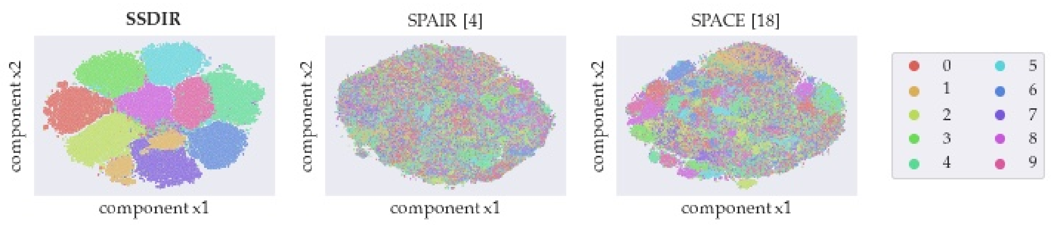


Figure 3.3: t-SNE visualisation of $\mathbf{z}_{what}$ latent representations for the multi-scale scattered MNIST test dataset. Each point corresponds to an individual object representation, with colours indicating ground truth digit classes, assigned by matching predicted $\mathbf{z}_{\text{where}}$ bounding boxes to ground truth using maximum Intersection over Union. SSDIR shows a well-structured latent space with separated digit clusters. In contrast, SPAIR (middle) and SPACE (right) exhibit no clusterisation in the latent space, largely due to the tendency to fragment larger objects.

A comparison of the latent spaces reveals that SSDIR produces a latent space in which individual object classes (in this case: digits) are clearly separated. Additionally, the latent manifold is smooth and continuous, with no apparent distortions. While the latent spaces generated by the reference methods (SPAIR and SPACE) are continuous as well, they do not support clear separation between object classes. This limitation likely stems from their tendency to segment larger objects into smaller fragments, constrained by their predefined object size, as shown in Subsection 3.4.1.

To evaluate the usefulness of the learned latent representations, a downstream task of digit classification was performed. For each method, models were trained on the multi-scale scattered MNIST dataset using three random seeds. Then, latent representations were extracted for both train and test subset, with digit labels assigned to each object's $\mathbf{z}_{\text{what}}$ based on Intersection over Union (IoU) between $\mathbf{z}_{\text{where}}$ and the corresponding ground truth bounding boxes. Then, a logistic regression classifier was trained for each model and seed to predict digit class from latent representation. Classification results on the test set are presented in Table 3.2. The SSDIR latent space demonstrates superior utility for the classification task, consistently achieving higher performance across all evaluation metrics compared to the baseline methods. The weaker results of the reference methods are similarly caused by the fragmentation behaviour, which leads to less discriminative latent representations.

Table 3.2: Downstream digit classification performance using object-centric latent representations and logistic regression. Results are averaged over 3 random seeds.

	accuracy	**precision**	**recall**	**F1-score**
SSDIR	**0.9789** $\pm$ 0.0016	**0.9787** $\pm$ 0.0017	**0.9786** $\pm$ 0.0016	**0.9786** $\pm$ 0.0016
SPAIR [31]	0.1919 $\pm$ 0.0073	0.1825 $\pm$ 0.0087	0.2019 $\pm$ 0.0092	0.1803 $\pm$ 0.0102
SPACE [120]	0.2121 $\pm$ 0.0432	0.2020 $\pm$ 0.0431	0.2158 $\pm$ 0.0435	0.1992 $\pm$ 0.0462

3.4.3 Ablation Study: Influence of Dataset Properties

To assess how characteristics of the input data affect model performance, an ablation study was conducted using variants of the multi-scale scattered MNIST dataset. Each dataset was generated by randomly selecting cells from a pre-defined grid and placing MNIST digits of varying sizes and offsets within them. The generation process was controlled by hyperparameters, such as the number and size of grids, as well as the minimum and maximum size of a digit. These parameters were systematically varied to study their individual effect on model performance.

For each dataset configuration, an SSDIR model was trained and subsequently evaluated on its test subset using mean squared reconstruction error (MSE, Equation 1.9) as the evaluation metric. The results of this analysis are presented in Figure 3.4.

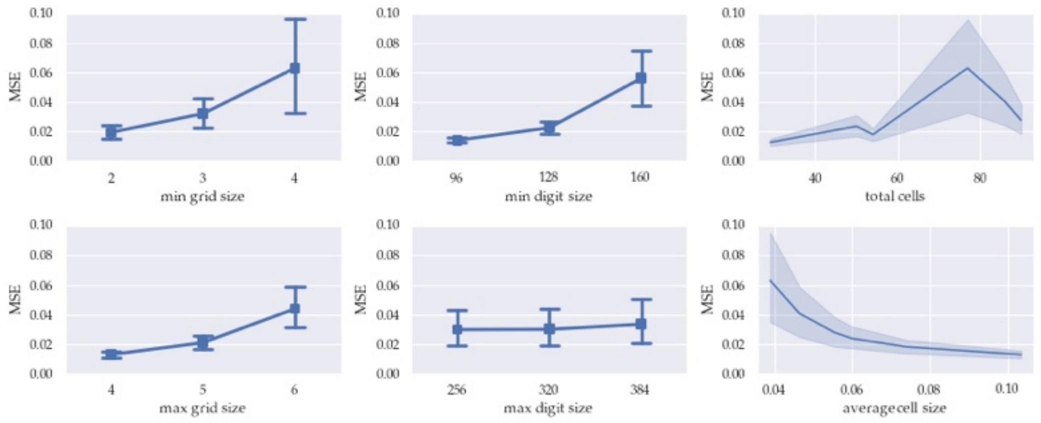


Figure 3.4: Ablation study showing the impact of the dataset generation parameters on model performance, measured by reconstruction MSE. Parameters leading to a dataset with larger or more occluded digits negatively affect the model's reconstruction quality. SSDIR works best on datasets containing smaller, non-overlapping digits. Error bars indicate standard deviation across three runs.

The study reveals the model's sensitivity to the size of objects present in the input image. Larger digits lead to an increase in reconstruction error, likely due to the mismatch between the fixed-sized latent reconstruction ant the required upscaling for transformation in the output image. Another factor contributing to the higher error is the number of digits in the image, which increases the likelihood of occlusions. This effect is particularly visible when increasing the minimum and maximum grid size, as well as the total number of cells used during dataset generation.

3.5 Summary

This chapter presented SSDIR, a single-shot convolutional generative model designed to learn scale-invariant object representations. The proposed approach extends existing multi-object representation learning frameworks with a multi-scale feature pyramid and leveraging knowledge learned by a pre-trained object detection model. The effectiveness of SSDIR was demonstrated through qualitative and downstream evaluation, which confirmed its ability to infer structured, scale-invariant representations of objects across both simple and moderately complex visual scenes.

However, several challenges remain, particularly in dealing with complex, real-world data. SSDIR exhibits limitation related to fixed input resolution, which makes it struggle with complicated scenes, especially in case of occlusions. What is more, learning representations of objects in complex scenes could be improved by using a more advanced feature extractor and reconstructing approach. The subsequent chapters address these aspect by exploring more expressive encoder architecture and improved image reconstruction approaches.

Chapter 4

Implicit Temporal Encoder for Multi-Object Representation Learning on Videos

This chapter presents the part of the PhD research focused on extending the multi-object representation learning model to the temporal domain, by applying it to videos and addressing the challenge of capturing temporally coherent object representations. The model proposed in this work, Recurrent Detect, Infer, Repeat (RDIR), enhances the previously developed semi-supervised single-shot model (SSDIR) by introducing an implicit recurrent mechanism, which refines internal feature maps of the convolutional backbone through time directly, without requiring explicit object association and tracking. Furthermore, several architectural enhancements are proposed to overcome the limitations of SSDIR, including a significantly more powerful convolutional backbone and cross-scale mixer modules that promote sharing context across multi-scale convolutional grids.

The motivation for this research emerged during the analysis of the static image-based method developed in the earlier phase. Following the move towards complex, real-world scenarios in this research area, the need for ensuring temporally-stable representations became increasingly important. The selected reference methods were the state-of-the-art models for object-centric learning in videos at the time of developing this approach, utilising the most prominent approaches: spatial attention models, scene-mixture models and transformer-based architectures, complemented by an improved variant of the image-based model from the previous phase, used as a baseline for assessing the benefits of the proposed temporal extension.

The methodology and results reported in this chapter are part of the publication [209], for which I served as the principal author. I was responsible for the development of the method, design and implementation of the model and experimental setup, evaluation and analysis of the results, and preparation of the manuscript, including visual materials.

The structure of this chapter is organised as follows: Section 4.1 introduces the motivation, and the relevant literature in the domain of video-based object-centric learning is reviewed in Section 4.2. Section 4.3 presents the proposed RDIR model, detailing the recurrent encoder and architectural improvements over SSDIR. Section 4.4 outlines the experimental framework, presents evaluation results in both synthetic and real-world settings, and includes ablation studies. Finally, Section 4.5 summarises the findings and discusses the broader implications of the proposed approach.

4.1 Introduction

Human's ability to perceive and understand the visual world allows us to comprehend the compositionality of the scene captured by our eyesight. This cognitive process is based not only on observing the surroundings at a given moment but also on comprehending the temporal variance of the scene and how the objects move and interact with each other, enabling a deep understanding of visual scenes. The complexity of this natural process is a topic of vivid research. Recent machine learning and computer vision methods aim at learning similar comprehension as a result of supervised learning for particular tasks, such as object detection, instance segmentation, visual question answering etc. A group of methods allowing for a more general understanding of scenes is often referred to as multi-object representation learning models.

Downstream models that operate on object-centric representations are usually easier to train; this approach can also reduce the amount of data required to achieve good performance. However, the success of these algorithms relies heavily on the quality of embeddings produced by the representation learning model [8]. Recent methods, building upon previous developments in this area, extend the image-based approach to videos and infer temporal changes of objects in scenes: their movement, variation of shape, etc. [32, 92, 95, 104, 197]. This makes it possible to capture and understand the underlying dynamics of complex scenes as they change through time.

Recent developments in multi-object representation learning have increasingly moved from fully unsupervised methods toward semi- and self-supervised approaches [48, 103, 208]. By incorporating additional knowledge these approaches provide more robust object representations and can attend to individual objects in complex scenes more easily. However, many of these methods rely on computationally expensive training procedures, often requiring multiple days of training on high-performance GPUs to reach success. Furthermore, the quality of the representations and their temporal and spatial stability have been given insufficient attention, as most models are compared by the quality of scene decomposition (i.e. the accuracy of segmentation masks).

This chapter addresses the challenge of capturing object-centric, temporally, and spatially stable representations in videos. It introduces RDIR, a method for multi-object representation learning on videos that incorporates a recurrent mechanism to provide temporally consistent object representations. The approach builds on the recent shift toward semi- and self-supervised learning by extending a pre-trained one-stage, multi-scale object detection model with a recurrent mechanism for encoding each object representation without the need for additional supervision. By applying a pre-trained object detection model, RDIR offers a deeper understanding of detected objects, which can be obtained on any unannotated dataset, with improved scalability and shorter training. Experimental comparisons with existing video-based object-centric models demonstrate that RDIR captures more consistent object representations and supports downstream tasks effectively.

The contributions of the paper are as follows. We introduce a model for learning object-centric representations on videos and explain the theoretical underpinnings. We present a comparison of the performance and the quality of the representations of this model and other state-of-the-art methods for multi-object representation learning. We provide an experimental approach for evaluating the temporal and spatial coherence of representations and show how dataset characteristics and model architecture influence the performance of the approach.

4.2 Related Works

The presented research lies within the domain of multi-object visual representation learning, with the objective of learning structured representations of multiple objects present in a scene.

Multi-object representation learning on images. This task is commonly approached using unsupervised, VAE-based models [101, 150]. These methods can be broadly classified into spatial-attention models and scene-mixture models. Spatial-attention models use geometrically defined regions (typically rectangular bounding boxes) to attend to individual objects, allowing for efficient processing and highly interpretable inference. These models can work by iteratively predicting subsequent objects (as in AIR [53]) or predicting all objects with a single-shot model (SPAIR [31] and SPACE [120]). However, their strong inductive bias limits their ability to handle objects of irregular shapes and struggle with large objects, often resulting in over-segmentation or fragmentation. In contrast, scene-mixture models such as MONet [14], IODINE [65], Slot Attention [124] or GENESIS [49, 50] do not restrict the shape of masks used to split the scene into parts, enabling the model to infer on more complex scenes. However, these methods are expensive to train and infer, as they rely on recurrent mask refinement or sequential attention. Moreover, they tend to encapsulate multiple objects in a single representation on complex datasets, unable to discover meaningful entities.

RDIR builds on the approach suggested in SSDIR (Chapter 3), where a spatial-attention model is extended with a multi-scale convolutional encoder and a flexible attention mechanism that supports variable object sizes. This architecture retains the computational efficiency of spatial-attention models while improving the model's ability to attend to diverse object scales in a single forward pass.

Multi-object representation learning on videos. Multi-object representation learning on videos has typically been approached by extending image-based models to sequential data. A common strategy involves integrating a recurrent module such as LSTM [83] or GRU [28] to capture temporal dependencies. This paradigm was applied to both spatial-attention models (in a recurrent setup [104] or in single-shot form [32, 92]), as well as to scene-mixture models [33, 197], achieving promising results on synthetic datasets. However, without supervision, these methods often fail to scale to more complex, real-world scenarios, encountering limitations similar to their image-based counterparts. What is more, most existing approaches emphasise the reconstruction quality and scene segmentation capabilities of these models, rarely reviewing or comparing the quality of inferred object representations. In addition, the use of recurrent modules introduces additional computational overhead.

An alternative formulation was introduced in SIMONe [95], which uses a transformer-based architecture to decompose latent space into temporal and per-object latents explicitly. By inferring the entire sequence at once, the model can extract split representations referring to how the frame changes over time, and encapsulate each object's appearance in its per-sequence representation. However, this approach is constrained by its fully unsupervised training setup and significant computational demands, limiting its scalability to real-world datasets.

RDIR adopts a recurrent mechanism inspired by discovery-propagation approaches in spatial-attention video models. Unlike methods that explicitly track and associate new objects across frames, RDIR applies the recurrent cell to the encoder's feature maps, enabling temporal refinement of features through hidden state propagation. This facilitates implicit object association across time without additional tracking logic.

Supervision in multi-object representation learning models. Supervision is increasingly considered a key factor for enabling multi-object representation learning to scale to real-world datasets. In Slot Attention for Videos [103], self-supervision is introduced through optical flow conditioning, allowing the model to incorporate motion cues. A related extension [48], conditions the model on depth maps rather than RGB images, demonstrating improved scene understanding. While these approaches show improved segmentation performance, they do not review the expressiveness or consistency of learned representations. Furthermore, these methods rely on auxiliary signals such as depth maps or optical flow, which require annotations or online estimations during training.

RDIR builds on the semi-supervised framework introduced in Chapter 3, where a pre-trained single-shot object detector provides spatial attention locations. RDIR incorporates a staged training procedure designed to balance computational efficiency and training effectiveness. Assuming access to a reliable object detector, the model can accurately attend to object classes of interest and learn temporally consistent object representations from video data without requiring further supervision.

4.3 Recurrent Detect, Infer, Repeat

This section introduces RDIR (Recurrent Detect, Infer, Repeat), a structured autoencoder designed for multi-object representation learning in videos. RDIR extends the semi-supervised framework by incorporating a recurrent encoder network that enables implicit temporal reasoning over sequences of frames. The model combines pre-trained object detection with learned object-centric representation encoders, facilitating temporally consistent and semantically meaningful embeddings.

RDIR builds upon the concept of extending a one-stage object detection model with dedicated encoding heads for object-centric representation learning trained in an autoencoding framework. Each frame in a video sequence is processed to produce a structured representation consisting of a set of object-specific latent vectors. These vectors encode information about object appearance, location, presence and depth in the scene. The representations are then filtered and passed to the decoder, which produces per-object reconstructions, which are merged to create a full image reconstruction.

The overall model includes two core components: the encoder $q_\phi(\mathbf{z}|\mathbf{x})$, which maps the input video $\mathbf{x}$ to the latent representation $\mathbf{z}$, and the decoder $p_\theta(\mathbf{x}|\mathbf{z})$, which reconstructs the input video from the inferred latent representation. Both functions are trained jointly.

4.3.1 RDIR Latent Space

The latent space in RDIR is structured to encode information about individual objects present in the scene. Leveraging a one-stage object detection model and multi-scale feature maps, the architecture produces object-centric representations for multiple grids, referring to spatial locations in the input image. Each cell in all detection grids contributes a set of latent variables:

1. $\mathbf{z}^i_{\text{where}} = [x_i, y_i, w_i, h_i] \in \mathbb{R}^4$, which describes the i-th object's position and size (as bounding box centre coordinates x_i, y_i, width w_i and height h_i),
2. $\mathbf{z}^i_{\text{present}} = \max \mathbf{c} \in [0, 1]$, used to encode the object's presence confidence, computed as the maximum of object's class confidences $\mathbf{c}$,
3. $\mathbf{z}^i_{\text{what}} \in \mathbb{R}^D$, a D-dimensional appearance vector capturing the visual characteristics of the object, used during the reconstruction of per-object patches,
4. $\mathbf{z}^i_{\text{depth}} \in \mathbb{R}$, which represents the relative depth of the object, facilitating correct compositing order in the decoder.

Following the semi-supervised approach, RDIR uses a pre-trained object detection model to provide $\mathbf{z}_{\text{where}}$ and $\mathbf{z}_{\text{present}}$ latent variables, simplifying the process of object discovery. The remaining latent variables ($\mathbf{z}_{\text{what}}$ and $\mathbf{z}_{\text{depth}}$) are learned in the representation learning framework.

4.3.2 RDIR Encoder

The RDIR encoder (illustrated in Figure 4.1) is designed to capture latent representations that are both spatially and temporally consistent across video sequences. The architecture builds upon the CSPDarknet53 convolutional backbone introduced in YOLOv4 [12], offering substantially enhanced feature extraction capacity compared to SSDIR.

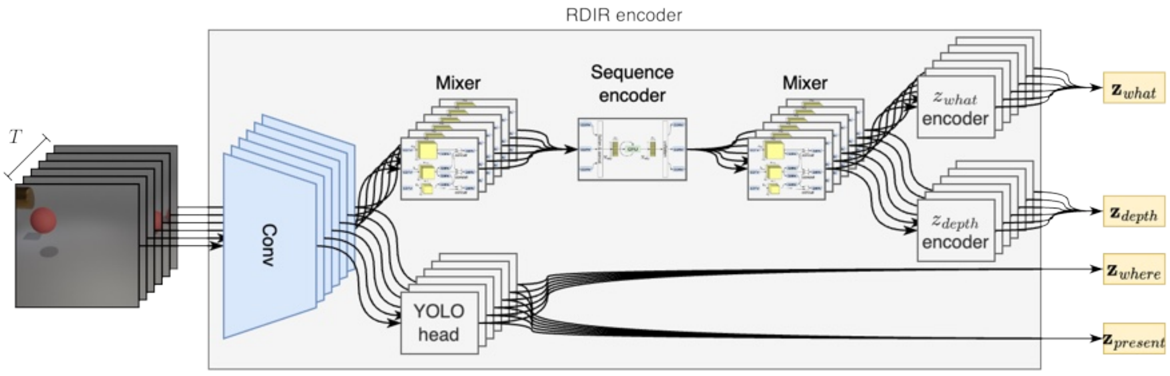


Figure 4.1: RDIR encoder architecture. Input videos, treated as sequences of frames, are processed individually by the convolutional backbone (*Conv*). The intermediate features are passed through a pre-trained YOLO head to infer $\mathbf{z}_{\text{where}}$ and $\mathbf{z}_{\text{present}}$ latents for each grid cell. To infer $\mathbf{z}_{\text{what}}$ and $\mathbf{z}_{\text{depth}}$ embeddings, these intermediate features are forwarded to the *Mixer* module (see Figure 4.3) to share information across all levels of grid resolution. A *Sequence encoder* (Figure 4.2) follows, propagating temporal context across the input sequence. Another *Mixer* module is provided to share temporal context across scales. Finally, the resulting latent features are used by dedicated encoders for $\mathbf{z}_{\text{what}}$ and $\mathbf{z}_{\text{depth}}$.

YOLOv4 is a well-established single-shot object detection framework. Its backbone encodes multi-scale feature maps that are interpreted as spatial grids of intermediate representations, used for predicting bounding box coordinates and class confidences at each cell via spatial grid-based attention. By default, this model utilises three levels of feature grids, each with different spatial resolutions and channel depths.

RDIR leverages this backbone to extract intermediate features for each time-step $t \in 1, ..., T$ of the input sequence. These features are processed through a pre-trained YOLO detection head, which infers objects' locations ($\mathbf{z}_{\text{where}}$) and class confidences (used to compute $\mathbf{z}_{\text{present}}$). The same multi-scale feature maps are reused to learn appearance embeddings $\mathbf{z}_{\text{what}}$ and depth cues $\mathbf{z}_{\text{depth}}$ for all grid cells. This design extends the approach introduced in SSDIR, with key architectural advances to improve the quality of object-centric representations. To stabilise training, predicted bounding boxes are adjusted to be square, preventing the spatial transformer from introducing excessive aspect ratio distortion.

Sequence encoder. The key contribution in RDIR is the design of the Sequence encoder (illustrated in Figure 4.2), which enables implicit modelling of temporal structure in videos. Unlike prior spatial-attention models that require explicit object discovery and propagation across frames, RDIR leverages grid-based inference to incorporate temporal dynamics directly in the latent feature space.

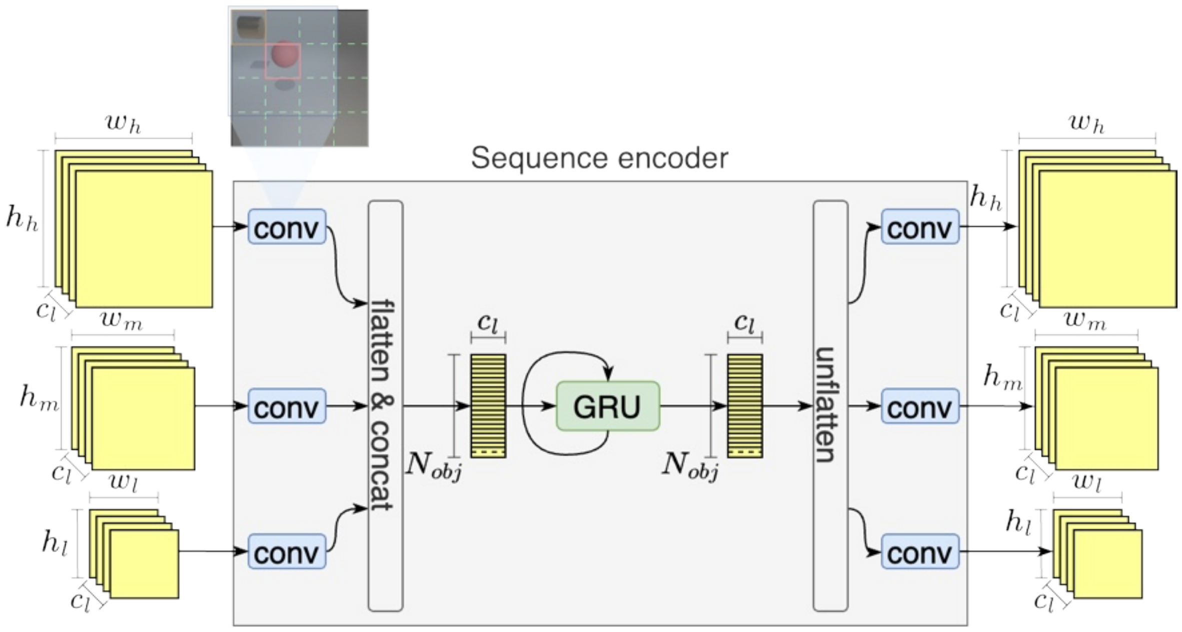


Figure 4.2: Sequence encoder module in RDIR. Multi-scale feature maps from each video frame are first processed by dimension-preserving convolutional layers (*conv*), which enable local spatial information to be propagated across adjacent cells using 2D kernels (of size 3×3). The resulting feature maps are flattened and concatenated into a tensor of shape $[N_{\text{obj}}, c_l]$ and passed through a GRU cell to propagate temporal context. Next, the original grid structure is restored via unflattening, and additional convolutional layers are applied to distribute temporal information across spatial regions. The output retains the same shape as the input.

The model flattens all grid cells from the multi-scale feature maps into a single tensor of shape $[N_{\text{objs}}, D]$ (where $N_{\text{objs}} = \sum_{i=1}^{3} w_i * h_i$ is the total number of cells across all three grid levels, and $D = c_l$ denotes the number of channels shared across feature maps). This flattened representation is passed through a recurrent cell for propagating temporal information through its hidden state. After recurrent processing, the original spatial grid structure is restored, preserving the original shape and order of cells.

This formulation allows the model to incorporate temporal consistency into the latent features of all cells before per-object representation learning, without the need for explicitly associating objects across frames. However, since each of the N_{objs} vectors is processed independently by the recurrent unit, temporal information cannot be shared between adjacent spatial regions, which is a limitation that may affect objects moving across the scene. To solve this issue, RDIR integrates dimension-preserving convolutional layers before and after the recurrent module. These layers enable spatial context to be shared among neighbouring cells, allowing information to flow across regions of image that correspond to the same object over time.

Mixer. An important part of the RDIR encoder is the Mixer module (illustrated in Figure 4.3). Multi-scale feature maps struggle with sharing contextual information across objects detected on various grid levels. This limitation arises because each grid is extracted from a different stage in the convolutional backbone, resulting in feature maps that lack shared lower-level representations. However, such shared context is particularly important for consistent and robust representation learning.

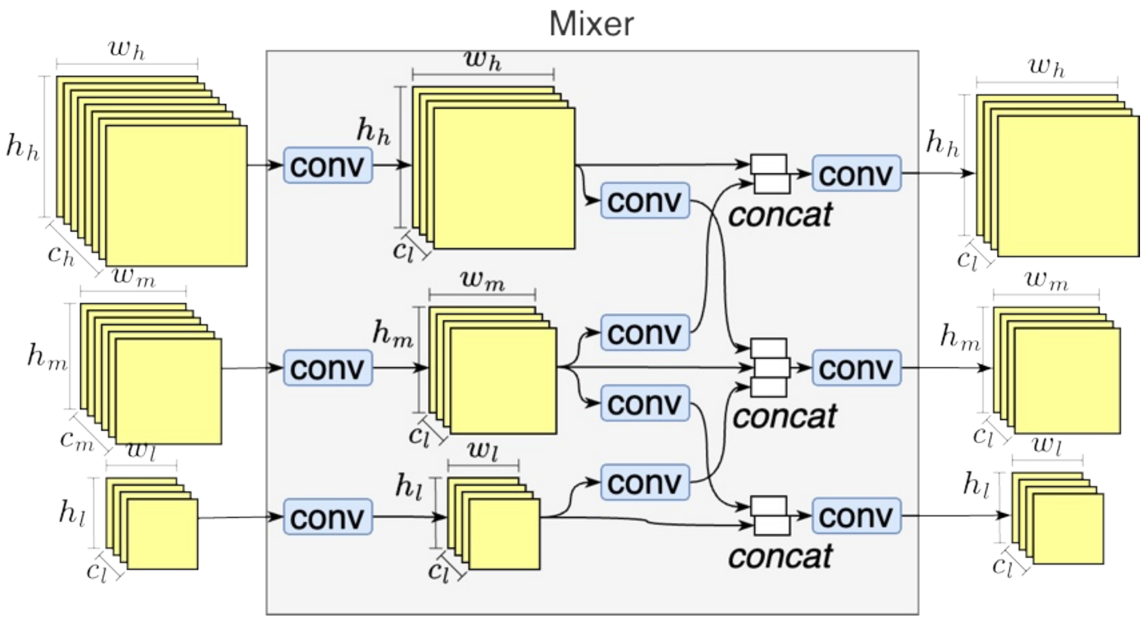


Figure 4.3: RDIR Mixer module used for context sharing across multi-level feature maps. The initial set of *conv* layers (each consisting of a 2D convolution, batch normalisation, and Leaky ReLU activation) unifies the number of channels in all feature maps to c_l. To enable information exchange across scales, additional *conv* layers perform upsampling and downsampling of neighbouring grids. The resulting feature maps are concatenated and passed through a final set of *conv* layers to restore the unified channel dimension c_l.

To address this, the Mixer module enables cross-scale propagation by performing upsampling and downsampling operations to align the dimensions of adjacent feature maps. These rescaled features are concatenated with the original grid-level features, allowing each grid cell to incorporate context from other resolutions while preserving the spatial ordering of cells. The design ensures that each cell can access relevant information from corresponding regions across feature levels.

Each Mixer module consists of a 2D convolutional layer followed by batch normalisation and a Leaky ReLU activation. The number and parameters of these convolutions are chosen to allow spatial alignment and concatenation of feature maps across grid levels. The output of the Mixer retains the original number of multi-scale grids (three in the YOLOv4 setup), with the smallest shared channel dimension across all levels.

Two Mixer modules are used in the RDIR encoder: one preceding the Sequence encoder and one following it. This approach allows the model to propagate information across feature scales both before and after the integration of temporal context via the GRU cell.

Latents encoding. The resulting temporally and contextually enriched feature maps are used to predict $\mathbf{z}_{\text{what}}$ and $\mathbf{z}_{\text{depth}}$. The $\mathbf{z}_{\text{what}}$ encoder applies a stack of 3×3 convolutional blocks, followed by a 1×1 convolution to produce latent maps with the target dimensionality ($\mathbf{z}_{\text{what}}$ size). The $\mathbf{z}_{\text{depth}}$ encoder consists of a single convolutional block and a final 1×1 convolution to output depth scores.

Optionally, the representation learning module of the encoder can use a cloned backbone and neck, allowing additional trainable capacity. In this case, the cloned neck is optimised with the encoders while sharing the frozen weights of the pre-trained backbone. The original YOLO detection head remains frozen throughout.

The overall encoding pipeline is summarised in Algorithm 2. The encoder network retains the ability to scale to images of varying resolutions, addressing a key limitation of SSDIR. Furthermore, the use of a more advanced convolutional backbone enables the model to operate effectively on complex datasets with higher-resolution inputs. As in SSDIR, the YOLO backbone and detection head are initialised with weights from a pre-trained object detection model and remain frozen during training, whereas the encoder part is trained together with the decoder without supervision, allowing the model to focus on semantically meaningful regions of the image during learning representations.

4.3.3 RDIR Decoder

The RDIR decoder follows the general structure of spatial-attention decoders used in prior multi-object representation learning models, particularly SSDIR. The key enhancements include support for non-max suppression (NMS), the use of batch normalisation and configurable object reconstruction size.

Algorithm 2: RDIR Encoder

```
   Input  : Image sequence $\mathbf{x}$
   Output : Latent representation for each image in sequence:
            $\{\mathbf{z}_{\text{where}}, \mathbf{z}_{\text{present}}, \mathbf{z}_{\text{what}}, \mathbf{z}_{\text{depth}}\}$
1  foreach $\mathbf{x}_i$ in $\mathbf{x}$ do
2      features $\leftarrow$ BackboneNeck($\mathbf{x}_i$)
3      [$\mathbf{z}_{\text{where}}, \mathbf{z}_{\text{present}}$] $\leftarrow$ YOLOHead(features)
4      $\text{features}_{\text{mixed}} \leftarrow$ Mixer(features)
5      Append $\mathbf{z}_{\text{where}}$, $\mathbf{z}_{\text{present}}$, $\text{features}_{\text{mixed}}$ to lists
6  end
7  $\text{features}_{\text{seq}} \leftarrow$ SeqEncoder($\text{features}_{\text{mixed}}$)
8  foreach $\mathit{features}^{(t)}_{\mathit{seq}}$ in $\mathit{features}_{\mathit{seq}}$ do
9      $\text{features}_{\text{seqmixed}} \leftarrow$ SeqMixer($\text{features}^{(t)}_{\text{seq}}$)
10     $\mathbf{z}_{\text{what}} \leftarrow$ WhatEncoder($\text{features}_{\text{seqmixed}}$)
11     $\mathbf{z}_{\text{depth}} \leftarrow$ DepthEncoder($\text{features}_{\text{seqmixed}}$)
12     Append $\mathbf{z}_{\text{what}}$, $\mathbf{z}_{\text{depth}}$ to lists
13 end
14 return $\{\mathbf{z}_{where}, \mathbf{z}_{present}, \mathbf{z}_{what}, \mathbf{z}_{depth}\}$
```

The decoder operates independently on each frame and does not leverage temporal information. During training, latent representations are filtered using stochastic sampling based on $\mathbf{z}_{\text{present}}$. This stochasticity was found to improve learning efficiency. To improve training stability and ensure the model learns a broader distribution, negative examples are included in addition to those inferred by the detector. At inference time, sampling of $\mathbf{z}_{\text{present}}$ is replaced with a fixed thresholding operation, and negative examples are omitted. Non-maximum suppression (Algorithm 1) is applied to the $\mathbf{z}_{\text{present}}$ scores with an Intersection over Union threshold of 0.45 to eliminate redundant detections and reduce duplication.

Then, the selected appearance latents $\mathbf{z}_{\text{what}}$ are decoded using a convolutional decoder network consisting of a sequence of 2D transposed convolution layers to produce object reconstructions. Each layer is followed by batch normalisation and a Leaky ReLU activation function. The architecture is parameterised by the input latent dimensionality, the number of decoder channels and the target resolution of the per-object reconstructions. The final transposed convolution layer produces three output channels followed by a sigmoid activation, resulting in individual RGB object patch reconstructions.

Each object patch is then spatially transformed to the original image size based on the corresponding $\mathbf{z}_{\text{where}}$ latent using a spatial transformer [87]. The compositing is performed by depth order, determined by sorting the objects based on their $\mathbf{z}_{\text{depth}}$ values. This allows the model to produce a more accurate and compact set of object representations, particularly in cluttered scenes containing many objects. The resulting image contains only the reconstructed foreground objects; no background modelling is performed.

The high-level flow of the decoder is presented in Algorithm 3.

Algorithm 3: RDIR Decoder

```
Input  : Latent representation for each image in sequence:
         $\{\mathbf{z}_{\text{where}}, \mathbf{z}_{\text{present}}, \mathbf{z}_{\text{what}}, \mathbf{z}_{\text{depth}}\}$, sorted by $\mathbf{z}_{\text{depth}}$
Output : Image sequence reconstruction $\hat{\mathbf{x}}$
1  foreach $\{\mathbf{z}_{where}, \mathbf{z}_{present}, \mathbf{z}_{what}, \mathbf{z}_{depth}\}$ in $\mathbf{z}$ do
2      $\mathbf{z}_{\text{what}} \leftarrow \text{Filter}(\mathbf{z}_{\text{what}}, \mathbf{z}_{\text{present}})$
3      $\mathbf{z}_{\text{where}} \leftarrow \text{Filter}(\mathbf{z}_{\text{where}}, \mathbf{z}_{\text{present}})$
4      $\mathbf{z}_{\text{depth}} \leftarrow \text{Filter}(\mathbf{z}_{\text{depth}}, \mathbf{z}_{\text{present}})$
5      foreach $(\mathbf{z}^i_{what}, \mathbf{z}^i_{where})$ in $(\mathbf{z}_{what}, \mathbf{z}_{where})$ do
6          $\mathbf{o}^i_{\text{dec}} \leftarrow \text{Decoder}(\mathbf{z}^i_{\text{what}})$
7          $\mathbf{o}^i_{\text{transformed}} \leftarrow \text{STN}(\mathbf{o}^i_{\text{dec}}, \mathbf{z}^i_{\text{where}})$
8          $\hat{\mathbf{x}} \leftarrow \text{Merge}(\mathbf{o}^i_{\text{transformed}}, \mathbf{z}^i_{\text{depth}})$
9      end
10 end
11 return $\hat{\mathbf{x}}$
```

4.3.4 Model Training

RDIR is trained in a classic autoencoder setup. Since the probabilistic properties of variational autoencoders are not utilised explicitly in this research, training is performed using the mean squared error (MSE) loss function, minimising the reconstruction error between the input sequence of images $\mathbf{x}$ and their reconstruction $\hat{\mathbf{x}}$ (Equation (1.9)). Although this formulation does not use probabilistic components, the model could be trivially extended to a VAE framework. In the context of generative modelling, the training objective could be viewed as maximising the likelihood of the data under the assumed model, given the noise introduced by the grid-based encoding and the injection of negative samples during training [57]. Since RDIR does not explicitly reconstruct the background, the reconstruction error measured over the entire image may be inflated due to the complexity of the background. To mitigate this, an additional loss term is introduced computing the reconstruction error only within regions corresponding to detected objects. The final loss function is defined in Equation (4.1):

$$\mathcal{L} = \lambda_r \cdot \text{MSE}\left(\mathbf{x}, \hat{\mathbf{x}}\right) + \lambda_{\text{obj}} \frac{1}{K} \sum_{i}^{K} \text{MSE}\left(\mathbf{x}^i_{\text{obj}}, \hat{\mathbf{x}}^i_{\text{obj}}\right) \tag{4.1}$$

where:

λ_r is the reconstruction MSE component coefficient,
$\mathbf{x}$ is the input image sequence,
$\hat{\mathbf{x}}$ denotes the reconstruction,
λ_{obj} is the per-object MSE component coefficient,
K is the number of objects in all images in the sequence,
$\mathbf{x}^i_{\text{obj}}, \hat{\mathbf{x}}^i_{\text{obj}}$ refer to detected objects' original appearances and reconstructions.

A staged training procedure is used to train the model effectively. First, the weights of a pre-trained object detection model are transferred to an SSDIR-like architecture, identical to RDIR's, but without the Sequence encoder and the second Mixer. This model is trained in an unsupervised autoencoding setup that initialises components responsible for multi-object representations in images, in particular, the $\mathbf{z}_{\text{what}}$ encoder and decoder. In the next stage, the model is extended with the Sequence encoder and the second Mixer, and a full RDIR model is trained as an unsupervised autoencoder on videos. This results in faster convergence and improved final performance.

4.4 Experiments

This section presents the experimental setup and evaluation methodology used to assess the performance of RDIR. The goal was to evaluate the quality and consistency of the object representations learned by the model in comparison with reference multi-object representation learning approaches. The evaluation includes downstream task performance and visual review of the image and object reconstructions. In addition, an ablation study was conducted to examine the role of the Mixer module in the encoder and the model's robustness to varying numbers of objects in the sequence.

Baseline methods To evaluate the performance of RDIR, a set of representative baseline methods was selected. The goal was to include one method from each major category of multi-object representation learning approaches for videos. The selected baselines span spatial-attention models, scene-mixture models, and transformer-based architectures:

- **SCALOR** [92], a recurrent spatial-attention model that performs object discovery and propagation using a proposal-rejection mechanism across frames,
- **PROVIDE** [197], a recurrent scene-mixture model, that runs iterative amortised inference using a 2D-LSTM, integrating context from previous refinement steps and earlier frames,

- **SIMONe** [95], a transformer-based variational autoencoder that factorises latent representations into temporally invariant per-object components and global sequence-level context,
- **SSDIR-YOLO** [208], an enhanced version of SSDIR that incorporates the YOLOv4 backbone (same as in RDIR) and the Mixer module; used to isolate the contribution of the Sequence encoder to the model, and overcoming the major limitations of the original model, such as low-resolution processing and weak convolutional backbone.

All models were configured to use the same dimensionality of latent representation. For experiments on the Multi-Scale Moving MNIST dataset, the original resolution is 64×64; however, RDIR and SSDIR-YOLO operated on upscaled inputs (128×128) due to their grid resolution dependencies.

In the MOT15 object tracking experiments, each model was evaluated at the highest input resolution supported by its architecture and GPU constraints: SCALOR and PROVIDE used their default resolution of 64×64, SIMONe was run at 128×128, while SSDIR-YOLO and RDIR operated at 416×416, demonstrating their scalability to higher-resolution inputs.

Datasets The experiments were conducted using a combination of synthetic and real-world datasets. The inclusion of simulated datasets allowed for precise control over various scene parameters, which is crucial for evaluating the robustness and generalisation capability of the model. Figure 4.4 presents sample sequences from each dataset.

1. **Multi-Scale Moving MNIST**[4]. This synthetic dataset extends the concept of multi-scale digit sequences introduced in Chapter 3, providing a structured benchmark for assessing the model's performance in the presence of size variation, object motion and occlusion. Each sequence begins with a frame of size 128×128 pixels, containing a random number of MNIST digits (the number sampled uniformly between $n_{\min}$ and $n_{\max}$). Digits are independently scaled to one of the predefined sizes $\{s_1, s_2, \dots, s_N\}$ and placed at random positions.

   Throughout the T-frame sequence, each digit moves with an initial velocity (v_x, v_y) sampled uniformly from the range $(-1, 1)$ in both directions. Digits bounce off the image boundaries with symmetric reflection. Additionally, digits undergo

[4]https://github.com/piotlinski/moving_multiscalemnist

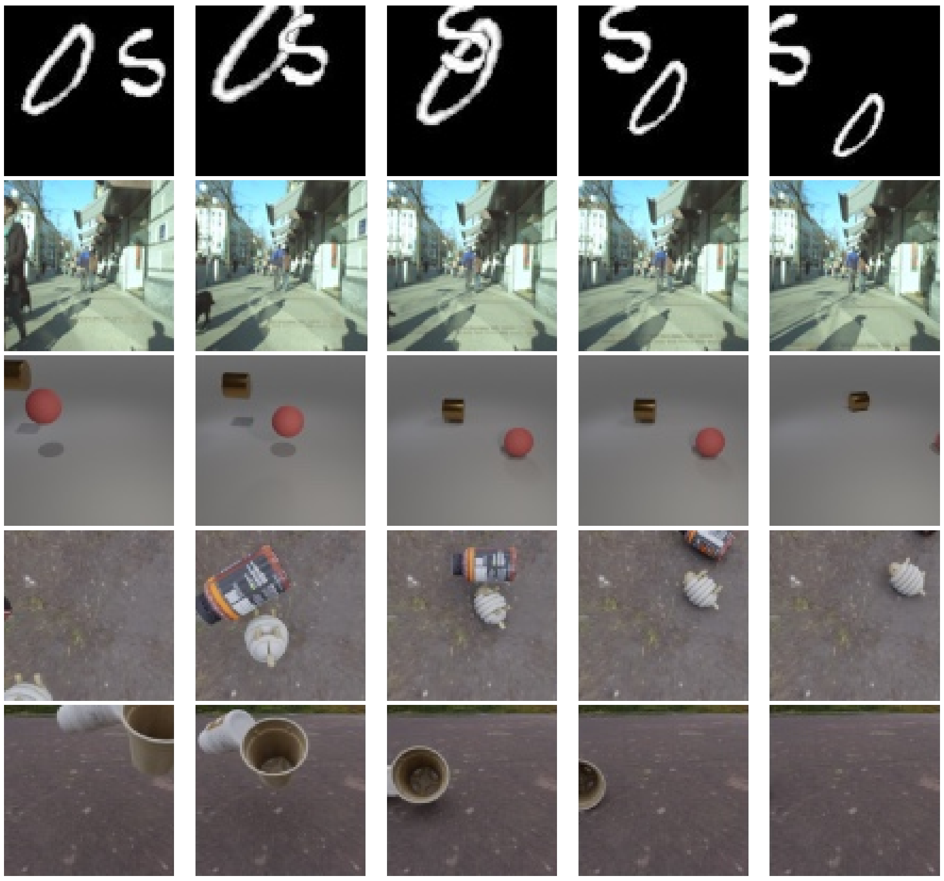


Figure 4.4: Example sequences from datasets used in the experiments. Each row shows a sequence of five frames from datasets: (1) Multi-Scale Moving MNIST, (2) MOT15, (3) MOVi-A, (4) MOVi-C, (5) MOVi-E.

sinusoidal size oscillation with amplitude sv (e.g., sv = 0.1 corresponds to 10% size variation) and period op (where op = 1.0 corresponds to one full oscillation per second).

The primary training dataset uses the configuration summarised in Table 4.1. For evaluation, two additional variants were created:

- *No scaling*: digits remain at a fixed size (sv = 0), allowing for evaluation without scale changes,
- *No translation*: digit velocities are set to zero, removing motion from the sequences.

For the ablation study, further variants involved modifying the number of objects per sequence (ranging from 1 to 7 digits per frame), which allowed evaluation of the model's robustness to varying object counts and occlusion levels. Each variant contained 1000 sequences.

parameter	value
train sequences	8000
test sequences	2000
T	10
n_{min}	2
n_{max}	5
s	$\{48, 72, 96\}$
op	$\{1.0, 1.5, 2.0\}$
sv	$\{0.0, 0.1, 0.2, 0.3\}$
FPS	10

Table 4.1: Parameters used for generating the Multi-Scale Moving MNIST dataset.

2. **MOT15** [111]. This dataset served as a real-world benchmark for the task of multi-object tracking, enabling the evaluation of object representations in a more complex context. The training split includes 11 video sequences containing multiple walking pedestrians, captured in urban environments. Provided annotations include bounding boxes and consistent object IDs. Since the official test set does not include public annotations, a validation subset was created from the training data according to Table 4.2.

	sequence	resolution	frames
train	ADL-Rundle-8	1920×1080	654
	ETH-Bahnhof	640×480	1000
	ETH-Pedcross2	640×480	837
	KITTI-13	1242×375	340
	PETS09-S2L1	768×576	795
	TUD-Campus	640×480	71
	TUD-Stadtmitte	640×480	179
	Venice-2	1920×1080	600
			$\mathbf{\Sigma = 4467}$
val	ADL-Rundle-6	1920×1080	525
	ETH-Sunnyday	640×480	354
	KITTI-17	1224×370	145
			$\mathbf{\Sigma = 1024}$

Table 4.2: MOT15 dataset split details.

For model training, each video was divided into overlapping sequences of 10 frames. Full-length sequences were used during the evaluation. The size of the frames was adjusted to the maximum resolution supported by each model configuration.

3. **COCO** [119]. It is a large-scale benchmark widely used in object detection tasks. It contains images depicting everyday scenes with 80 object categories, including people, animals, vehicles and household items. In this research, COCO was used

to train the initial object detection model, focusing solely on the *person* class. The resulting model was initialising the detection components of the representation learning model on MOT15, ensuring proper detection of pedestrians.

4. **MOVi** [64]. It is a video benchmark designed for evaluating object-centric models in visually complex, synthetic environments. In this research, these datasets were used to qualitatively assess the stability of reconstructions produced by RDIR. Three subsets: MOVI-A, MOVi-C and MOVi-E were selected based on their progressively increasing difficulty. Each one consists of sequences containing 24 images of resolution 256×256, with detailed annotations including object masks, bounding boxes and identities. Dataset statistics are summarised in Table 4.3.

	MOVi-A	**MOVi-C**	**MOVi-E**
train sequences	9703	9737	9749
test sequences	250	250	250

Table 4.3: Number of sequences in each MOVi dataset.

Model training setup RDIR[5] was trained using a staged protocol on a single NVIDIA A40 GPU:

1. An object detection model was first trained on an annotated dataset with supervision. YOLOv4 was used as the base detector, and its backbone and prediction head were later reused in the encoder of RDIR.

2. In the second stage, the model was trained to learn object-centric representations on static images in an unsupervised autoencoding setup. Here, the Sequence encoder and the second Mixer module were removed while the model operated as a single-frame autoencoder. The YOLO backbone and detection head were kept frozen, while the representation learning modules were updated. The checkpoint achieving the lowest validation loss was selected for the next phase. This intermediate stage was found to improve final model performance and reduce overall training time compared to directly training the full recurrent model.

3. Finally, the pre-trained image-based model was extended with the Sequence encoder and the second Mixer module to form the complete RDIR architecture. Training began with a short warmup phase, during which only the newly added modules were updated while the rest of the model remained frozen. This allowed

[5] `https://github.com/piotlinski/rdir`

the recurrent components to initialise effectively. Afterwards, training proceeded in the same manner as the second stage, with the backbone and YOLO heads remaining frozen and sequence-level loss used as the objective.

Model hyperparameters used in the experiments are summarised in Table 4.4.

parameter	**MNIST**	**MOT15**	**MOVi-A**	**MOVi-C**	**MOVi-E**
batch size	256	4	64	64	64
image size	128	416	256	256	256
decoded size	32	32	32	32	32
decoder channels	16	128	256	256	256
z_{what} size	64	128	256	256	256
z_{what} hidden	3	5	5	5	5
λ_{total}	5	5	5	5	5
λ_{obj}	10	10	10	10	10
RNN type	BiGRU	GRU	BiGRU	BiGRU	BiGRU
RNN cells	2	2	2	2	2
seq CNN hidden	2	2	2	2	2
seq CNN kernel size	5	5	5	5	5

Table 4.4: Model parameters used for learning representations on each dataset.

4.4.1 Experiment 1: Predicting the Sum of Digits in a Sequence

To assess the informativeness of the learned representations, a downstream task was conducted in which the objective was to predict the sum of digits in a video sequence, following the approach proposed in [53]. Each model was trained on the full Multi-Scale Moving MNIST dataset, after which latent representations were extracted for both the train and validation subsets. A linear regression model was then fitted on the representations to predict the sum of digits in each sequence, using ground truth labels to compute the target.

Since each method produces multiple latent vectors for each image (corresponding to detected objects or inferred attention masks), these were aggregated across each frame by summation. The final sequence-level feature was computed as the average of the aggregated per-frame vectors, yielding a 64-dimensional representation per sequence. This strategy enables a uniform comparison across models, including those that do not explicitly produce per-object representations. For instance, scene-mixture-based models (such as PROVIDE and SIMONe) often generate masks covering multiple objects, while spatial attention models with fixed object sizes (such as SCALOR) tend to fragment large objects. Aggregating over sequence mitigates these differences and enables comparison based on overall scene encoding quality.

Performance was measured using the R^2 metric, computed across three random seeds for both training and validation subsets. The results are reported in Table 4.5.

dataset variant	**model**	**train R^2**	**val R^2**
MNIST	SCALOR	0.301 ± 0.009	0.294 ± 0.028
	PROVIDE	0.298 ± 0.258	0.282 ± 0.258
	SIMONe	0.580 ± 0.020	0.577 ± 0.023
	SSDIR-YOLO	0.574 ± 0.015	0.573 ± 0.012
	RDIR	0.579 ± 0.010	0.581 ± 0.002
MNIST (no scaling)	SCALOR	0.353 ± 0.018	0.357 ± 0.024
	PROVIDE	0.349 ± 0.300	0.345 ± 0.312
	SIMONe	0.652 ± 0.013	0.658 ± 0.070
	SSDIR-YOLO	0.641 ± 0.013	0.647 ± 0.015
	RDIR	0.625 ± 0.006	0.636 ± 0.002
MNIST (no translation)	SCALOR	0.288 ± 0.017	0.274 ± 0.010
	PROVIDE	0.266 ± 0.232	0.253 ± 0.236
	SIMONe	0.493 ± 0.018	0.478 ± 0.019
	SSDIR-YOLO	0.396 ± 0.011	0.404 ± 0.019
	RDIR	0.425 ± 0.009	0.429 ± 0.008

Table 4.5: Downstream task: regression of the sum of digits in a sequence. RDIR achieves best results on the most complex dataset but is slightly worse than SIMONe and SSDIR in simpler datasets (without scaling or translation). Values are averaged over 3 random seeds.

RDIR achieves the highest performance on the full dataset, which includes scaling and translation, demonstrating the effectiveness of incorporating the Sequence encoder into the SSDIR baseline. The results of SIMONe are competitive with those of RDIR, highlighting the advantages of its factorised latent space for capturing global sequence-level information.

On the simplified variants of the dataset (without scaling or translation), RDIR performs slightly below SIMONe and SSDIR. Importantly, the inclusion of temporal modelling via recurrent units offers a practical advantage over SSDIR, particularly on most complex data, and when stable object representations across time are required. Unlike transformer-based models, RDIR's use of recurrent cells supports online inference, eliminating the need to observe the full sequence before processing.

The performance of SCALOR and PROVIDE is notably worse across all dataset variants. Both models exhibited difficulties in correctly identifying objects in the scene, often producing an excessive number of latent vectors per frame. These outcomes suggest limitations in their capacity to handle dynamic, cluttered environments and extract coherent object representations under objects' scale and motion variability.

4.4.2 Experiment 2: Representations-Based Object Tracking

To explore the utility of RDIR representations in real-life scenarios, their performance was evaluated in the task of object tracking. The experimental setup is as follows: each model was trained on the training subset of the MOT15 dataset. Next, representations were extracted from the validation dataset. A simple object tracking algorithm was used, matching objects between consecutive frames via the Hungarian algorithm based on the cosine similarity between object representations, while maintaining unique IDs for tracked entities. Tracking was performed using each model's bounding box predictions combined with assigned object IDs, and evaluated using the TrackEval framework [126].

Tracking performance was quantified using the HOTA metric [127] (Equation (1.25)), averaged over three random seeds on both training and validation subsets. Results for all models, along with a baseline object detector (*YOLO*), are reported in Table 4.6.

	MOT15 HOTA	
	train	**val**
SCALOR	1.452 ± 0.063	1.305 ± 0.041
PROVIDE	0.753 ± 0.038	0.698 ± 0.036
SIMONe	0.495 ± 0.275	0.724 ± 0.597
SSDIR-YOLO	31.774 ± 2.193	20.752 ± 1.190
RDIR	30.582 ± 2.201	20.749 ± 0.344
YOLO	20.100 ± 0.520	14.263 ± 0.525

Table 4.6: Downstream task: object tracking using learned representations. RDIR and SSDIR-YOLO achieve substantially better results than the baseline methods. Values are averaged over 3 random seeds.

SSDIR-YOLO and RDIR achieve comparable performance, both showing a substantial improvement over the baseline *YOLO* detection model. For PROVIDE and SIMONe, which produce segmentation masks rather than bounding boxes, an additional step was required to estimate object locations. Each mask was converted to a bounding box by computing its centre of mass and assigning a fixed $3:1$ height-to-width ratio, appropriate for the pedestrian class. Despite this adjustment, both PROVIDE and

SIMONe fail to consistently localise individual objects, instead segmenting the scene into regions inconsistent with pedestrians. This leads to low tracking accuracy, as reflected in the HOTA metric.

Leveraging a pre-trained object detection model enables the transfer of knowledge from external datasets. Table 4.7 presents the performance of models built on a detector pre-trained on the COCO dataset (restricted to the 'person' class). While the baseline detector (*YOLO*@COCO) performs poorly on this task due to limited domain alignment, incorporating representations learned by SSDIR-YOLO and RDIR leads to a substantial improvement in object tracking accuracy. In contrast, a model trained end-to-end, without the benefit of a frozen pre-trained object detection, exhibits performance similar to baseline methods, failing to reliably focus on relevant objects.

	MOT15 HOTA	
	train	**val**
YOLO@MOT15	20.100 ± 0.520	14.263 ± 0.525
YOLO@COCO	8.240 ± 0.154	9.457 ± 0.141
SSDIR@MOT15	31.774 ± 2.193	20.752 ± 1.190
SSDIR@COCO	14.788 ± 0.607	14.029 ± 0.751
RDIR@MOT15	30.582 ± 2.201	20.749 ± 0.344
RDIR@COCO	13.922 ± 0.781	13.752 ± 0.777
RDIR E2E	0.634 ± 0.076	0.582 ± 0.120

Table 4.7: Downstream task: object tracking using learned representations. '@' denotes the dataset on which the base detection model was trained. The addition of representations improves tracking performance, even for less-fitting object detection models. Values are averaged over 3 random seeds.

The marginal difference in performance between SSDIR-YOLO and RDIR requires attention. The SSDIR baseline used here incorporates architectural improvements proposed in this work, which enhance its capabilities beyond the original version. However, RDIR is able to learn from video sequences and incorporate temporal context into object representations, leading to more general embeddings, which is particularly valuable when objects become partially occluded or ambiguous in single frames. Due to the use of recurrent cells, RDIR could scale better to large unlabelled video datasets.

4.4.3 Ablation Study 1: Influence of the Mixer Module on Model Performance

To evaluate the impact of the Mixer module in RDIR, an ablation study was conducted by training three architectural variants. The standard model is referred to as *RDIR*. In *no-mixer*, the latent features were left unmodified, requiring separate recurrent cells for each feature map due to differing channel dimensions. The *downscaler* variant unifies the channel dimensions to enable a single shared recurrent cell but omits feature-level mixing. All models were evaluated on the digit summation task (Subsection 4.4.1), with R^2 metrics summarised in Table 4.8.

	MNIST	
R^2	**train**	**val**
RDIR	0.579 ± 0.010	0.581 ± 0.002
no-mixer	0.561 ± 0.023	0.562 ± 0.022
downscaler	0.543 ± 0.027	0.542 ± 0.027

Table 4.8: Ablation study: the influence of the Mixer module on the quality of representations. Adding the Mixer to RDIR improves the performance of the linear regression model applied to its representations. Values are averaged over 3 random seeds.

The results demonstrate that incorporating the Mixer module into RDIR leads to improved representation quality, as evidenced by higher R^2 scores. Interestingly, simply reducing the number of channels to enable a single recurrent cell (*downscaler*) does not yield better performance than using multiple recurrent cells without mixing (*no-mixer*). This suggests that the key advantage of the Mixer module lies not in unifying feature dimensions alone, but in allowing feature maps at different scales to share contextual information prior to sequence modelling. Without this integration, the model cannot effectively leverage complementary information from other grid levels.

4.4.4 Ablation Study 2: Influence of the Number of Objects in the Sequence on Model Performance

To evaluate RDIR's robustness to the number of objects visible in a scene, additional versions of the Multi-Scale Moving MNIST dataset were generated, containing between 1 and 7 digits per sequence. Each dataset was split into training and validation subsets (80 : 20 ratio). Following the same digit summation evaluation protocol as in Subsection 4.4.1, linear regression models were trained on the extracted representations. The R^2 scores and corresponding standard deviations are presented in Figure 4.5.

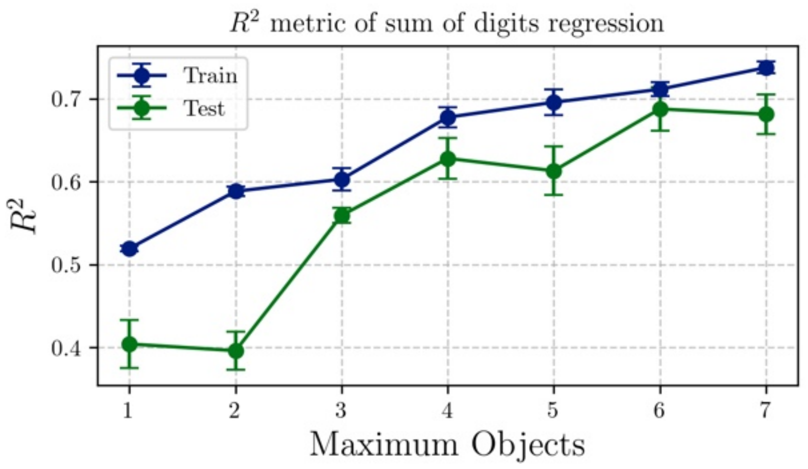


Figure 4.5: Impact of the number of objects per sequence on representation quality in RDIR. Each point shows the R^2 score of a linear regression model predicting the sum of digits for all variants of the dataset (with a given number of objects). Performance improves as the number of objects increases, indicating RDIR's ability to generalise to more complex scenes. Error bars denote the standard deviation over three random seeds.

The results confirm that RDIR produces representations robust to an increasing number of objects in the scene. As the number of objects increases, the downstream model maintains or improves its performance, demonstrating RDIR's capacity to encode scene information in cluttered settings. Notably, a decline in performance is observed in sequences containing only one or two objects.

4.4.5 Computational Expense

This section presents a comparative analysis of the computational requirements of RDIR relative to the baseline methods. The computational expense of deep learning models depends on several factors, including hardware configuration, software stack, model architecture and dataset characteristics. To provide a comprehensive view, Table 4.9 reports each model's training time to convergence, while Table 4.10 presents each model's iteration speed.

The baseline methods were trained on different hardware than the proposed approach due to computational constraints. The batch size in all cases was tuned to utilise the memory capacity of a single GPU fully. Despite these differences, the results demonstrate that the staged training strategy adopted by RDIR leads to substantially lower computational costs compared to the reference methods.

Model	**Training time (10^3 s)**	**Batch size**	**GPU**
SCALOR	435.54 ± 3.92	16	RTX TITAN
PROVIDE	523.11 ± 3.22	4	RTX TITAN
SIMONe	238.71 ± 15.17	16	RTX TITAN
SSDIR-YOLO	9.34 ± 0.68	256	A40
RDIR	22.34 ± 1.36	32	A40

Table 4.9: Model training expense comparison. Although SSDIR-YOLO and RDIR were trained on a more powerful GPU, the large gap in training time highlights the substantially greater computational efficiency of these architectures.

	iterations per second
SCALOR	4.899
PROVIDE	2.758
SIMONe	8.938
SSDIR-YOLO	31.277
RDIR	27.645

Table 4.10: Model inference speed. SSDIR-YOLO and RDIR are much faster than other methods, despite using a higher input resolution.

The inference speed benchmark was conducted on a consistent hardware setup using an NVIDIA A40 GPU. Each model was tested on 1000 randomly selected sequences, preloaded to exclude data loading time. All models were evaluated with a batch size of 1. The input resolution was set to 64×64, except for SSDIR-YOLO and RDIR, which operated on upscaled 128×128 inputs, following the protocol established in earlier experiments. The results indicate a faster inference speed for SSDIR-YOLO and RDIR compared to baseline methods. This performance gain is primarily due to the heavy reliance on recurrent modules and iterative inference in baseline approaches, whereas RDIR was designed for efficient, fully parallel inference throughout its architecture.

4.4.6 Inference Visualisations

This section presents qualitative results illustrating the behaviour of the RDIR model. Each visualisation includes the input image (input), the model's full reconstruction (reconstruction), reconstruction overlaid with attention indicators (attention - bounding boxes), and a selection of individual object-level reconstructions (objects) showing how the model decomposes and interprets the scene. Visualisations are provided for several timesteps in the sequence ($T = \{1, 4, 7, 10\}$), using two example sequences from each dataset.

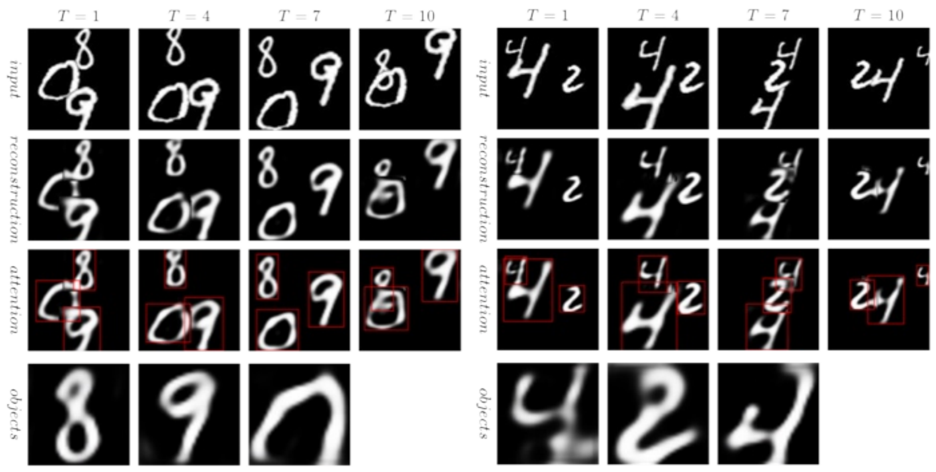


Figure 4.6: RDIR inference on Multi-Scale Moving MNIST dataset.

Multi-Scale Moving MNIST Figure 4.6 presents inference results on the Multi-Scale Moving MNIST dataset. The pre-training phase enables RDIR to reliably attend to individual digits throughout the sequence. Incorporating sequential context further improves RDIR performance in cases of overlap between objects.

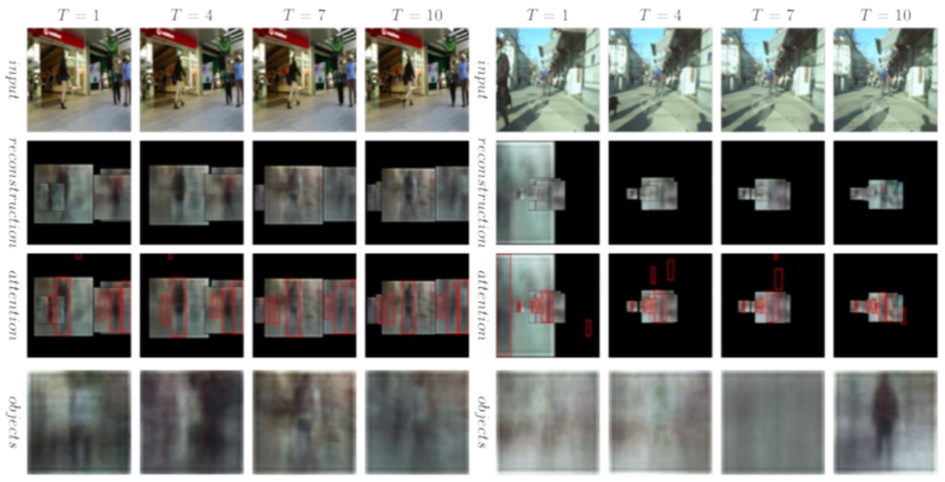


Figure 4.7: RDIR inference on MOT15 dataset.

MOT15 Figure 4.7 shows RDIR inference on the MOT15 dataset. As expected, RDIR does not reconstruct the entire scene but instead composes the output by pasting individual per-object reconstructions. This results in a lower overall image fidelity but highlights the model's ability to isolate and reconstruct human-like silhouettes. Despite the limited resolution, these object reconstructions preserve coarse appearance details, demonstrating the model's capacity to encode meaningful per-object representations.

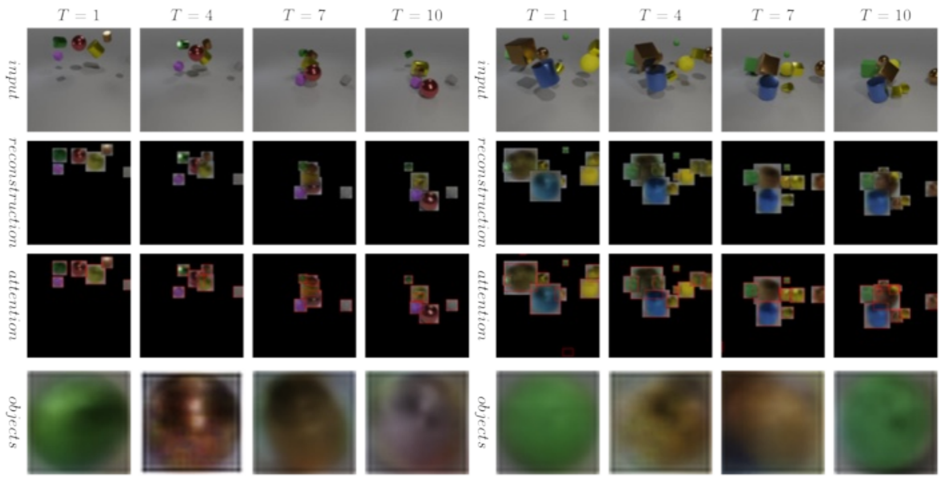


Figure 4.8: RDIR inference on MOVi-A dataset.

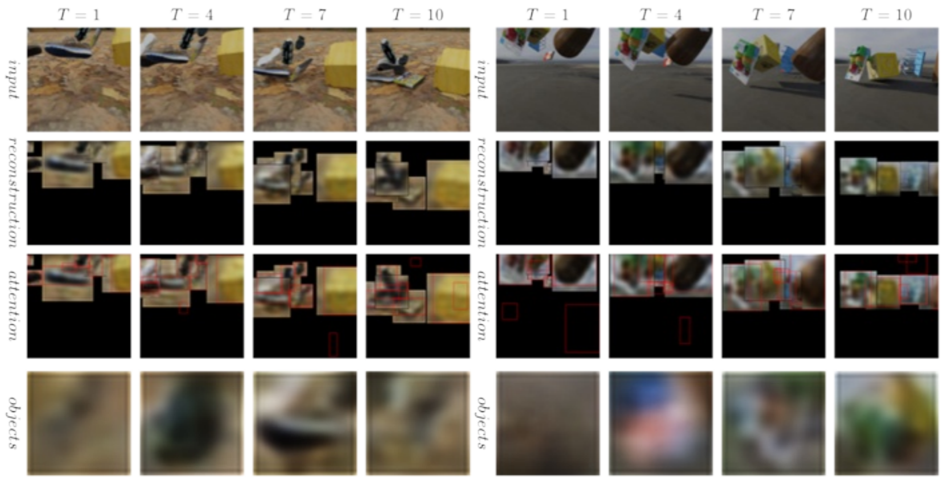


Figure 4.9: RDIR inference on MOVi-C dataset.

MOVi datasets Figures 4.8, 4.9, 4.10 show RDIR inference results on the MOVi datasets. The staged training strategy enables the model to attend accurately to individual objects, even in more complex scenes. However, the increased visual complexity leads to occasional false positives. These are effectively suppressed in the final image reconstruction, as seen in the *reconstructions* row. RDIR performs best on the simpler MOVi-A dataset, which contains basic geometric objects. The quality of individual object reconstructions remains limited due to the low-resolution format prior to spatial transformation. While increasing the resolution of intermediate object images could improve reconstruction fidelity, it would also substantially raise computational cost. Despite these constraints, the final reconstructions preserve the structure of the input in regions corresponding to detected objects.

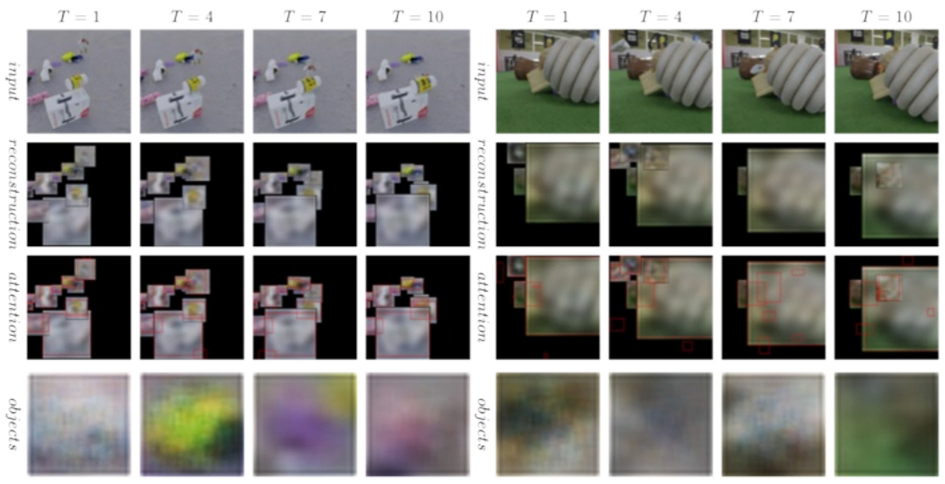


Figure 4.10: RDIR inference on MOVi-E dataset.

4.5 Summary

This chapter introduced RDIR, a recurrent extension of a semi-supervised object-centric model for learning stable, multi-object representations in video sequences. Building on a pre-trained object detection backbone, RDIR enables the extraction of spatially grounded and temporally consistent embeddings, which can be effectively applied in downstream tasks. The introduction of the Mixer module improves information sharing across multi-scale feature maps, allowing representations at each spatial resolution to benefit from contextual cues present in other grid levels. This cross-resolution integration enhances the quality and robustness of object representations. Additionally, the Sequence encoder enables the model to propagate information across time, using features from previous frames to improve the encoding of each object's latent representation. The method was shown to perform across various datasets, demonstrating robustness to varying numbers of objects and enabling knowledge transfer from models pre-trained on complex datasets. Besides representation quality, experimental results confirm RDIR's efficiency in training and inference.

Despite its advantages, RDIR inherits a key limitation from spatial grid-based attention models, in particular, its predecessor SSDIR: its reconstructions are based solely on per-object representations, without explicitly modelling the background, utilising a structured convolutional decoding methodology. While simplifying the learning task and enforcing object disentanglement, this leads to incomplete scene reconstructions and low generative capacity of the model. Improving reconstruction quality is the topic of the subsequent chapter of this thesis. Additional research could also investigate more expressive modelling of object interactions or focus on extending the architecture to support instance segmentation-based attention, enhancing the precision and interpretability of representations in complex real-world scenes.

Chapter 5

Detection Guidance for Conditioning Latent Diffusion Model in Multi-Object Representation Learning Setup

This chapter presents the stage of the PhD research dedicated to advancing multi-object representation learning on images through improved generative capabilities of the model. The key contribution of this work is the proposal of Detection-Guided Latent Diffusion, which follows the approach outlined in previous phases and integrates a pre-trained object detection model into a latent diffusion-based generative framework, where the object representations are used as conditioning for the denoising process. By leveraging spatial priors deriving from a detection model and incorporating them in the diffusion-based model, DetDiff is capable of focusing on semantically meaningful regions in the scene. The method integrates detection guidance, which enhances object representations with location information via positional encoding, and enforces focus on objects during training through localised object-aware loss. The model is trained as a latent diffusion model (LDM), yielding improvements in reconstruction fidelity compared to prior autoencoder-based approaches.

This work stems from unresolved challenges encountered in earlier research phases, particularly the poor generative capabilities of rendering-based decoders used in SSDIR and RDIR. At the time of this study, reference methods were selected to reflect the latest advancements in the area of object-centric learning, including other LDM-based models, advanced transformer-based methods, together with an improved SSDIR baseline from the preceding phase of this research.

The methodology, findings and experimental results discussed in this chapter are included in a manuscript currently under review for publication. As in previous research phases, I served as the principal author of this work, developing the proposed method, designing and executing the experimental protocols, conducting evaluation and ablation studies, and preparing the manuscript.

The structure of this chapter is organised as follows: Section 5.1 outlines the motivation and background, followed by a relevant literature review in Section 5.2. Section 5.3 introduces the DetDiff model, emphasising the novel use of detection-guided diffusion, whereas Section 5.4 presents a comprehensive evaluation of the model across several datasets and tasks, including an ablation study to isolate the impact of the detection guidance mechanism. Finally, Section 5.5 provides conclusions and discusses opportunities for further work.

5.1 Introduction

Understanding complex visual scenes is a fundamental challenge in computer vision, aiming to emulate human perception of the surrounding world. Many supervised methods acquire this ability implicitly through learning to solve high-level tasks such as semantic segmentation or visual question answering. Object-centric representations decompose images into structured entities, capturing object attributes and relationships in a compositional and more interpretable manner. Methods for multi-object representation learning provide a way to map images to structured representations [8], object-centric embeddings that support generalisation and downstream inference.

The early models for multi-object representation learning leveraged structured variational autoencoders (VAEs) [101, 150], which learned to reconstruct simple images through a structured bottleneck in an unsupervised way [31, 53, 104]. The primary challenge addressed in later work was scaling these methods to more complex datasets by introducing more sophisticated object encoding mechanisms [14, 65, 124], utilising sequential information from videos [92, 103], or introducing weak or self-supervision [48, 103, 155]. However, these methods often struggled with more complex datasets, entangling multiple object representations and lacking the ability to generate high-quality samples.

Recently, research has shifted towards enhancing decoder capability, demonstrating its crucial role in determining the fidelity of reconstruction and, consequently, the quality of the learned representations [90, 158, 188]. Despite applying strong generative frameworks such as latent diffusion models [152], modern methods still tend to merge multiple objects into a single representation. Furthermore, existing approaches entangle all object attributes, such as their appearance, location, and inter-object relationships, making structured manipulation challenging [188].

This chapter introduces Detection-Guided Latent Diffusion (DetDiff), a novel object-centric generative framework that integrates detection guidance into representation learning. By leveraging a pre-trained object detector, DetDiff ensures that object encodings are grounded in actual detected entities, addressing challenges in object localisation and correspondence that existing generative models struggle with. Unlike transformer-based or recurrent approaches, DetDiff employs spatial attention with multi-scale grids, enabling efficient encoding of object-centric representations even from high-resolution images. Furthermore, DetDiff introduces a more disentangled representation space by separating object appearance from spatial information through dedicated positional encodings and a per-object alignment loss. Trained as a Latent Diffusion Model (LDM), this design not only retains generation quality but also produces richer and more controllable object representations, facilitating compositional reasoning.

The performance of DetDiff is evaluated by assessing the fidelity of generated images, analysing the object-to-representation matching, and applying the representations in a downstream vision–language reasoning task. The results demonstrate the advantages of detection guidance compared to state-of-the-art baselines, particularly in terms of representation quality and objects' disentanglement.

5.2 Related Works

Object-centric learning methods aim to decompose scenes into structured representations that correspond to distinct entities within an image. Unsupervised multi-object representation learning is typically facilitated through reconstruction-based learning, in which models infer a structured latent space that encodes attributes, spatial relationships, and compositionality.

Early approaches built on structured variational autoencoders (VAEs) [101, 150], with AIR [53] introducing a spatial attention mechanism to sequentially attend to and reconstruct individual objects via box-based inference and a patch-based decoder. To improve encoding efficiency, SPAIR [31] extended this idea by encoding object representations through spatial grid-based attention, allowing multi-object representation inference in a single forward pass. DetDiff builds on this encoding scheme, enhancing it with multi-scale grids [208] to better handle objects of varying sizes in natural, densely packed scenes.

To enhance reconstruction quality later approaches replaced boxes with mask-based mixing for decoding images [14, 49, 65, 120, 124], merging each object's appearance based on its iteratively inferred and/or refined attention mask. These methods achieve higher fidelity in reconstructions at the cost of increased computational complexity; they continue to face challenges when applied to real-world datasets. Recent research attempted to improve realistic image generation by employing a powerful transformer-based decoder [158], which autoregressively reconstructs discretised visual tokens from dVAE-encoded images. Another approach is to implement a latent diffusion model (LDM [81, 152]), in which the denoising process is conditioned on structured scene representations [90, 188]. DetDiff follows the LDM-based paradigm, leveraging object-centric representations as conditioning to improve reconstruction quality and robustness in complex, multi-object scenes.

A fully unsupervised setup for learning a structured scene representation often leads to an entangled representation, in which multiple objects are merged into a single latent entity, limiting applicability in tasks that require object-level reasoning. To address this, additional training signals have been incorporated to improve object-to-representation matching, such as temporal consistency in video-based approaches [32, 92, 103, 104, 160], weak supervision [48, 208, 209], or modified training protocols [96, 155]. DetDiff follows the approach proposed in Chapters 3 and 4, incorporating knowledge from a pre-trained object detection model to guide representation learning.

Diffusion models (DMs) [81] achieve high-quality image synthesis by progressively refining noisy inputs. Latent Diffusion Models (LDMs) [152], which improve the scalability of DMs, are commonly conditioned on text embeddings. While this can provide semantic control, recent work introduced layout conditioning to facilitate spatial control via layout-aware generation [51, 189, 204]. However, these methods continue to rely on text captions. DetDiff infers scene concepts directly from images, enabling explicit spatial control without textual supervision. In contrast to previous layout-guided LDMs, DetDiff places emphasis on representation learning rather than image generation.

5.3 Detection-Guided Latent Diffusion

This section introduces Detection-Guided Latent Diffusion (DetDiff), an object-centric representation learning model that enhances disentanglement by incorporating detection guidance from a pre-trained object detector.

5.3.1 DetDiff Latent Diffusion Model

The proposed method is a generative model learning structured object-centric representations of complex visual scenes in images. A pre-trained one-stage object detection model is used to identify object locations, providing detection guidance to capture each object representation via a spatial grid-based attention mechanism inspired by the SSDIR approach [208]. These representations serve as encodings that condition a latent diffusion model (following [90] and [188]), allowing reconstruction of the input image through a reverse diffusion process.

Object Representation Encoding. The DetDiff encoder (Figure 5.1) incorporates the spatial grid-based attention approach within its architecture. Specifically, multi-level feature maps, derived using a convolutional backbone, serve as spatial attention grids. A pre-trained YOLOv4 model is integrated [12], a one-stage, multi-scale object detection model known for its balance between detection accuracy and computational efficiency. The YOLOv4 architecture comprises the CSPDarknet53 backbone and the PANet neck [121]; during training, the backbone remains frozen, while the neck is fine-tuned to allow enhanced feature extraction for learning multiple object representations.

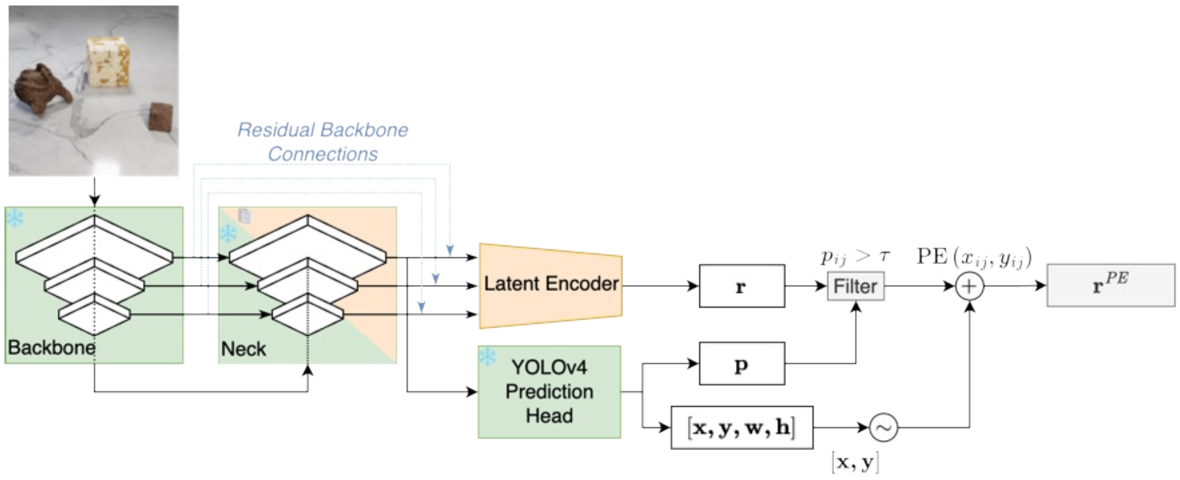


Figure 5.1: Overview of the object representation encoding (ϕ) in DetDiff. Given an input image, feature extraction is performed using a modified YOLOv4 model. Then, the Latent Encoder produces representations $\mathbf{r}$, while YOLOv4 Prediction Head outputs object presence probabilities $\mathbf{p}$ and box location $[\mathbf{x}, \mathbf{y}, \mathbf{w}, \mathbf{h}]$. The representations are filtered using the probabilities, and then enriched with 2D positional encodings, creating final representations $\mathbf{r}^{\text{PE}}$.

The YOLOv4 implementation in DetDiff produces predictions at three distinct resolutions (grids), facilitating robust feature representations for objects of varying sizes, particularly useful in densely populated scenes. Additionally, residual connections from the backbone to the multi-scale grids are introduced, enriching the dimensionality of the feature maps and enhancing the quality of the representations.

To enable information sharing across multi-scale feature maps, DetDiff employs cross-resolution convolutional integration modules, similar to the Mixer modules introduced in Chapter 4. Each integrated feature map is subsequently processed by a sequence of convolutional layers (each consisting of 3×3 convolutional layers, batch normalisation, and Mish activation function), producing three grids with dimensions $H^{(l)} \times W^{(l)} \times D$, where $H^{(l)}$ and $W^{(l)}$ denote the height and width of the l-th grid, respectively, and D denotes the dimensionality of the latent representations (a result of applying a final 1×1 convolutional layer).

Object representations are filtered based on their associated detection confidence scores. To stabilise training and reduce variance in input conditioning, only top 10 detected objects are retained during model fitting. Positional encodings are computed from the spatial coordinates of the inferred objects, with positional resolution downscaled to 128×128 to limit spatial variability.

Detection Guidance. The detection guidance approach improves object representation by leveraging a pre-trained object detector in two steps: (1) by enforcing focus on plausible object locations through confidence-based filtering, and (2) by encoding spatial information via positional encodings.

DetDiff incorporates filtering of spatial features using confidence scores obtained from a pre-trained YOLOv4 detector. Specifically, for each grid cell (i, j), YOLOv4 predicts a detection confidence score, interpreted here as the object presence probability p_{ij}, which is spatially aligned with the encoder's latent representation $\mathbf{r}_{ij}$. These pairs are matched across all grid levels and cells, and only the latent vectors corresponding to cells where $p_{ij} > \tau$ are retained. This thresholding step ensures that only regions likely to contain objects contribute to the learned representation.

Positional encoding is employed to explicitly embed spatial information within the object representations. Positional encodings [176], based on the YOLOv4-predicted object centre coordinates (x_{ij}, y_{ij}), are introduced. Two-dimensional sinusoidal encodings $\mathrm{PE}(x_{ij}, y_{ij})$, as defined in [183], are used and combined with each filtered latent representation $\mathbf{r}_{ij}$:

$$\mathbf{r}_{ij}^{\mathrm{PE}} = \mathbf{r}_{ij} + \mathrm{PE}\left(x_{ij}, y_{ij}\right) \tag{5.1}$$

This approach disentangles an object's identity and its spatial location, improving representation alignment with corresponding objects and enabling generalisation across various spatial configurations during the diffusion process.

Representation-Conditioned Latent Diffusion. Diffusion models are generative models capable of reconstructing data by iteratively removing noise [81]. Given an initial data sample $\mathbf{x}^{(\mathbf{0})}$, the forward diffusion process gradually adds Gaussian noise over a sequence of time steps t. The reverse generative process iteratively denoises the sample, starting from random noise $\mathbf{x}^{(T)}$ and progressing backwards through time. In the context of generative modelling, the noise to be subtracted from the sample is predicted using a denoising neural network.

Latent diffusion models reduce the computational cost by performing the diffusion process in a lower-dimensional latent space, using pre-trained latent space mapping models [152]. In DetDiff, an auto-encoder pre-trained on OpenImages (KL-F8) is employed. This consists of an encoder e, which maps images into latent representations, and a decoder d, which reconstructs images from these latents. The auto-encoder remains frozen throughout training.

In this setup, the diffusion process is formulated as:

$$\mathbf{z}^{(0)} = e\left(\mathbf{x}\right); \quad q\left(\mathbf{z}^{(t)} \mid \mathbf{z}^{(t-1)}\right) = \mathcal{N}\left(\mathbf{z}^{(t)}; \sqrt{1-\beta_t}\mathbf{z}^{(t-1)}, \beta_t \mathbf{I}\right) \tag{5.2}$$

In LDMs, conditioning the denoising network θ is typically achieved using supervised text embeddings. In DetDiff, following [90] and [188], the denoising network is conditioned on the representations $\mathbf{r}_{\phi}^{\mathrm{PE}}$ inferred by the encoder ϕ, which is jointly trained with the denoising network without supervision. The denoising step at each time t is defined as:

$$p_\theta\left(\mathbf{z}^{(t-1)} \mid \mathbf{z}^{(t)}, \mathbf{r}_{\phi}^{\mathrm{PE}}\right) = \mathcal{N}\left(\mathbf{z}^{(t-1)}; \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{z}^{(t)} - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta\left(\mathbf{z}^{(t)}, \mathbf{r}_{\phi}^{\mathrm{PE}}, t\right)\right), \beta_t \mathbf{I}\right) \tag{5.3}$$

where:

$\epsilon_\theta\left(\mathbf{z}^{(t)}, \mathbf{r}_{\phi}^{\mathrm{PE}}, t\right)$ is the noise prediction model,
β_t is a variance schedule value at timestep t,

$\alpha_t = 1 - \beta_t$,
$\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$.

Overall, the reverse diffusion process models the posterior distribution:

$$p_\theta\left(\mathbf{z}^{(0:T)} \mid \mathbf{r}_\phi^{\text{PE}}\right) = p\left(\mathbf{z}^{(T)}\right) \prod_{t=1}^{T} p_\theta\left(\mathbf{z}^{(t-1)} \mid \mathbf{z}, \mathbf{r}_\phi^{\text{PE}}\right) \tag{5.4}$$

By sampling from this posterior (the denoising procedure [81]), the model iteratively generates progressively denoised latent representations from noisy samples $(\mathbf{z}^{(T)}, \mathbf{z}^{(T-1)}, ...\mathbf{z}^{(0)})$, leading towards the original latent representation. Finally, it can be decoded back to the image space using the decoder: $\hat{\mathbf{x}} = d\left(\mathbf{z}^{(0)}\right)$.

The reverse diffusion process assumes the use of a denoising network to predict the noise at each time step. The approach used in DetDiff follows the original design of LDMs [152], employing an adapted UNet architecture augmented with a representation-conditioned transformer. The network comprises a stack of multiple UNet blocks, each followed by a cross-attention transformer layer, integrating the representation $\mathbf{r}_\phi^{\text{PE}}$.

5.3.2 Training Procedure

The DetDiff training procedure adjusts the LDM training paradigm [152], jointly training the DetDiff encoder ϕ and the noise predictor θ. Figure 5.2 presents an overview of DetDiff's latent diffusion model training setup. Given an input image $\mathbf{x}$, structured representations are first obtained as $\mathbf{r}_\phi^{\text{PE}} = \phi(\mathbf{x})$, along with the latent space embedding of the image $\mathbf{z}^{(0)} = e(\mathbf{x})$. A random timestep t is sampled uniformly, and the latent $\mathbf{z}^{(0)}$ is noised accordingly:

$$\mathbf{z}^{(t)} = \sqrt{\bar{\alpha}_t}\mathbf{z}^{(0)} + \sqrt{1 - \bar{\alpha}_t}\boldsymbol{\epsilon}_t, \quad \boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \tag{5.5}$$

The noise prediction network is used to estimate the noise given the noised latent $\mathbf{z}^{(t)}$, the representations $\mathbf{r}_\phi^{\text{PE}}$, and the timestep t: $\hat{\boldsymbol{\epsilon}}_t = \epsilon_\theta\left(\mathbf{z}^{(t)}, \mathbf{r}_\phi^{\text{PE}}, t\right)$. The network is trained by minimising the mean squared error between the predicted noise and the true noise:

$$\mathcal{L}_{\text{LDM}}(\theta, \phi) = \mathbb{E}_{t,\boldsymbol{\epsilon}}\left[\|\hat{\boldsymbol{\epsilon}}_t - \boldsymbol{\epsilon}_t\|^2\right] \tag{5.6}$$

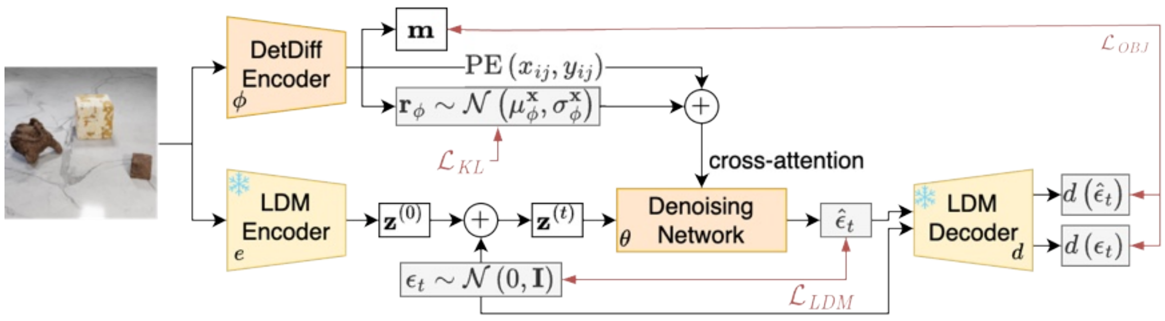


Figure 5.2: Overview of DetDiff LDM training. Given an input image, the DetDiff encoder (Fig. 5.1) produces normal distribution parameters, from which structured object representation $\mathbf{r}_\phi$ is sampled, as well as objects binary mask, created from YOLOv4 bounding box prediction. During training, the Denoising Network predicts the noise $\hat{\boldsymbol{\epsilon}}_t$ added to the latent space encoding of the input image ($\mathbf{z}^{(0)}$). Then, the predicted noise and the true noise are mapped to the original image space via the LDM Decoder. Red lines indicate each loss component: $\mathcal{L}_{\text{LDM}}$ (Eq. 5.6), $\mathcal{L}_{\text{OBJ}}$ (Eq. 5.7) and $\mathcal{L}_{\text{KL}}$ (Eq. 5.8).

To improve reconstruction quality in regions where objects are detected, DetDiff incorporates an additional pixel-space loss. As YOLOv4 provides bounding boxes that indicate likely object locations, these are used as a self-supervised signal to guide the model's focus during training. Specifically, a mask $\mathbf{m}$ is derived from the bounding boxes and applied to the image-space noise prediction error loss, placing greater emphasis on accurate reconstruction within object regions:

$$\mathcal{L}_{\text{OBJ}}(\theta, \phi) = \mathbb{E}_{t,\epsilon}\left[\left\|\mathbf{m} \odot \left(d\left(\hat{\boldsymbol{\epsilon}}_t\right) - d\left(\boldsymbol{\epsilon}_t\right)\right)\right\|^2\right] \tag{5.7}$$

where:

$\mathbf{m}$ is the binary mask that indicates the presence of the object according to the bounding boxes predicted by YOLOv4,

$\odot$ denotes element-wise multiplication.

Finally, to encourage a smooth and continuous latent space, DetDiff includes a KL divergence regularisation term. The encoder ϕ outputs the parameters of a Gaussian distribution, from which latent representations $\mathbf{r}_\phi$ are sampled: $\mathbf{r}_\phi \sim \mathcal{N}(\mu_\phi(\mathbf{x}), \sigma_\phi(\mathbf{x}))$. The KL term penalises deviations from the standard normal prior, promoting smooth interpolation in the latent space and improved generalisation:

$$\mathcal{L}_{\text{KL}}(\phi) = D_{\text{KL}}\left[\mathcal{N}\left(\mu_\phi(\mathbf{x}), \sigma_\phi(\mathbf{x})\right) \,||\, \mathcal{N}(\mathbf{0}, \mathbf{I})\right] \tag{5.8}$$

The final training objective combines these three terms (weighted by λ):

$$\mathcal{L} = \mathcal{L}_{\text{LDM}} + \lambda_{\text{OBJ}}\mathcal{L}_{\text{OBJ}} + \lambda_{\text{KL}}\mathcal{L}_{\text{KL}} \tag{5.9}$$

5.4 Experiments

This section presents the results of a comprehensive evaluation of the DetDiff model across a range of tasks, aimed at demonstrating its effectiveness and robustness. The quality of image reconstructions produced via the reverse diffusion process is assessed, alongside an analysis of the structural quality of the learned latent representations, particularly their alignment with corresponding objects within the images. Furthermore, the practical utility of DetDiff's representations is evaluated through application to a visual question answering (VQA) task. Finally, an ablation study is conducted to examine the impact of the proposed detection guidance components.

Implementation Details. Table 5.1 presents the key model hyperparameters for all datasets. The pre-trained YOLOv4 model is trained separately for each dataset. During DetDiff training, YOLOv4 predictions are sampled using a Bernoulli distribution, with the predicted confidence score serving as the sampling probability. In addition, a small number of negative examples are included at this stage to improve the generalisation capability of the encoder. At inference time, a fixed threshold $\tau = 0.25$ is applied, followed by non-maximum suppression to eliminate duplicate detections.

To stabilise training, particularly when integrating pre-trained components like the detector and autoencoder, a learning rate schedule is employed, consisting of a one-epoch warm-up phase followed by linear annealing. The KL divergence loss component is scheduled trapezoidally: it is initially suppressed, then gradually increased and subsequently held constant. This allows the model to learn useful representations prior to being encouraged to form a well-structured latent space. All models are trained on a single NVIDIA H100 GPU using bfloat16 precision.

Datasets. DetDiff is evaluated on three datasets of increasing complexity, enabling assessment of model's performance across progressively more challenging scenarios, ranging from synthetic compositional scenes to real-world images:

1. **CLEVRTex** [98] is a synthetic dataset augmenting the CLEVR dataset, featuring procedurally generated scenes with multiple objects placed on complex textured backgrounds. It contains 50 000 images, of which the default split is used: 80% for

Table 5.1: Implementation details of DetDiff.

hyperparameter	**value**
Learning rate	0.0001 (1 epoch warmup; linear annealing)
Image size	256×256
UNet sample size	32
UNet attention head dimension	24
UNet block out channels	[192, 384, 768, 768]
Per object loss coefficient	$\lambda_{\mathrm{OBJ}} = 10$
KL divergence scheduler	trapezoidal, max value $\lambda_{\mathrm{KL}} = 0.1$, warmup 1 epoch
Noise scheduler	DDPMScheduler, $\beta_{start} = 0.00085$, $\beta_{end} = 0.012$
Optimiser	AdamW8bit, weight decay 0.01
Number of training epochs	100 with early stopping
Gradient clipping	1.0
Latent representation size	192
Number of encoder convolutional blocks	2
Negative objects added	1%
NMS threshold	0.45
Positional encoding resolution	128

training, 10% for validation, and 10% for testing. Each image includes between 3 and 10 objects from 4 possible shape categories. The original annotations are converted into bounding boxes, which are used for training the YOLOv4 model.

2. **MOVi-E** [64] is a dataset designed for object-centric learning, featuring dynamic and occluded objects rendered in realistic 3D environments. The default split provides 9 749 sequences of 24 frames for training (approximately 234 000 images), and 250 sequences of equal length for testing (6 000 images). As DetDiff operates on static images, each frame is treated as an independent image, and 10% of the training set is extracted for validation. Compared to other versions of the MOVi dataset, MOVi-E introduces viewpoint variation and a high degree of occlusion, with scenes containing up to 23 objects from 17 possible categories.

3. **MS COCO** [119] is a real-world dataset containing natural images with a wide variety of object categories, backgrounds, and occlusions. It represents the most complex setting in the evaluation. Additional VQA annotations [63] are used to support the downstream task experiment. The dataset comprises approximately 118 000 training images and 5 000 validation images (used as the test subset, since annotations for the test set are not publicly available). For DetDiff training, all images without annotated objects are removed, and 10% of the training set is extracted as a validation subset.

For each dataset, an input resolution of 256×256 is used; however, the convolutional nature of DetDiff's encoder allows for straightforward scalability to higher resolutions.

Reference Methods. DetDiff was compared with four representative reference methods:

1. **Latent Slot Diffusion** [90] is a slot attention-based latent diffusion model similar in setup to DetDiff. The denoising network is conditioned on object-centric slots, which are inferred without supervision from the input image.
2. **SlotDiffusion** [188], similar to Latent Slot Diffusion, is a latent diffusion model conditioned on object slots. Key differences include the use of a different latent space encoding model: SlotDiffusion employs a VQ-VAE trained jointly with the model, rather than a pre-trained autoencoder as in Latent Slot Diffusion.
3. **RDIR** [209], specifically its non-recurrent variant, employs a structured convolutional encoder similar to that used in DetDiff but relies on a rendering-based reconstruction method. Although this rendering approach constrains the fidelity of reconstructed images, the learned representations demonstrate strong performance in downstream tasks relative to other baselines.
4. **SPOT** [96] utilises an autoregressive transformer decoder in place of a latent diffusion model. By enhancing segmentation quality through a self-training stage, SPOT achieves robust and high-quality scene representations within the slot attention framework, offering insight into state-of-the-art transformer-based generative techniques.

Table 5.2 presents summary of key differences between each of the reference methods used in this research.

5.4.1 Experiment 1: Reconstruction Fidelity

To evaluate the reconstruction quality of DetDiff, it is compared with other LDM-based methods. The testing subsets of each dataset (CLEVRTex, MOVi-E, and COCO) are used; each image is passed through the respective model's encoder to obtain a representation of the scene objects (either as slots or structured encodings). This representation is then used as conditioning for the reverse diffusion process to generate a set of reconstructions. The Fréchet Inception Distance (FID) score [78] is computed by comparing the generated images with those from the full training dataset. Additionally,

Table 5.2: Architectural comparison of reference methods.

	LSD	**SlotDiffusion**	**RDIR**	**SPOT**	**DetDiff**
encoding	slot attention	slot attention	spatial attention	slot attention	spatial attention
decoding	LDM	LDM	rendering CNN	autoregressive transformer	LDM
backbone	simple CNN encoder	ResNet18 / ViT-S/8	YOLOv4	ViT-B/16 / ViT-S/8	YOLOv4
image size	256	128 / 224	256	224	256
training objective	noise prediction error	noise prediction error	reconstruction error	reconstruction error + self-training	noise prediction error + detection guidance
latent space encoding	Pre-trained AE	VQ-VAE trained jointly	-	-	Pre-trained AE

reconstruction fidelity is assessed using Mean Squared Error (MSE), calculated between each generated image and its corresponding original input image; the generation process is repeated three times with different random seeds. The results are presented in Table 5.3.

Table 5.3: FID and MSE scores of generated images for the test subset. MSE values are reported in $\times 10^{-2}$, averaged over 3 random seeds. DetDiff reconstruction quality is generally lower, which may result from the lack of explicit background modeling.

dataset	**method**	**FID** $\downarrow$	**MSE** $\downarrow$
CLEVRTex	SlotDiffusion	19.18	1.646 $\pm$ 0.001
	LSD	**8.08**	**0.468** $\pm$ 0.001
	DetDiff	23.51	2.405 $\pm$ 0.002
MOVi-E	SlotDiffusion	54.70	**1.282** $\pm$ 0.002
	LSD	**25.61**	1.377 $\pm$ 0.002
	DetDiff	28.75	2.128 $\pm$ 0.002
MS COCO	SlotDiffusion	**15.51**	5.774 $\pm$ 0.006
	LSD	19.02	**2.554** $\pm$ 0.001
	DetDiff	38.10	5.09 $\pm$ 0.223

Example reconstructions for each LDM-based method are shown in Figure 5.3.

While DetDiff demonstrates competitive reconstruction performance in certain cases, it struggles with background modeling. Both SlotDiffusion and LSD leverage mask-based approaches, that enable comprehensive scene reconstruction, whereas DetDiff

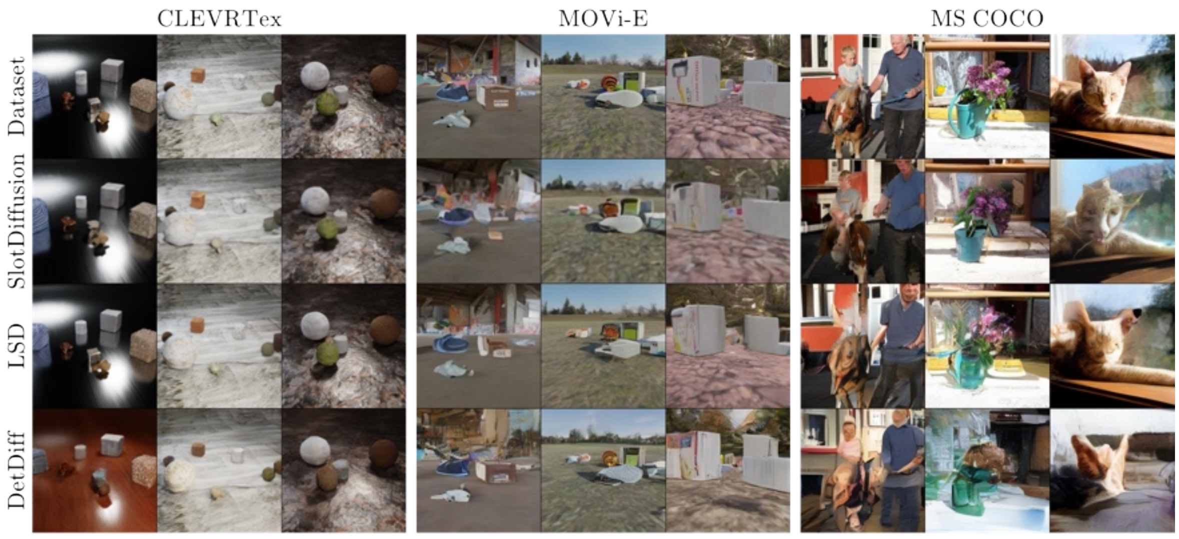


Figure 5.3: Qualitative comparison of generated samples from test images' representation. Similar infidelity issues occur, with lower quality of DetDiff's background.

learns localised object representations based on detected entities, which forces the model to encode background details within object representations, reducing overall reconstruction quality.

DetDiff performs worse than other methods on CLEVRTex both in the case of FID and MSE. However, the results are better for the MOVi-E dataset, which, despite featuring a more complex object set, uses simpler backgrounds. In contrast, for the real-world MS COCO dataset, the fidelity of DetDiff images is worse, even though its MSE is lower than SlotDiffusion's.

Visual inspection supports this observation: even though DetDiff effectively reconstructs individual objects' appearances, the background quality is often inferior compared to other methods. This suggests that DetDiff's per-object loss function helps preserve object integrity but at the cost of overall scene coherence.

5.4.2 Experiment 2: Object-to-Representation Matching Quality

This experiment provides a setup to evaluate how well the model understands each object within the scene and its capability of matching object representations its appearance in the image space. The idea is to assess whether removing an object from the encoded representation results in a coherent reconstruction where the correct object is successfully omitted. For that, experimentation follows these steps:

1. Each model's encoder processes the input image to generate a structured representation of the scene.
2. The representations are used in the LDM to generate a reconstructed image, establishing a baseline output for each model.
3. A pre-trained object detection model is employed as an *oracle* to detect objects in both the original input image and the reconstructed output. Average Precision (AP) is computed with respect to the ground-truth bounding boxes, denoted as AP_{in} for the input image and AP_{rec} for the reconstruction.
4. Each encoded object representation is matched with the corresponding ground-truth annotation and predicted bounding box using Intersection over Union (IoU).
5. One object is randomly removed from the representation set; it is also excluded from the ground-truth annotations.
6. The new, modified representation is used in a second reverse diffusion process to generate an image with the object removed.
7. The AP is recomputed for the original input image (excluding the removed object from both annotations and predictions), and again based on the oracle's predictions for the newly reconstructed image.

This process is shown in Figure 5.4 for a sample image. Successful object-to-representation alignment should result in a coherent scene with the correct object removed, which should be reflected by a proportional drop in AP following object removal on both the input image and the reconstructed image (since the removal of a true positive (TP) object reduces precision). This experiment is conducted on the CLEVRTex and MOVi-E datasets, with the results reported in Table 5.4.

Across both datasets, DetDiff exhibits consistent reconstruction behaviour when removing an object, as indicated by the similarity in Δ values between *all* and -1. On CLEVRTex, SlotDiffusion reaches comparable AP_{rec}, and Δ values between *all* and -1, indicating that it maintains a structured approach to object representation. LSD performs even better, reaching a higher baseline AP_{rec}. However, both SlotDiffusion and LSD struggle significantly on MOVi-E, where their reconstruction AP deteriorates.

The performance drop of SlotDiffusion and LSD on MOVi-E is particularly notable, as both models exhibit low AP scores on reconstructed images. This may suggest that they struggle to dedicate slots to individual objects in scenes with a large number of

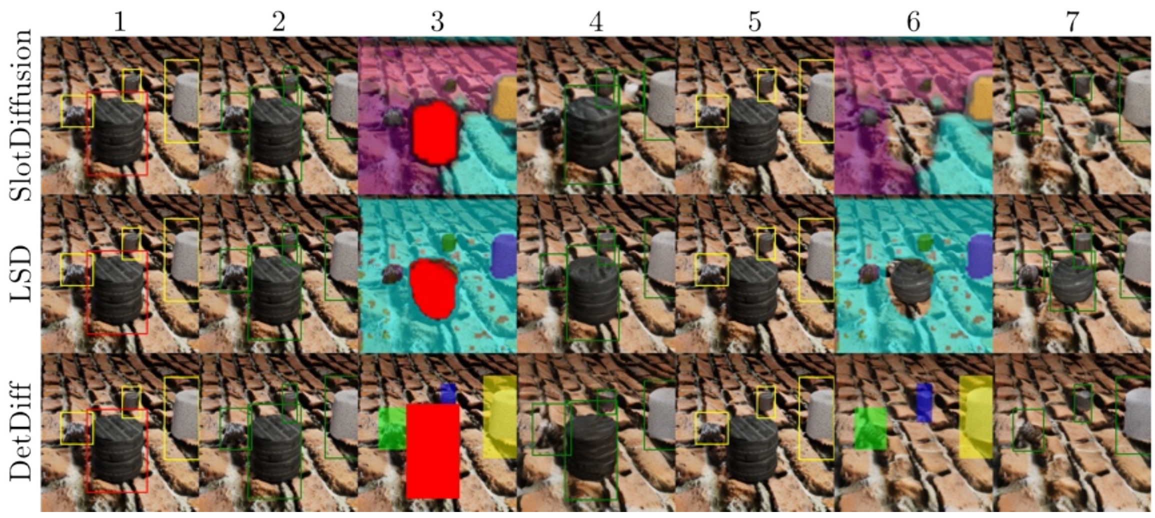


Figure 5.4: Object-to-representation experiment steps. Images 1, 2 and 3 present the annotation, oracle prediction and model's attention on the original image respectively; the red colour indicates the object selected for removal. Image 4 is the reconstruction with the oracle's prediction. Image 5 shows annotations with the selected object removed. Image 6 is the reconstruction with the model's attention where the object's representation was removed. Image 7 shows the oracle's prediction on the new reconstruction.

overlapping entities. In contrast, DetDiff achieves a higher AP on reconstructions thanks to the explicit guidance from object detection, allowing the allocation of independent representations to all objects.

5.4.3 Experiment 3: Applying Representations in Visual Question Answering Downstream Task

To assess the effectiveness of the learned representations in complex reasoning tasks, DetDiff is evaluated on the Visual Question Answering (VQA) task using the VQA-v2 dataset [63], which provides questions about images from the MS COCO dataset [119]. A similar experimental setup to that described in [129] is followed, adapting their downstream framework to DetDiff and other reference methods:

1. The VQA-v2 questions are filtered to include only yes/no and numeric answers ranging from 0 to 14, reducing the answer space to 17 possible choices.

2. Each method's encoder is used to extract a visual representation **r** from images, while a pre-trained Text-to-Text Transfer Transformer (T5) [144] provides embeddings **t** for the text-based questions.

Table 5.4: Object-to-representation matching evaluation via oracle's object detection performance. Average Precision (AP) is measured using an oracle detector on input images (in) and their reconstructions (rec), for the full scene (*all*) and after removing a randomly selected object (−1). Δ represents the relative drop in AP. Similar Δ values across conditions suggest successful object removal, while smaller drops indicate more faithful object reconstruction.

dataset	**method**		$\mathbf{AP_{in}}$	$\mathbf{AP_{rec}}$	Δ [%]
CLEVRTex	SlotDiffusion	*all*	0.463	0.238	−48.7
		−1	0.393	0.223	−43.3
	LSD	*all*	0.486	**0.412**	−15.2
		−1	0.411	**0.374**	−9.1
	DetDiff	*all*	0.486	0.275	−43.5
		−1	0.408	0.277	−32.3
MOVi-E	SlotDiffusion	*all*	0.455	0.088	−80.7
		−1	0.418	0.078	−81.4
	LSD	*all*	0.456	0.106	−76.7
		−1	0.418	0.097	−76.8
	DetDiff	*all*	0.457	**0.152**	−66.7
		−1	0.398	**0.130**	−67.3

3. Both representations are projected to a unified latent space of equal embedding size ($\mathbf{r}'$ and $\mathbf{t}'$); then sinusoidal positional encodings are applied to $\mathbf{t}'$ and a one-hot vector is appended to differentiate image and text encodings.

4. The concatenated representations are processed by a transformer encoder layer with N_t layers, with a trainable CLS token. The token is then passed through an MLP classifier, predicting probability over the 17 possible answers.

Table 5.5 presents the classification accuracy across 5 different seeds, with both image and text encodings generated offline for faster processing. Two control experiments are also included: ResNet50 [76], where features from a pre-trained convolutional backbone are reshaped to form tokens and used as object encodings, and Text-only, where no image encodings were used. In this study, accuracy was found to be consistent with the F1-score; therefore, only the simpler accuracy metric is reported.

The results show that DetDiff improves over previous spatial attention-based methods, demonstrating the benefits of using an LDM-based decoding strategy over a structured autoencoder from RDIR. DetDiff performs slightly worse than SlotDiffusion and LSD and falls behind SPOT, which attains the highest accuracy. However, unlike

Table 5.5: Accuracy on VQA-v2 downstream task. DetDiff provides significant improvement over control text-only approach and pre-trained convolutional network, but falls behind slot attention-based models. Results are averaged over 5 random seeds.

method	**accuracy** $\uparrow$
SlotDiffusion	0.542 ± 0.004
LSD	0.536 ± 0.003
RDIR	0.517 ± 0.001
SPOT	$\mathbf{0.561 \pm 0.001}$
DetDiff	0.530 ± 0.003
ResNet50	0.517 ± 0.004
Text only	0.497 ± 0.001

DetDiff, neither of these methods provides positional disentanglement or explicit representations for each entity; instead, they tend to group similar or close objects in one representation, which can be beneficial in the case of the VQA task.

5.4.4 Ablation Study: Detection Guidance

An ablation study is conducted to assess the contribution of the detection guidance components in DetDiff. Two additional model variants are evaluated:

1. without per-object loss $\mathcal{L}_{\text{OBJ}}$ (Equation (5.7)) - OBJ$_\sim$: the object-aware loss component is removed to evaluate the importance of object-focused learning;
2. without positional encoding (Equation (5.1)) - PE$_\sim$: positional encodings are omitted from the representation to assess their role in the model's spatial awareness.

Table 5.6: Ablation results on the reconstruction quality (FID and MSE). OBJ$_\sim$ omits the object-aware loss, and PE$_\sim$ removes positional encoding. MSE values are reported in $\times 10^{-2}$ (averaged over 3 random seeds). Full guidance improves reconstruction quality on CLEVRTex, while removing positional encoding slightly improves metrics on MOVi-E.

dataset	**variant**	**FID** $\downarrow$	**MSE** $\downarrow$
CLEVRTex	DetDiff	**23.51**	**2.405** $\pm$ 0.018
	OBJ$_\sim$	24.42	3.282 $\pm$ 0.032
	PE$_\sim$	24.56	2.679 $\pm$ 0.030
MOVi-E	DetDiff	28.75	2.128 $\pm$ 0.002
	OBJ$_\sim$	28.63	2.365 $\pm$ 0.001
	PE$_\sim$	**27.85**	**2.073** $\pm$ 0.008

The Reconstruction Quality (5.4.1) and Object-to-Representation Matching Quality (5.4.2) experiments are repeated using the same experimental setup. The results of these ablation experiments are presented in Table 5.6 and 5.7.

Table 5.7: Ablation results on object-to-representation matching. Average Precision (AP) is reported for input images (AP_{in}) and reconstructions (AP_{rec}) across two conditions: full scene (*all*) and with one object removed (-1). Δ indicates the percentage drop in AP from input to reconstruction. $\text{OBJ}_{\sim}$ removes the object-aware loss; $\text{PE}_{\sim}$ removes positional encoding. Full detection guidance yields better object-to-representation matching.

dataset	**variant**		$\mathbf{AP_{in}}$	$\mathbf{AP_{rec}}$	Δ [%]
CLEVRTex	DetDiff	*all*	0.486	**0.275**	−43.5
		−1	0.408	**0.277**	−32.3
	$\text{OBJ}_{\sim}$	*all*	0.486	0.254	−47.7
		−1	0.408	0.258	−34.8
	$\text{PE}_{\sim}$	*all*	0.486	0.240	−50.7
		−1	0.408	0.231	−43.2
MOVi-E	DetDiff	*all*	0.457	**0.152**	−66.7
		−1	0.398	**0.130**	−67.3
	$\text{OBJ}_{\sim}$	*all*	0.457	0.141	−69.1
		−1	0.397	0.120	−69.8
	$\text{PE}_{\sim}$	*all*	0.457	0.141	−68.1
		−1	0.398	**0.130**	−67.3

The full DetDiff model consistently achieves the best performance on the CLEVRTex dataset, both in terms of image reconstruction quality and object-to-representation matching. However, on MOVi-E, the model without positional encoding ($\text{PE}_{\sim}$) outperforms the default model in terms of image generation quality. Despite this, full detection guidance demonstrates better object-to-representation matching, achieving the best AP scores. This indicates that in datasets consisting of videos with multiple similar frames and slight positional variations, positional encoding contributes less to object representation learning and may introduce unnecessary spatial bias.

5.5 Summary

This chapter introduced the Detection-Guided Latent Diffusion Model, which integrates detection guidance into a latent diffusion model-based framework for multi-object representation learning. By integrating a pre-trained object detector, DetDiff leverages the spatial grid-based attention mechanism to effectively encode complex scenes, ensuring that object representations align with actual entities, reducing entanglement and improving structured representations.

Experimental results demonstrate the high quality of learned representations while maintaining fidelity in generated images. It shows that DetDiff produces structured and explicitly separated object representations, successfully disentangling object position from appearance. Additionally, ablation studies confirm the critical role of detection guidance, with each of its components contributing to improved scene representation quality and reconstruction fidelity.

The method has several limitations. While detection guidance enhances object-to-representation alignment, leading to more precise representations of relevant objects in the scene, reliance on a pre-trained object detector may introduce biases inherent to the detector itself. Furthermore, like other LDM-based object-centric models, DetDiff encounters challenges in decoding natural images, often leading to visible distortions in generated outputs.

Future work could explore direct background modeling, disentangling it from the object appearances to enhance scene consistency. Another promising direction is to investigate self-supervised detection guidance, reducing dependence on external object detectors. Lastly, following other object-centric contributions, research could be directed to extend DetDiff for temporal scene understanding in video-based applications.

Chapter 6

Conclusions

This thesis explores the problem of learning structured, modular representations of multiple objects in images and videos. Existing approaches in the literature often rely on fully unsupervised object discovery pipelines, which struggle to scale effectively to complex, real-world datasets. On the other hand, methods based on iterative attention and refinement are computationally expensive, limiting their practicality. This work addresses these gaps by joining a research trend in multi-object representation learning by incorporating a pre-trained object discovery mechanism into the original approaches and investigates how spatially structured intermediate features one-stage detectors can guide the learning of object-centric representations under semi-supervised regimes.

The main contributions of this thesis are as follows:

1. Formulation of the research problem of learning modular and transferable object representations from images and videos by utilising internal feature maps of a pre-trained one-stage object detection neural network,
2. Evaluation of visual features extracted from a pre-trained one-stage object detection network for deep reinforcement learning-based 3D robotic navigation,
3. Design and implementation of a scale-invariant encoding framework for multi-object representation learning in images, combining fully parallel spatial grid-based attention with multi-scale features and structured latent space,
4. Introduction of SSDIR, an object-centric generative model that integrates detection-based spatial supervision with unsupervised learning of appearance and depth latents,

5. Design of RDIR, a temporally consistent extension of SSDIR for video data, incorporating a recurrent Sequence encoder and a Mixer module to enable temporally coherent and robust object representations,
6. Proposal of a staged training procedure for improved training stability in the video-based object-centric model, involving re-using of a pre-trained object detection model, training an image-based model and fine-tuning with the temporal extension,
7. Introduction of a detection-guided object encoding pipeline for multi-object modelling, including positional encoding, multi-scale spatial fusion and object-conditioned latent diffusion,
8. Development of DetDiff, a generative framework that applies detection-guided supervision to a latent diffusion process conditioned on object-centric representations, achieving controllable object manipulation in image synthesis,
9. Comprehensive experimental validation of the proposed methods on synthetic and real-world datasets, demonstrating improvements in object disentanglement, scalability, temporal consistency, and utility in downstream tasks,
10. Validation of the thesis results through peer-reviewed publications.

The summary below outlines how each of these contributions addresses the research questions formulated in Chapter 1.4.

RQ 1: *To what extent can internal representations extracted from intermediate layers of an object detection neural network serve as effective visual inputs for deep reinforcement learning agents, especially in 3D robotic navigation?*

As shown in Chapter 2 (published as[210]), internal representations extracted from an object detection network (specifically the "basic features", which refer to mid-level convolutional features of a pre-trained TinyYOLO model) can serve as effective visual inputs for deep reinforcement learning agents in 3D robotic navigation, though with trade-offs. The study demonstrates that these features enable learning correct navigation behaviour, achieving over 40% success rates in reaching the target destination. This shows that intermediate representations of an object detection model retain semantically meaningful and general information that the agent can exploit for task-relevant decision-making, outperforming penultimate YOLO feature maps and end-to-end trainable visual encoders (Subsection 2.3.6).

However, the effectiveness of these internal representations is worse than the more interpretable bounding box-based location of the target object. Moreover, while the use of internal feature maps provides the RL agent with more information in the context of no detection, it also introduces a computational cost, with prediction speeds dropping by over 35% compared to bounding box-based designs.

RQ 2: *To what extent does fully convolutional spatial grid-based attention, which replaces sequential encoding or iterative refinement, enable the learning of high-quality object-centric representations in multi-object visual scenes?*

Chapter 3 (published as[208]) introduces SSDIR, a multi-object representation learning model that employs a fully convolutional spatial grid-based attention mechanism to infer representations of multiple objects in an image. This is achieved by leveraging the multi-scale feature maps of an SSD object detector, enabling parallel encoding of object representations without the need for iterative image analysis or object glimpse extraction. See Subsection 3.3.2 for more details.

The learned representations demonstrate strong performance on synthetic datasets, resulting in high-quality object reconstructions (Subsection 3.4.1), effective in downstream object classification (Subsection 3.4.2). However, the use of a simple convolutional backbone in SSDIR limits its performance on more realistic data, leading to degraded reconstruction quality (Subsection 3.4.1). Furthermore, the spatial grid-based attention mechanism inherits the limitations of one-stage object detectors, particularly their sensitivity to densely packed or overlapping objects (Subsection 3.4.3).

These scalability constraints were addressed in Chapter 4 (published as [209]), where the backbone was replaced with a more powerful architecture (Subsection 4.3.2), leading to improved object representation quality. Nonetheless, the method remains limited by the low resolution of reconstructed object patches and the constraints imposed by the rendering-based decoder.

RQ 3: *How does utilising multi-scale feature maps improve the performance of object-centric representations in downstream tasks, particularly for objects of varying sizes in complex visual scenes?*

The use of multi-scale feature maps as spatial grids improves the quality of object-centric representations in spatial attention models, particularly for objects of varying sizes in complex scenes. In SSDIR (Chapter 3), multi-scale feature maps extracted from a pre-trained SSD object detector are utilised as attention grids to localise and encode object representations, addressing the limitations of earlier models such as SPAIR and SPACE, which rely on a single fixed-resolution grid and struggle to capture objects of varying sizes appropriately. In contrast, SSDIR is able to encode objects across a range of scales, as demonstrated on the synthetic scattered multi-scale MNIST dataset. See Subsection 3.4.1 for more details.

This advantage is especially visible in the structure of the learned latent space. SSDIR produces scale-invariant representations that clearly separate digits into distinct clusters, while baseline methods often fragment objects into partial components, aligned with their fixed grid size. The effectiveness of these representations is confirmed in a downstream classification task, where SSDIR performs better than reference methods on the same multi-scale dataset (See Subsection 3.4.2).

In RDIR (Chapter 4), this design was enhanced through the introduction of the Mixer module, which facilitates cross-resolution information sharing across all grids (Subsection 4.3.2). This addition improves the consistency of representations by enabling the model to integrate features across scales. The importance of the addition of the Mixer module was analysed in the ablation study (see Subsection 4.4.3 for more details).

RQ 4: *How can an unsupervised learning framework that incorporates knowledge from a pre-trained object detector be used to learn structure latent variables, such as appearance and depth?*

SSDIR is an unsupervised model that enables learning of structured latent variables by incorporating prior knowledge from a pre-trained object detection model (Chapter 3). The encoder of SSDIR shares the convolutional backbone with a pre-trained SSD object detector, which serves as a feature extractor throughout the training process. See 3.3.2 for more details. SSDIR adopts the spatial grid-based attention mechanism used in one-stage detectors, adding object appearance and depth encoders to SSD prediction heads. During training, the weights of the backbone and detection heads (location and class confidence encoders) are frozen, and only the representation encoders are trained as a structured variational autoencoder. This setup enables the model learn representations of objects, attending only to regions associated with detected objects, eliminating the influence of background or irrelevant entities, as shown in experiments on more complex datasets (Subsection 3.4.1). The model benefits from the object detector's knowledge, while still learning appearance and depth representations without additional supervision.

RQ 5: *How can implicit representation-based temporal mechanisms be used to provide consistent representations for tracking object identities across video frames?*

Chapter 4 of this thesis presents RDIR, which applies an implicit temporal mechanism to improve object representations stability over time. It uses the Sequence encoder, which operates directly on an intermediate feature map extracted using a convolutional backbone. Treating it as a spatial grid, RDIR processes them using a

GRU-based module propagating temporal information through hidden states across video frames, without an explicit mechanism for object matching. To further improve the consistency of representations, RDIR integrates the Mixer module before and after the recurrent processing, mitigating the disconnect between objects detected at different grid resolutions. See Subsection 4.3.2 for more details.

Together, these architectural components improve the temporal stability of object-centric representations, which is demonstrated by the performance of models operating on learned representations in downstream tasks (Subsection 4.4.1 and 4.4.2).

RQ 6: *How does staged training procedure (translating from image-based to video-based representation learning) impact the ability to learn objects' representations in a video setting?*

The staged training procedure used in RDIR is critical to enable learning meaningful object representations in a video setting. This approach involves first training an image-based model using a pre-trained object detector, and then extending it with temporal modules for video processing. See Subsection 4.3.4 for more details. The downstream object tracking experiment (Subsection 4.4.2) shows that this protocol leads to better performance than training the model end-to-end, in which case it fails to focus on learning representations of people. By re-using a fixed pre-trained object detection model, RDIR can focus specifically on the regions containing objects of interest (in this case, people). Furthermore, the experiment explores the possibility of transferring knowledge from a large-scale dataset (MS COCO) into a new domain. Although the stage protocol enables adaptation, the quality of transferred representations is sensitive to domain-specific visual characteristics, which is shown by the drop in performance when the detector is transferred from another dataset to the object tracking task.

RQ 7: *How can detection-guided object representations be used to condition diffusion-based generative models for controllable image synthesis?*

In Chapter 5), a LDM-based method DetDiff infers detection-guided representations learned through a spatial grid-based attention mechanism as conditioning input of the Latent Diffusion Model (LDM). These representations are created by filtering multi-scale convolutional features based on object confidence predictions from a pre-trained YOLOv4 detector and then processed through encoder heads similar to SSDIR and RDIR. To disentangle spatial position from object appearance, DetDiff enriches each object's representation with positional encodings derived from predicted object centres. The conditioning follows the LDM approach, where the denoising network incorporates multi-head cross-attention to use auxiliary input in the reverse diffusion process. See Subsection 5.3.1 for more details.

RQ 8: *How can positional information from detection models be incorporated in a diffusion-based object-centric model to improve object-to-representation matching in generative applications and complex downstream tasks?*

In DetDiff (Chapter 5), positional guidance is integrated in two ways. First, the detected object centre coordinates, predicted by a pre-trained YOLOv4 detector, are used to compute 2D positional encodings, integrated into each object's representation; these representations are used to condition the denoising network in the Latent Diffusion Model setup, enabling the model to ground the reverse diffusion process with explicit object location guidance. See Subsection 5.3.1 for more details. Second, DetDiff uses a per-object pixel space loss based on detector-derived bounding boxes, guiding high-accuracy reconstructions of regions corresponding to detected entities (Subsection 5.3.2). This self-supervised loss improves object-to-representation alignment and supports the generation of images from interpretable and granular object representations.

This mechanism enhances object-to-representation alignment by improving disentanglement of object appearance and spatial location in the embeddings. As shown in Subsection 5.4.2, DetDiff provides a precise matching between scene entities and their inferred representations. The benefits of positional encodings and per-object loss are supported by the ablation study presented in Subsection 5.4.4, where removing detection guidance leads to degraded performance. However, due to the lack of explicit background modelling, background information is absorbed into object latents, leading to entanglement and lower reconstruction fidelity overall (Subsection 5.4.1). This highlights the importance of future work in decoupling background and foreground in inferred scene representation.

Future Work. Several promising research directions could be explored based on the findings presented in this thesis. Modelling background information explicitly as a separate latent variable, or employing a dedicated decoder could enhance the fidelity of the reconstructions and reduce the influence of the background on objects' representations. Similarly, shifting from the use of a pre-trained object detection network to internally learned or weakly supervised spatial cues might mitigate dataset-specific biases and improve domain generalisation. Extending the diffusion-based model into temporal domains follows the research direction in object-centric representation learning, and could lead to more stable object representations across video frames, enabling consistent temporal editing applications. In this setup, it could be useful to introduce explicit object relations modelling, enhancing the model's capacity to manage complex scenes with occlusions. Following the scene-mixture multi-object representation learning, an approach similar to detection guidance could be proposed, where bounding box-based attention could be replaced with instance segmentation, which performs better in the case of objects of highly irregular shapes. Finally, deploying object-centric representations directly on real robotic systems would allow for validating the practical applicability of these models in complex, real-world problems.

Appendix A

Selected scientific achievements

A.1 List of publications

- [210] **P. Zieliński** and U. Markowska-Kaczmar, *3D robotic navigation using a vision-based deep reinforcement learning model*, Applied Soft Computing, vol. 110, art. 107602, pp. 1-17, 2021, MNiSW points: 200.

  Citations (as of 09.10.2025): Scopus – 35, Web of Science – 23, Google Scholar – 47.

  Contributions: Conceptualization, Data Curation, Formal Analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – Original Draft Preparation, Writing – Review & Editing

- [208] **P. Zieliński** and T. Kajdanowicz, *Learning scale-invariant object representations with a single-shot convolutional generative model*, Computational Science - ICCS 2022, 22nd International Conference, London, UK, 21-23 June, 2022, Proceedings Part III, Springer, 2022, Lecture Notes in Computer Science, vol. 13352, pp. 613-626, MNiSW points: 140.

  Citations (as of 24.06.2025): Scopus – 1, Web of Science – 1, Google Scholar – 1.

  Contributions: Conceptualization, Data Curation, Formal Analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – Original Draft Preparation, Writing – Review & Editing

- [209] **P. Zieliński** and T. Kajdanowicz, *RDIR: capturing temporally-invariant representations of multiple objects in videos*, IEEE/CVF Winter Conference on Applications of Computer Vision Workshops, WACVW 2024 - Workshops, Waikoloa, HI, USA, 1-6 January, 2024, MNiSW points: 140.

Citations (as of 24.06.2025): Scopus – 0, Web of Science – 0, Google Scholar – 0.

Contributions: Conceptualization, Data Curation, Formal Analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – Original Draft Preparation, Writing – Review & Editing

A.2 List of conference speeches

- **P. Zieliński**, T. Kajdanowicz, *Learning scale-invariant object representations with a single-shot convolutional generative model*, 22nd International Conference on Computational Science, London, UK, 21-23 June 2022.
- **P. Zieliński**, T. Kajdanowicz, *RDIR: capturing temporally-invariant representations of multiple objects in videos*, IEEE/CVF Winter Conference on Applications of Computer Vision Workshops, Waikoloa, HI, USA, 1-6 January, 2024.

List of Figures

List of Tables

Bibliography

[1] Fatin H. Ajeil, Ibraheem Kasim Ibraheem, Mouayad A. Sahib, and Amjad J. Humaidi. Multi-objective path planning of an autonomous mobile robot using hybrid PSO-MFB optimization algorithm. *Applied Soft Computing*, 89:106076, April 2020.

[2] Görkay Aydemir, Weidi Xie, and Fatma Güney. Self-supervised Object-Centric Learning for Videos. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

[3] Mohammad Babaeizadeh, Chelsea Finn, Dumitru Erhan, Roy H. Campbell, and Sergey Levine. Stochastic Variational Video Prediction. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[4] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. BEiT: BERT Pre-Training of Image Transformers. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[5] Zhipeng Bao, Pavel Tokmakov, Allan Jabri, Yu-Xiong Wang, Adrien Gaidon, and Martial Hebert. Discovering Objects that Can Move. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 11779–11788. IEEE, 2022.

[6] Zhipeng Bao, Pavel Tokmakov, Yu-Xiong Wang, Adrien Gaidon, and Martial Hebert. Object Discovery from Motion-Guided Tokens. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pages 22972–22981. IEEE, 2023.

[7] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. SURF: Speeded Up Robust Features. In Aleš Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision – ECCV 2006*, pages 404–417, Berlin, Heidelberg, 2006. Springer.

[8] Y. Bengio, A. Courville, and P. Vincent. Representation Learning: A Review and New Perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, August 2013.

[9] Keni Bernardin and Rainer Stiefelhagen. Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics. *EURASIP J. Image Video Process.*, 2008, 2008.

[10] Beril Besbinar and Pascal Frossard. Self-Supervision By Prediction For Object Discovery In Videos. In *2021 IEEE International Conference on Image Processing, ICIP 2021, Anchorage, AK, USA, September 19-22, 2021*, pages 1509–1513. IEEE, 2021.

[11] Andreas Blattmann, Robin Rombach, Huan Ling, Tim Dockhorn, Seung Wook Kim, Sanja Fidler, and Karsten Kreis. Align Your Latents: High-Resolution Video Synthesis with Latent Diffusion Models. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 22563–22575, Vancouver, BC, Canada, June 2023. IEEE.

[12] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. YOLOv4: Optimal Speed and Accuracy of Object Detection, April 2020. arXiv:2004.10934 [cs].

[13] Andrew Brock, Jeff Donahue, and Karen 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.

[14] Christopher P. Burgess, Loic Matthey, Nicholas Watters, Rishabh Kabra, Irina Higgins, Matt Botvinick, and Alexander Lerchner. MONet: Unsupervised Scene Decomposition and Representation, January 2019. arXiv:1901.11390 [cs].

[15] Xiang Cao, Changyin Sun, and Mingzhong Yan. Target Search Control of AUV in Underwater Environment With Deep Reinforcement Learning. *IEEE Access*, 7:96549–96559, 2019.

[16] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-End Object Detection with Transformers. In *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I*, pages 213–229, Berlin, Heidelberg, August 2020. Springer-Verlag.

[17] Ignacio Carlucho, Mariano De Paula, Sen Wang, Bruno V. Menna, Yvan R. Petillot, and Gerardo G. Acosta. AUV Position Tracking Control Using End-to-End Deep Reinforcement Learning. In *OCEANS 2018 MTS/IEEE Charleston*, pages 1–8, October 2018. ISSN: 0197-7385.

[18] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised Learning of Visual Features by Contrasting Cluster Assignments. In *Advances in Neural Information Processing Systems*, volume 33, pages 9912–9924. Curran Associates, Inc., 2020.

[19] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jegou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging Properties in Self-Supervised Vision Transformers. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9630–9640, October 2021. ISSN: 2380-7504.

[20] João Carreira and Andrew Zisserman. Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 4724–4733. IEEE Computer Society, 2017.

[21] Chang Chen, Fei Deng, and Sungjin Ahn. ROOTS: Object-Centric Representation and Rendering of 3D Scenes. *J. Mach. Learn. Res.*, 22:259:1–259:36, 2021.

[22] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A Simple Framework for Contrastive Learning of Visual Representations. In *Proceedings of the 37th International Conference on Machine Learning*, pages 1597–1607. PMLR, November 2020. ISSN: 2640-3498.

[23] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[24] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved Baselines with Momentum Contrastive Learning, March 2020. arXiv:2003.04297 [cs].

[25] Xinlei Chen and Kaiming He. Exploring Simple Siamese Representation Learning. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15745–15753, June 2021. ISSN: 2575-7075.

[26] Xinlei Chen, Saining Xie, and Kaiming He. An Empirical Study of Training Self-Supervised Vision Transformers. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9620–9629, October 2021. ISSN: 2380-7504.

[27] Rewon Child. Very Deep VAEs Generalize Autoregressive Models and Can Outperform Them on Images. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021.* OpenReview.net, 2021.

[28] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics.

[29] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 539–546 vol. 1, June 2005. ISSN: 1063-6919.

[30] Fernando Cosío and Miguel Padilla. Autonomous robot navigation using adaptive potential fields. *Mathematical and Computer Modelling*, 40:1141–1156, November 2004.

[31] Eric Crawford and Joelle Pineau. Spatially Invariant Unsupervised Object Detection with Convolutional Neural Networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):3412–3420, July 2019. Number: 01.

[32] Eric Crawford and Joelle Pineau. Exploiting Spatial Invariance for Scalable Unsupervised Object Tracking. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):3684–3692, April 2020. Number: 04.

[33] Antonia Creswell, Rishabh Kabra, Chris Burgess, and Murray Shanahan. Unsupervised Object-Based Transition Models For 3D Partially Observable Environments. In *Advances in Neural Information Processing Systems*, volume 34, pages 27344–27355. Curran Associates, Inc., 2021.

[34] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893 vol. 1, 2005.

[35] Fei Deng, Zhuo Zhi, Donghun Lee, and Sungjin Ahn. Generative Scene Graph Networks. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[36] Emily Denton and Rob Fergus. Stochastic Video Generation with a Learned Prior. In *Proceedings of the 35th International Conference on Machine Learning*, pages 1174–1183. PMLR, July 2018. ISSN: 2640-3498.

[37] Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, Yuhuai Wu, and Peter Zhokhov. OpenAI Baselines, 2017. Publication Title: GitHub repository.

[38] Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: Non-linear Independent Components Estimation. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings*, 2015.

[39] 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.

[40] Carl Doersch, Abhinav Gupta, and Alexei A. Efros. Unsupervised Visual Representation Learning by Context Prediction. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1422–1430, December 2015. ISSN: 2380-7504.

[41] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial Feature Learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[42] Jeff Donahue and Karen Simonyan. Large Scale Adversarial Representation Learning. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[43] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth

16x16 Words: Transformers for Image Recognition at Scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[44] Alexey Dosovitskiy, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. Discriminative Unsupervised Feature Learning with Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[45] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Alex Lamb, Martín Arjovsky, Olivier Mastropietro, and Aaron C. Courville. Adversarially Learned Inference. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[46] Bartłomiej Józef Dzieńkowski, Christopher Strode, and Urszula Markowska-Kaczmar. Employing game theory and computational intelligence to find the optimal strategy of an Autonomous Underwater Vehicle against a submarine. In *2016 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pages 31–40, September 2016.

[47] Sebastien Ehrhardt, Oliver Groth, Aron Monszpart, Martin Engelcke, Ingmar Posner, Niloy Mitra, and Andrea Vedaldi. RELATE: Physically Plausible Multi-Object Scene Synthesis Using Structured Latent Spaces. In *Advances in Neural Information Processing Systems*, volume 33, pages 11202–11213. Curran Associates, Inc., 2020.

[48] Gamaleldin F. Elsayed, Aravindh Mahendran, Sjoerd van Steenkiste, Klaus Greff, Michael C. Mozer, and Thomas Kipf. SAVi++: towards end-to-end object-centric learning from real-world videos. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, pages 28940–28954, Red Hook, NY, USA, November 2022. Curran Associates Inc.

[49] Martin Engelcke, Adam R. Kosiorek, Oiwi Parker Jones, and Ingmar Posner. GENESIS: Generative Scene Inference and Sampling with Object-Centric Latent Representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[50] Martin Engelcke, Oiwi Parker Jones, and Ingmar Posner. GENESIS-V2: Inferring Unordered Object Representations without Iterative Refinement. In *Advances in Neural Information Processing Systems*, volume 34, pages 8085–8094. Curran Associates, Inc., 2021.

[51] Dave Epstein, Allan Jabri, Ben Poole, Alexei A. Efros, and Aleksander Holynski. Diffusion Self-Guidance for Controllable Image Generation. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, pages 16222–16239, Red Hook, NY, USA, December 2023. Curran Associates Inc.

[52] Russell A. Epstein and Chris I. Baker. Scene Perception in the Human Brain. *Annual Review of Vision Science*, 5:373–397, September 2019.

[53] S. M. Ali Eslami, Nicolas Heess, Theophane Weber, Yuval Tassa, David Szepesvari, koray kavukcuoglu, and Geoffrey E Hinton. Attend, Infer, Repeat: Fast Scene Understanding with Generative Models. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[54] Yuxin Fang, Wen Wang, Binhui Xie, Quan Sun, Ledell Wu, Xinggang Wang, Tiejun Huang, Xinlong Wang, and Yue Cao. EVA: Exploring the Limits of Masked Visual Representation Learning at Scale. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 19358–19369. IEEE Computer Society, June 2023.

[55] Pedro F. Felzenszwalb, Ross B. Girshick, David McAllester, and Deva Ramanan. Object Detection with Discriminatively Trained Part-Based Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, September 2010.

[56] Basura Fernando, Hakan Bilen, Efstratios Gavves, and Stephen Gould. Self-Supervised Video Representation Learning with Odd-One-Out Networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5729–5738. IEEE Computer Society, July 2017. ISSN: 1063-6919.

[57] Partha Ghosh, Mehdi S. M. Sajjadi, Antonio Vergari, Michael J. Black, and Bernhard Schölkopf. From Variational to Deterministic Autoencoders. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[58] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised Representation Learning by Predicting Image Rotations. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[59] Rohit Girdhar and Deva Ramanan. CATER: A diagnostic dataset for Compositional Actions & TEmporal Reasoning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[60] Ross Girshick. Fast R-CNN. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, December 2015. ISSN: 2380-7504.

[61] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*, pages 580–587. IEEE Computer Society, 2014.

[62] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[63] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the V in VQA Matter: Elevating the Role of Image Understanding in Visual Question Answering. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6325–6334, July 2017. ISSN: 1063-6919.

[64] Klaus Greff, Francois Belletti, Lucas Beyer, Carl Doersch, Yilun Du, Daniel Duckworth, David J Fleet, Dan Gnanapragasam, Florian Golemo, Charles Herrmann, Thomas Kipf, Abhijit Kundu, Dmitry Lagun, Issam Laradji, Hsueh-Ti Liu, Henning Meyer, Yishu Miao, Derek Nowrouzezahrai, Cengiz Oztireli, Etienne Pot, Noha Radwan, Daniel Rebain, Sara Sabour, Mehdi S. M. Sajjadi, Matan Sela, Vincent Sitzmann, Austin Stone, Deqing Sun, Suhani Vora, Ziyu Wang, Tianhao Wu, Kwang Moo Yi, Fangcheng Zhong, and Andrea Tagliasacchi. Kubric: A scalable dataset generator. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3739–3751, June 2022. ISSN: 2575-7075.

[65] Klaus Greff, Raphaël Lopez Kaufman, Rishabh Kabra, Nick Watters, Christopher Burgess, Daniel Zoran, Loic Matthey, Matthew Botvinick, and Alexander Lerchner. Multi-Object Representation Learning with Iterative Variational Inference. In *Proceedings of the 36th International Conference on Machine Learning*, pages 2424–2433. PMLR, May 2019. ISSN: 2640-3498.

[66] Klaus Greff, Sjoerd van Steenkiste, and Jürgen Schmidhuber. Neural Expectation Maximization. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[67] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, Bilal Piot, koray kavukcuoglu, Remi Munos, and Michal Valko. Bootstrap Your Own Latent - A New Approach to Self-Supervised Learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 21271–21284. Curran Associates, Inc., 2020.

[68] Chunzhi Gu, Haoran Xie, Xuequan Lu, and Chao Zhang. CGMVAE: Coupling GMM Prior and GMM Estimator for Unsupervised Clustering and Disentanglement. *IEEE Access*, 9:65140–65149, 2021.

[69] Yuwei Guo, Ceyuan Yang, Anyi Rao, Zhengyang Liang, Yaohui Wang, Yu Qiao, Maneesh Agrawala, Dahua Lin, and Bo Dai. AnimateDiff: Animate Your Personalized Text-to-Image Diffusion Models without Specific Tuning. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

[70] Shir Gur, Sagie Benaim, and Lior Wolf. Hierarchical Patch VAE-GAN: Generating Diverse Videos from a Single Sample. In *Advances in Neural Information Processing Systems*, volume 33, pages 16761–16772. Curran Associates, Inc., 2020.

[71] Akshay Kumar Guruji, Himansh Agarwal, and D. K. Parsediya. Time-efficient A* Algorithm for Robot Path Planning. *Procedia Technology*, 23:144–149, January 2016.

[72] Tengda Han, Weidi Xie, and Andrew Zisserman. Video Representation Learning by Dense Predictive Coding. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 1483–1492, October 2019. ISSN: 2473-9944.

[73] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked Autoencoders Are Scalable Vision Learners. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15979–15988, June 2022. ISSN: 2575-7075.

[74] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum Contrast for Unsupervised Visual Representation Learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9726–9735, June 2020. ISSN: 2575-7075.

[75] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, October 2017. ISSN: 2380-7504.

[76] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, June 2016. ISSN: 1063-6919.

[77] Paul Henderson and Christoph H Lampert. Unsupervised object-centric video generation and decomposition in 3D. In *Advances in Neural Information Processing Systems*, volume 33, pages 3106–3117. Curran Associates, Inc., 2020.

[78] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[79] Irina Higgins, Loïc Matthey, Arka Pal, Christopher P. Burgess, Xavier Glorot, Matthew M. Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[80] Ashley Hill, Antonin Raffin, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, Rene Traore, Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, and Yuhuai Wu. Stable Baselines, 2018. Publication Title: GitHub repository.

[81] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models. In *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc., 2020.

[82] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J. Fleet. Video diffusion models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, pages 8633–8646, Red Hook, NY, USA, November 2022. Curran Associates Inc.

[83] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, November 1997.

[84] Drew A. Hudson, Daniel Zoran, Mateusz Malinowski, Andrew K. Lampinen, Andrew Jaegle, James L. McClelland, Loic Matthey, Felix Hill, and Alexander Lerchner. SODA: Bottleneck Diffusion Models for Representation Learning. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 23115–23127, June 2024. ISSN: 2575-7075.

[85] Yujia Huo, Yiping Li, and Xisheng Feng. Model-Free Recurrent Reinforcement Learning for AUV Horizontal Control. *IOP Conference Series: Materials Science and Engineering*, 428(1):012063, September 2018. Publisher: IOP Publishing.

[86] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. Let there be color!: joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification. *ACM Trans. Graph.*, 35(4):110:1–110:11, July 2016.

[87] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and koray kavukcuoglu. Spatial Transformer Networks. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[88] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. Scaling Up Visual and Vision-Language Representation Learning With Noisy Text Supervision. In *Proceedings of the 38th International Conference on Machine Learning*, pages 4904–4916. PMLR, July 2021. ISSN: 2640-3498.

[89] Jindong Jiang and Sungjin Ahn. Generative Neurosymbolic Machines. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[90] Jindong Jiang, Fei Deng, Gautam Singh, and Sungjin Ahn. Object-Centric Slot Diffusion. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

[91] Jindong Jiang, Fei Deng, Gautam Singh, Minseung Lee, and Sungjin Ahn. Slot State Space Models. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

[92] Jindong Jiang, Sepehr Janghorbani, Gerard de Melo, and Sungjin Ahn. SCALOR: Generative World Models with Scalable Object Representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[93] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross Girshick. CLEVR: A Diagnostic Dataset for Compositional Language and Elementary Visual Reasoning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1988–1997, July 2017. ISSN: 1063-6919.

[94] Arthur Juliani, Vincent-Pierre Berges, Ervin Teng, Andrew Cohen, Jonathan Harper, Chris Elion, Chris Goy, Yuan Gao, Hunter Henry, Marwan Mattar, and Danny Lange. Unity: A General Platform for Intelligent Agents, May 2020. arXiv:1809.02627 [cs].

[95] Rishabh Kabra, Daniel Zoran, Goker Erdogan, Loic Matthey, Antonia Creswell, Matt Botvinick, Alexander Lerchner, and Chris Burgess. SIMONe: View-Invariant, Temporally-Abstracted Object Representations via Unsupervised Video Decomposition. In *Advances in Neural Information Processing Systems*, volume 34, pages 20146–20159. Curran Associates, Inc., 2021.

[96] Ioannis Kakogeorgiou, Spyros Gidaris, Konstantinos Karantzalos, and Nikos Komodakis. SPOT: Self-Training with Patch-Order Permutation for Object-Centric Learning with Autoregressive Transformers. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pages 22776–22786. IEEE, 2024.

[97] Hwan Il Kang, Byunghee Lee, and Kabil Kim. Path Planning Algorithm Using the Particle Swarm Optimization and the Improved Dijkstra Algorithm. In *2008 IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application*, volume 2, pages 1002–1004, December 2008.

[98] Laurynas Karazija, Iro Laina, and Christian Rupprecht. ClevrTex: A Texture-Rich Benchmark for Unsupervised Multi-Object Segmentation. In Joaquin Vanschoren and Sai-Kit Yeung, editors, *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, 2021.

[99] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive Growing of GANs for Improved Quality, Stability, and Variation. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[100] Tero Karras, Samuli Laine, and Timo Aila. A Style-Based Generator Architecture for Generative Adversarial Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(12):4217–4228, December 2021.

[101] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[102] Durk P Kingma and Prafulla Dhariwal. Glow: Generative Flow with Invertible 1x1 Convolutions. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[103] Thomas Kipf, Gamaleldin Fathy Elsayed, Aravindh Mahendran, Austin Stone, Sara Sabour, Georg Heigold, Rico Jonschkowski, Alexey Dosovitskiy, and Klaus Greff. Conditional Object-Centric Learning from Video. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[104] Adam Kosiorek, Hyunjik Kim, Yee Whye Teh, and Ingmar Posner. Sequential Attend, Infer, Repeat: Generative Modelling of Moving Objects. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[105] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[106] Jonáš Kulhánek, Erik Derner, Tim de Bruin, and Robert Babuška. Vision-based Navigation Using Deep Reinforcement Learning. In *2019 European Conference on Mobile Robots (ECMR)*, pages 1–8, September 2019.

[107] Tejas D. Kulkarni, Karthik R. Narasimhan, Ardavan Saeedi, and Joshua B. Tenenbaum. Hierarchical deep reinforcement learning: integrating temporal abstraction and intrinsic motivation. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pages 3682–3690, Red Hook, NY, USA, December 2016. Curran Associates Inc.

[108] Manoj Kumar, Mohammad Babaeizadeh, Dumitru Erhan, Chelsea Finn, Sergey Levine, Laurent Dinh, and Durk Kingma. VideoFlow: A Conditional Flow-Based Model for Stochastic Video Generation. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[109] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 1558–1566. PMLR, June 2016. ISSN: 1938-7228.

[110] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Learning Representations for Automatic Colorization. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 577–593, Cham, 2016. Springer International Publishing.

[111] Laura Leal-Taixé, Anton Milan, Ian Reid, Stefan Roth, and Konrad Schindler. MOTChallenge 2015: Towards a Benchmark for Multi-Target Tracking, April 2015. arXiv:1504.01942 [cs].

[112] Karel Lenc and Andrea Vedaldi. Understanding Image Representations by Measuring Their Equivariance and Equivalence. *International Journal of Computer Vision*, 127(5):456–476, May 2019.

[113] Hong Li, Mingyong Liu, and Kun Liu. Bio-inspired geomagnetic navigation method for autonomous underwater vehicle. *Journal of Systems Engineering and Electronics*, 28(6):1203–1209, December 2017.

[114] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models. In *Proceedings of the 40th International Conference on Machine Learning*, pages 19730–19742. PMLR, July 2023. ISSN: 2640-3498.

[115] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. BLIP: Bootstrapping Language-Image Pre-training for Unified Vision-Language Understanding and Generation. In *Proceedings of the 39th International Conference on Machine Learning*, pages 12888–12900. PMLR, June 2022. ISSN: 2640-3498.

[116] Xiaodan Liang, Lisa Lee, Wei Dai, and Eric P. Xing. Dual Motion GAN for Future-Flow Embedded Video Prediction. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 1762–1770. IEEE Computer Society, 2017.

[117] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[118] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal Loss for Dense Object Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(2):318–327, February 2020.

[119] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common Objects in Context. In David J. Fleet, Tomás Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V*, volume 8693 of *Lecture Notes in Computer Science*, pages 740–755. Springer, 2014.

[120] Zhixuan Lin, Yi-Fu Wu, Skand Vishwanath Peri, Weihao Sun, Gautam Singh, Fei Deng, Jindong Jiang, and Sungjin Ahn. SPACE: Unsupervised Object-Oriented Scene Representation via Spatial Attention and Decomposition. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[121] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path Aggregation Network for Instance Segmentation. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 8759–8768. Computer Vision Foundation / IEEE Computer Society, 2018.

[122] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single Shot MultiBox Detector. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part I*, volume 9905 of *Lecture Notes in Computer Science*, pages 21–37. Springer, 2016.

[123] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep Learning Face Attributes in the Wild. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 3730–3738. IEEE Computer Society, 2015.

[124] Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-Centric Learning with Slot Attention. In *Advances in Neural Information Processing Systems*, volume 33, pages 11525–11538. Curran Associates, Inc., 2020.

[125] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004.

[126] Jonathon Luiten and Arne Hoffhues. TrackEval, 2020.

[127] Jonathon Luiten, Aljosa Osep, Patrick Dendorfer, Philip Torr, Andreas Geiger, Laura Leal-Taixé, and Bastian Leibe. HOTA: A Higher Order Metric for Evaluating Multi-object Tracking. *International Journal of Computer Vision*, 129(2):548–578, February 2021.

[128] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial Autoencoders, May 2016. arXiv:1511.05644 [cs].

[129] Amir Mohammad Karimi Mamaghan, Samuele Papa, Karl Henrik Johansson, Stefan Bauer, and Andrea Dittadi. Exploring the Effectiveness of Object-Centric Representations in Visual Question Answering: Comparative Insights with Foundation Models. In *The Thirteenth International Conference on Learning Representations*, 2025.

[130] Ishan Misra, C. Lawrence Zitnick, and Martial Hebert. Shuffle and Learn: Unsupervised Learning Using Temporal Order Verification. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part I*, volume 9905 of *Lecture Notes in Computer Science*, pages 527–544. Springer, 2016.

[131] Sarthak Mittal, Korbinian Abstreiter, Stefan Bauer, Bernhard Schölkopf, and Arash Mehrjou. Diffusion Based Representation Learning. In *Proceedings of the 40th International Conference on Machine Learning*, pages 24963–24982. PMLR, July 2023. ISSN: 2640-3498.

[132] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous Methods for Deep Reinforcement Learning. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 1928–1937. PMLR, June 2016. ISSN: 1938-7228.

[133] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015. Publisher: Nature Publishing Group.

[134] Michael Niemeyer and Andreas Geiger. GIRAFFE: Representing Scenes As Compositional Generative Neural Feature Fields. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 11453–11464. Computer Vision Foundation / IEEE, 2021.

[135] Mehdi Noroozi and Paolo Favaro. Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VI*, volume 9910 of *Lecture Notes in Computer Science*, pages 69–84. Springer, 2016.

[136] Mehdi Noroozi, Hamed Pirsiavash, and Paolo Favaro. Representation Learning by Learning to Count. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 5899–5907. IEEE Computer Society, 2017.

[137] Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural Discrete Representation Learning. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 6306–6315, 2017.

[138] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jégou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning Robust Visual Features without Supervision. *Trans. Mach. Learn. Res.*, 2024, 2024.

[139] Tian Pan, Yibing Song, Tianyu Yang, Wenhao Jiang, and Wei Liu. VideoMoCo: Contrastive Video Representation Learning With Temporally Adversarial Examples. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 11205–11214. Computer Vision Foundation / IEEE, 2021.

[140] Deepak Pathak, Philipp Krähenbühl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. Context Encoders: Feature Learning by Inpainting. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 2536–2544. IEEE Computer Society, 2016.

[141] Rui Qian, Shuangrui Ding, Xian Liu, and Dahua Lin. Semantics Meets Temporal Correspondence: Self-supervised Object-centric Learning in Videos. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, pages 16629–16641. IEEE, 2023.

[142] Rui Qian, Tianjian Meng, Boqing Gong, Ming-Hsuan Yang, Huisheng Wang, Serge J. Belongie, and Yin Cui. Spatiotemporal Contrastive Video Representation Learning. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 6964–6974. Computer Vision Foundation / IEEE, 2021.

[143] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning Transferable Visual Models From Natural Language Supervision. In *Proceedings of the 38th International Conference on Machine Learning*, pages 8748–8763. PMLR, July 2021. ISSN: 2640-3498.

[144] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2020.

[145] William M. Rand. Objective Criteria for the Evaluation of Clustering Methods. *Journal of the American Statistical Association*, 66(336):846–850, December 1971. Publisher: ASA Website _eprint: https://www.tandfonline.com/doi/pdf/10.1080/01621459.1971.10482356.

[146] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 779–788. IEEE Computer Society, 2016.

[147] Joseph Redmon and Ali Farhadi. YOLO9000: Better, Faster, Stronger. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 6517–6525. IEEE Computer Society, 2017.

[148] Joseph Redmon and Ali Farhadi. YOLOv3: An Incremental Improvement, April 2018. arXiv:1804.02767 [cs].

[149] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 91–99, 2015.

[150] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In *Proceedings of the 31st International Conference on Machine Learning*, pages 1278–1286. PMLR, June 2014. ISSN: 1938-7228.

[151] Ergys Ristani, Francesco Solera, Roger S. Zou, Rita Cucchiara, and Carlo Tomasi. Performance Measures and a Data Set for Multi-target, Multi-camera Tracking. In Gang Hua and Hervé Jégou, editors, *Computer Vision - ECCV 2016 Workshops - Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part II*, volume 9914 of *Lecture Notes in Computer Science*, pages 17–35, 2016.

[152] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-Resolution Image Synthesis with Latent Diffusion Models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 10674–10685. IEEE, 2022.

[153] Florian Schroff, Dmitry Kalenichenko, and James Philbin. FaceNet: A unified embedding for face recognition and clustering. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 815–823. IEEE Computer Society, 2015.

[154] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms, August 2017. arXiv:1707.06347 [cs].

[155] Maximilian Seitzer, Max Horn, Andrii Zadaianchuk, Dominik Zietlow, Tianjun Xiao, Carl-Johann Simon-Gabriel, Tong He, Zheng Zhang, Bernhard Schölkopf, Thomas Brox, and Francesco Locatello. Bridging the Gap to Real-World Object-Centric Learning. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[156] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[157] Uriel Singer, Adam Polyak, Thomas Hayes, Xi Yin, Jie An, Songyang Zhang, Qiyuan Hu, Harry Yang, Oron Ashual, Oran Gafni, Devi Parikh, Sonal Gupta, and Yaniv Taigman. Make-A-Video: Text-to-Video Generation without Text-Video Data. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[158] Gautam Singh, Fei Deng, and Sungjin Ahn. Illiterate DALL-E Learns to Compose. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[159] Gautam Singh, Yeongbin Kim, and Sungjin Ahn. Neural Systematic Binder. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[160] Gautam Singh, Yi-Fu Wu, and Sungjin Ahn. Simple Unsupervised Object-Centric Learning for Complex and Naturalistic Videos. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *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.

[161] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep Unsupervised Learning using Nonequilibrium Thermodynamics. In Francis R. Bach and David M. Blei, editors, *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*, pages 2256–2265. JMLR.org, 2015.

[162] Maksim Sporyshev and Alexander Scherbatyuk. Reinforcement learning approach for cooperative AUVs in underwater surveillance operations. In *2019 IEEE Underwater Technology (UT)*, pages 1–4, April 2019. ISSN: 2573-3796.

[163] Sjoerd van Steenkiste, Michael Chang, Klaus Greff, and Jürgen Schmidhuber. Relational Neural Expectation Maximization: Unsupervised Discovery of Objects and their Interactions. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[164] Yushan Sun, Junhan Cheng, Guocheng Zhang, and Hao Xu. Mapless Motion Planning System for an Autonomous Underwater Vehicle Using Policy Gradient-based Deep Reinforcement Learning. *Journal of Intelligent & Robotic Systems*, 96(3):591–601, December 2019.

[165] Yushan Sun, Xiangrui Ran, Guocheng Zhang, Hao Xu, and Xiangbin Wang. AUV 3D Path Planning Based on the Improved Hierarchical Deep Q Network. *Journal of Marine Science and Engineering*, 8(2):145, February 2020. Number: 2 Publisher: Multidisciplinary Digital Publishing Institute.

[166] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning - an introduction, 2nd Edition*. MIT Press, 2018.

[167] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 1–9. IEEE Computer Society, 2015.

[168] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the Inception Architecture for Computer Vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 2818–2826. IEEE Computer Society, 2016.

[169] Luming Tang, Menglin Jia, Qianqian Wang, Cheng Perng Phoo, and Bharath Hariharan. Emergent Correspondence from Image Diffusion. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

[170] Zhan Tong, Yibing Song, Jue Wang, and Limin Wang. VideoMAE: Masked Autoencoders are Data-Efficient Learners for Self-Supervised Video Pre-Training. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *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.

[171] Manuel Traub, Sebastian Otte, Tobias Menge, Matthias Karlbauer, Jannik Thümmel, and Martin V. Butz. Learning What and Where: Disentangling Location and Identity Tracking Without Supervision. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[172] Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. MoCoGAN: Decomposing Motion and Content for Video Generation. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 1526–1535. Computer Vision Foundation / IEEE Computer Society, 2018.

[173] Jasper R. R. Uijlings, Koen E. A. van de Sande, Theo Gevers, and Arnold W. M. Smeulders. Selective Search for Object Recognition. *Int. J. Comput. Vis.*, 104(2):154–171, 2013.

[174] Thomas Unterthiner, Sjoerd van Steenkiste, Karol Kurach, Raphaël Marinier, Marcin Michalski, and Sylvain Gelly. FVD: A new Metric for Video Generation. In *Deep Generative Models for Highly Structured Data, ICLR 2019 Workshop, New Orleans, Louisiana, United States, May 6, 2019*. OpenReview.net, 2019.

[175] Arash Vahdat and Jan Kautz. NVAE: A Deep Hierarchical Variational Autoencoder. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[176] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017.

[177] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In William W. Cohen, Andrew McCallum, and Sam T. Roweis, editors, *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*, volume 307 of *ACM International Conference Proceeding Series*, pages 1096–1103. ACM, 2008.

[178] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pages I–I, December 2001. ISSN: 1063-6919.

[179] Carl Vondrick, Abhinav Shrivastava, Alireza Fathi, Sergio Guadarrama, and Kevin Murphy. Tracking Emerges by Colorizing Videos. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XIII*, volume 11217 of *Lecture Notes in Computer Science*, pages 402–419. Springer, 2018.

[180] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Scaled-YOLOv4: Scaling Cross Stage Partial Network. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 13029–13038. Computer Vision Foundation / IEEE, 2021.

[181] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pages 7464–7475. IEEE, 2023.

[182] Limin Wang, Bingkun Huang, Zhiyu Zhao, Zhan Tong, Yinan He, Yi Wang, Yali Wang, and Yu Qiao. VideoMAE V2: Scaling Video Masked Autoencoders with Dual Masking. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pages 14549–14560. IEEE, 2023.

[183] Zelun Wang and Jyh-Charn Liu. Translating math formula images to LaTeX sequences using deep neural networks with sequence-level training. *Int. J. Document Anal. Recognit.*, 24(1):63–75, 2021.

[184] C.W. Warren. A technique for autonomous underwater vehicle route planning. *IEEE Journal of Oceanic Engineering*, 15(3):199–204, July 1990.

[185] Donglai Wei, Joseph J. Lim, Andrew Zisserman, and William T. Freeman. Learning and Using the Arrow of Time. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 8052–8060. Computer Vision Foundation / IEEE Computer Society, 2018.

[186] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *2017 IEEE International Conference on Image Processing, ICIP 2017, Beijing, China, September 17-20, 2017*, pages 3645–3649. IEEE, 2017.

[187] Keyu Wu, Mahdi Abolfazli Esfahani, Shenghai Yuan, and Han Wang. Depth-based Obstacle Avoidance through Deep Reinforcement Learning. In *Proceedings of the 5th International Conference on Mechatronics and Robotics Engineering*, ICMRE'19, pages 102–106, New York, NY, USA, February 2019. Association for Computing Machinery.

[188] Ziyi Wu, Jingyu Hu, Wuyue Lu, Igor Gilitschenski, and Animesh Garg. SlotDiffusion: Object-Centric Generative Modeling with Diffusion Models. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

[189] Jinheng Xie, Yuexiang Li, Yawen Huang, Haozhe Liu, Wentian Zhang, Yefeng Zheng, and Mike Zheng Shou. BoxDiff: Text-to-Image Synthesis with Training-Free Box-Constrained Diffusion. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, pages 7418–7427. IEEE, 2023.

[190] Linhai Xie, Sen Wang, Andrew Markham, and Niki Trigoni. Towards Monocular Vision based Obstacle Avoidance through Deep Reinforcement Learning, June 2017. arXiv:1706.09829 [cs].

[191] Zhenda Xie, Zheng Zhang, Yue Cao, Yutong Lin, Jianmin Bao, Zhuliang Yao, Qi Dai, and Han Hu. SimMIM: a Simple Framework for Masked Image Modeling. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9643–9653, June 2022. ISSN: 2575-7075.

[192] Chao Yan, Xiaojia Xiang, and Chang Wang. Towards Real-Time Path Planning through Deep Reinforcement Learning for a UAV in Dynamic Environments. *Journal of Intelligent & Robotic Systems*, 98(2):297–309, May 2020.

[193] Shuo Yang, Ping Luo, Chen Change Loy, and Xiaoou Tang. WIDER FACE: A Face Detection Benchmark. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 5525–5533. IEEE Computer Society, 2016.

[194] Xingyi Yang and Xinchao Wang. Diffusion Model as Representation Learner. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, pages 18892–18903. IEEE, 2023.

[195] Kexin Yi, Chuang Gan, Yunzhu Li, Pushmeet Kohli, Jiajun Wu, Antonio Torralba, and Joshua B. Tenenbaum. CLEVRER: Collision Events for Video Representation and Reasoning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[196] Shixun You, Ming Diao, Lipeng Gao, Fulong Zhang, and Huan Wang. Target tracking strategy using deep deterministic policy gradient. *Applied Soft Computing*, 95:106490, October 2020.

[197] Polina Zablotskaia, Edoardo A. Dominici, Leonid Sigal, and Andreas M. Lehrmann. PROVIDE: a probabilistic framework for unsupervised video decomposition. In *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, pages 2019–2028. PMLR, December 2021. ISSN: 2640-3498.

[198] Andrii Zadaianchuk, Maximilian Seitzer, and Georg Martius. Object-Centric Learning for Real-World Videos by Predicting Temporal Feature Similarities. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

[199] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stephane Deny. Barlow Twins: Self-Supervised Learning via Redundancy Reduction. In *Proceedings of the 38th International Conference on Machine Learning*, pages 12310–12320. PMLR, July 2021. ISSN: 2640-3498.

[200] Fangyi Zhang, Jürgen Leitner, Michael Milford, Ben Upcroft, and Peter Corke. Towards Vision-Based Deep Reinforcement Learning for Robotic Motion Control, November 2015. arXiv:1511.03791 [cs].

[201] Hao Zhang, Feng Li, Shilong Liu, Lei Zhang, Hang Su, Jun Zhu, Lionel M. Ni, and Heung-Yeung Shum. DINO: DETR with Improved DeNoising Anchor Boxes for End-to-End Object Detection. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[202] Richard Zhang, Phillip Isola, and Alexei A. Efros. Colorful Image Colorization. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III*, volume 9907 of *Lecture Notes in Computer Science*, pages 649–666. Springer, 2016.

[203] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 586–595. Computer Vision Foundation / IEEE Computer Society, 2018.

[204] Guangcong Zheng, Xianpan Zhou, Xuewei Li, Zhongang Qi, Ying Shan, and Xi Li. LayoutDiffusion: Controllable Diffusion Model for Layout-to-Image Generation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pages 22490–22499. IEEE, 2023.

[205] Xiaomao Zhou, Yanbin Gao, and Lianwu Guan. Towards Goal-Directed Navigation Through Combining Learning Based Global and Local Planners. *Sensors*, 19(1):176, 2019.

[206] Weijin Zhu, Yao Shen, Linfeng Yu, and Lizeth Patricia Aguirre Sanchez. GMAIR: Unsupervised Object Detection Based on Spatial Attention and Gaussian Mixture, June 2021. arXiv:2106.01722 [cs].

[207] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable DETR: Deformable Transformers for End-to-End Object Detection. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[208] Piotr Zielinski and Tomasz Kajdanowicz. Learning Scale-Invariant Object Representations with a Single-Shot Convolutional Generative Model. In Derek Groen, Clélia de Mulatier, Maciej Paszynski, Valeria V. Krzhizhanovskaya, Jack J. Dongarra, and Peter M. A. Sloot, editors, *Computational Science - ICCS 2022 - 22nd International Conference, London, UK, June 21-23, 2022, Proceedings, Part III*, volume 13352 of *Lecture Notes in Computer Science*, pages 613–626. Springer, 2022.

[209] Piotr Zielinski and Tomasz Kajdanowicz. RDIR: Capturing Temporally-Invariant Representations of Multiple Objects in Videos. In *IEEE/CVF Winter Conference on Applications of Computer Vision Workshops, WACVW 2024 - Workshops, Waikoloa, HI, USA, January 1-6, 2024*, pages 588–597. IEEE, 2024.

[210] Piotr Zielinski and Urszula Markowska-Kaczmar. 3D robotic navigation using a vision-based deep reinforcement learning model. *Appl. Soft Comput.*, 110:107602, 2021.

[211] Zhuofan Zong, Guanglu Song, and Yu Liu. DETRs with Collaborative Hybrid Assignments Training. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, pages 6725–6735. IEEE, 2023.