

LEVERAGING DATA-BASED INDUCTIVE BIASES IN DEEP LEARNING

Dissertation

zur Erlangung des Grades eines
Doktors der Naturwissenschaften
der Technischen Universität Dortmund
an der Fakultät für Informatik

von
STEFAN SYLVIUS WAGNER

Dortmund
2025

Abstract

Deep learning has revolutionised pattern recognition and decision-making, yet its success crucially depends on the *inductive biases* embedded in models implicit assumptions that guide learning toward useful solutions. These biases determine not only what a model can learn, but also how efficiently it can do so. Despite their importance, the sources and effects of inductive biases are rarely viewed in a more general manner

This dissertation investigates how inductive biases can be systematically identified and leveraged to improve learning across diverse domains. It formalises a distinction between two complementary kinds of inductive biases: (i) **model-based inductive biases**, which are encoded in architectures, loss functions, and regularization terms; and (ii) **data-based inductive biases**, which derive directly from structural or statistical regularities in the data. While the former has driven much of modern progress from convolutional networks to attention-based models, the latter remains comparatively underexplored despite its potential to capture task-specific invariances that are difficult to encode architecturally.

The central contribution of this thesis is to define data-based inductive biases, which can be used to complement and enhance model-based inductive biases to create more robust and generalisable models. For this, a framework is introduced to describe how data-based inductive biases can be extracted and integrated into existing models, enhancing learning without the need for architectural redesign or extensive supervision. Latent variable modeling plays a key role in this framework, providing a principled means of discovering and encoding data structure as an additional learning signal.

The chapters in this thesis demonstrate this principle across multiple domains. *Cyclophobic Reinforcement Learning* introduces a hand-crafted data-based inductive bias via hierarchical latent spaces over environment views, improving exploration by discouraging cyclic trajectories across these hierarchical perspectives.

The Power of LLM-Generated Synthetic Data for Online Political Discussions uses synthetic data generated by large language models to define a controllable latent space, identifying ambiguous real-world samples that can be removed to reduce annotation cost or manually labelled to refine the model’s decision boundary.

Just Cluster It: An Approach for Exploration in High Dimensions Using Clustering and Pre-Trained Representations shows how clustering DINO representations with Gaussian mixture models enables learning a latent density model over the state space, yielding more informative pseudo-count estimates for intrinsic motivation in 3D environments.

Oh-A-DINO: Understanding and Enhancing Attribute-Level Information

in Self-Supervised Object-Centric Representations demonstrates how augmenting DINOv2 representations with object-centric latent features learned through a Variational Autoencoder mitigates limitations of foundation models in representing non-geometric features such as colour and texture.

Across these applications, the integration of data-derived structure consistently enhances performance and interpretability, showing that data-based inductive biases are a fundamental component to model learning.

Contents

1	<i>Introduction</i>	23
1.1	<i>Research Questions</i>	24
1.2	<i>Contributions</i>	25
1.3	<i>Thesis Outline</i>	25
1.4	<i>Notation</i>	27
2	<i>Inductive Biases in Deep Learning</i>	29
2.1	<i>Model-based Inductive Biases</i>	30
2.2	<i>Data-based Inductive Biases</i>	32
2.3	<i>A Framework for Model- and Data-based Inductive Biases</i>	34
3	<i>Learning Representations through Model-based Inductive Biases</i>	37
3.1	<i>Representations in Deep Learning</i>	37
3.2	<i>Training a Deep Learning Model</i>	39
3.3	<i>The (Variational) Autoencoders Strong Model-based Inductive Biases</i>	45
4	<i>Leveraging Latent Spaces for Data-based Inductive Biases</i>	49
4.1	<i>Latent Variables and Latent Spaces</i>	50
4.2	<i>Learning Latent Spaces with Latent Variable Models</i>	51
4.3	<i>Inducing Latent Spaces with Transformations On Data</i>	57
4.4	<i>Leveraging Latent Spaces Into a Learning Component</i>	58
4.5	<i>Self-Supervised Learning Integrates Model- and Data-based Inductive Biases</i>	61
4.6	<i>Summary</i>	65

5	<i>Cyclophobic Reinforcement Learning: A Cycle-Avoiding Data-Based Inductive Bias for Exploration in Reinforcement Learning</i>	69
5.1	<i>Preamble</i>	70
5.2	<i>Introduction</i>	70
5.3	<i>Contributions</i>	72
5.4	<i>Related Work</i>	73
5.5	<i>Cyclophobic Reinforcement Learning</i>	73
5.6	<i>Experiments</i>	78
5.7	<i>Discussion</i>	84
6	<i>Leveraging an LLM as a Data-Based Inductive Bias to Find Ambiguous Samples in Latent Space To Improve Stance Detection</i>	87
6.1	<i>Preamble</i>	89
6.2	<i>Introduction</i>	90
6.3	<i>Contributions</i>	91
6.4	<i>Related Work</i>	91
6.5	<i>Using Synthetic Data to Improve Stance Detection</i>	94
6.6	<i>Experiments</i>	97
6.7	<i>Ablations</i>	101
6.8	<i>Discussion</i>	103
7	<i>Improving Exploration by Learning a Latent Observational Space Through Representational Clustering</i>	107
7.1	<i>Preamble</i>	108
7.2	<i>Introduction</i>	109
7.3	<i>Contributions</i>	112
7.4	<i>Related Work</i>	113
7.5	<i>Just Cluster It: Exploration via Episodic and Global Clustering</i>	115
7.6	<i>Experiments</i>	118
7.7	<i>Discussion</i>	124

8	<i>Realigning Pre-Trained Vision Models with Object-Centric Representations for Multi-Object Instance Retrieval</i>	125
8.1	<i>Preamble</i>	126
8.2	<i>Introduction</i>	127
8.3	<i>Contributions</i>	128
8.4	<i>Related Work</i>	129
8.5	<i>Analysing Representation Quality with Object-Centric Instance Retrieval</i>	130
8.6	<i>Enhancing DINOv2 with Object-Centric Representations</i>	132
8.7	<i>Experiments</i>	136
8.8	<i>Stanford Cars: Real-World Instance Retrieval</i>	141
8.9	<i>Discussion</i>	141
9	<i>Conclusion and Outlook</i>	143
9.1	<i>Key Contributions and Insights</i>	143
9.2	<i>Outlook for Future Research</i>	145
9.3	<i>Concluding Remarks</i>	146
	<i>Bibliography</i>	147
	<i>Appendices</i>	163
A	<i>Cyclophobic Reinforcement Learning</i>	164
B	<i>Just Cluster It</i>	168
C	<i>The Power Of LLM-generated Synthetic Data For Online Political Discussions</i>	173
D	<i>Oh-A-DINO</i>	181

List of Figures

- 2.1 **Practical examples of model-based inductive biases:** We give an overview of components that induce model-based inductive biases together with their corresponding inductive biases. These components can be categorised into three distinct components: architecture, loss functions and regularisation. 30
- 2.2 **A framework for model- and data-based inductive biases:** The data-based inductive bias defines a learning signal (component) that encodes specific patterns in the data extracted by the structure inducing function T . This learning signal can then be used to guide representation learning in a deep learning model. 32
- 2.3 **Examples of data-based inductive biases.** We show examples of data-based inductive biases (increasing representation complexity) with their corresponding structure inducing function T and the resulting latent space. A simple data-based inductive bias is based on simply cropping the observations, which modifies the underlying environment transition dynamics. A more complex structure inducing function is a Gaussian Mixture Model which explicitly models a latent space over the input data. The structure inducing function T can also contain model-based inductive biases such as LLM encoder, in order to extract a latent space for the data-based inductive bias. 33
- 3.1 **Autoencoder vs Variational Autoencoder architecture:** The autoencoder is a simple deep learning model that learns representations by reconstructing the input data. The variational autoencoder is an extension of the autoencoder that learns representations by reconstructing the input data and assumes the data can be modelled under a Gaussian distribution. The variational autoencoder makes use of different model-based inductive biases to learn a Gaussian latent space. 45

- 4.1 **Examples of data transformations in practice:** These examples show how data transformations can help shape the structure inducing function T . Pre-trained representations can be used to extract features from raw data, which can then be leveraged into a latent space with a latent variable model such as a GMM. More simple transformations such as data augmentations can be used to expose invariances in the data, after which a (pretrained) neural network can be used to embed the transformed data into a latent space. Such combinations arise often in practice. 57
- 4.2 **Self-supervised learning under the model-based, data-based inductive bias view:** Self-supervised learning consists of model-based and data-based inductive bias paths. The data-based path's parameters are fixed and only updated periodically serving as a reference to the model-based path. The structure inducing function T makes use of data augmentations and a neural network to project the invariances defined by the data augmentations to a latent space. The latent representation from the data-based path is then used to update the model-based path. 62
- 4.3 Data augmentations 63
- 5.1 **Cropping and intrinsic rewards define the data-based inductive bias of Cyclophobic Reinforcement Learning** 71
- 5.2 **Hierarchical views allow us to transfer relevant invariances about the environment.** (a) The three different representations are obtained by cropping the observation for each state-action pair (s, a) . V_1 is the full view where the agent sees the whole environment or the largest portion of it. V_2 is an intermediate cropped representation of the agent's view that helps the agent generalize by reusing familiar observations. V_3 is the most restricted view where the agent only sees what is immediately in front of it. (b) Through the views V_1 to V_k the amount of cycles continuously increases as the observations in the higher views can be mapped multiple times to the same observation in the lower view. This naturally leads to each view being a separate POMDP describing the true MDP. 74
- 5.3 **Minigrid Exploration:** The hierarchical representations together with cyclophobic intrinsic reward allow for much quicker convergence than other methods. For the KeyCorridor and MultiRoom environments colour invariance is crucial to solving the environments which we analyse in our ablation study. 79

- 5.4 **MiniHack Navigation:** The agent converges quicker in the *Navigation* task than the intrinsic curiosity baselines such as RIDE [Raileanu and Rocktäschel, 2020b] and Random Network Distillation [Burda et al., 2018b]. This corroborates our hypothesis, that avoiding cycles is essential for quick exploration. 80
- 5.5 **MiniHack Skill:** We converge quicker than IMPALA [Espeholt et al., 2018b] in the *Skill* task. The *Skill* task defines over 75 different actions the agent must learn to use making it one of the most difficult sets of environments of the MiniHack suite. 81
- 5.6 **Studying the data-based inductive bias in cyclophobic reinforcement learning:** Hierarchical state representations are crucial to performance in cyclophobic reinforcement learning. We observe that the hierarchical views improve performance for both the optimistic initialization and cyclophobic agent, while leveraging the largest performance boost with the cyclophobic intrinsic reward. Toggling between the largest, intermediate and smallest views. We see that all 3 types of views deliver the best performance. 82
- 5.7 **Exploration behaviour(visitation counts for “DoorKey-16x16”):** The cyclophobic Q-function explores the environment more efficiently than the optimistic initialization, count and epsilon-greedy based counterparts. We record visitation counts for several variations of intrinsic reward and hierarchical state representations. Hierarchical state representations together with the cyclophobic intrinsic reward are the most efficient. 83
- 5.8 **Observational complexity ablation study (reducing colors):** The cyclophobic agent can gather rewards successfully in large “Key-Corridor” environments when reducing colors, showing that the exploration mechanism is very strong. Generally, the more distinct i.e. complex the observations in the environment are, the more the learned invariances from the neural network matter. 84
- 5.9 Using PPO together with only the cyclophobic intrinsic reward also converges far quicker than C-BET. Moreover, we are able to converge in KeyCorridorS3R3 confirming our theory in Section 5.7.1. 85
- 6.1 **LLM-generated synthetic data together with pre-trained language representations make up a distinct latent sapce that can be leveraged for improved stance detection.** 89

- 6.2 **We investigate the use of LLM-generated synthetic data for stance detection in online political discussions.** (A) We generate synthetic data  for specific questions using a Mistral-7B model. The synthetic data is then used to fine-tune the stance detection model. We show that fine-tuning with synthetic data improves the performance of the model, since the synthetic data is roughly faithful to the real data's  underlying distribution. However, some real world samples  cannot be captured by the synthetic data. (B) We therefore use the synthetic data to identify the most informative samples  in the unlabelled real data pool, which are better off labelled by human experts. Combining the synthetic data with the manually labelled most informative samples improves the performance of the model even further. 92
- 6.3 **Q1: Fine-tuning the model with synthetic data improves performance for increasing dataset size:** Shown are the F1-Score of fine-tuning with **Only Synthetic Data** (left) and **Synthetic Data + True Labels** (right) for increasing synthetic dataset size. Even if a dataset has been fully labelled, augmenting it with synthetic data proves equally as effective. 99
- 6.4 **(Q2) Analysing the synthetic data (M=1000):** The synthetic data aligns well with the real data, which is crucial for improving stance detection performance and to check for potential biases introduced by the synthetic data. SQBC selects the samples that are in between the two classes, i.e, that are the most ambiguous and informative for the model. 100
- 6.5 **(Q3) Fine-tuning with synthetic data improves stance detection, while combining most informative samples and synthetic data surpasses the baseline model fine-tuned with all true labels ( above dashed line ) using less manually labelled data:** The reason for the performance increase can be attributed to two phenomena: (i) the synthetic data smoothens the decision boundary of the model making it more robust to outliers. (ii) The most informative samples improve the model where the synthetic data distribution is not expressive enough. 101
- 6.6 **Synthetic data properties: (Left)** The per comment entropy of the synthetic data is similar to the entropy of the real data, where the synthetic data has a higher mean entropy than the real data, while the real data has higher variance. **(Right)** The generated synthetic data is of high quality, with only a few outliers. Interestingly, synthetic comments are longer than real ones, making real comments denser and synthetic ones more verbose. We argue that since the projected embeddings in Figure 6.4 show alignment between both datasets, BERT appears to be invariant to comment length. 103

- 7.1 **We induce a discrete latent space by clustering pre-trained DINO representations on high-dimensional 3D observations. This is then used to produce a count-based intrinsic reward that describes the latent structure and thus informs exploration.** 109
- 7.2 **“Just Cluster It” in a nutshell:** (a) Density estimation for exploration in 3-D environments is challenging, because the magnitude of pixel change is large, but the saliency of a single transition is low. This is in contrast to 2-D environments where every transition is distinct and therefore salient. Therefore, we propose clustering representations on an episodic and global level to estimate accurate pseudo-counts that reflect the state-space distribution. (b) We show pre-trained representations from a DINO model. The embeddings from the DINO model are able to extract relevant features from the observations, which is useful for clustering. The bottom images are thresholded embeddings. (c) To estimate pseudo-counts, we store episodic cluster centers in a global cluster table over time by matching episodic cluster centers with previously added cluster centers in the cluster table. A new cluster center is added to the table, if the cosine similarity to existing cluster centers is below a threshold κ . 110
- 7.3 **Clustering random features is effective for VizDoom:** Our method with clustering outperforms other traditional exploration methods. Interestingly, random features perform slightly better than the DINO features for VizDoom. While the complexity of the observations is not trivial, in this setting the priors in the pre-trained representations seem to be not more informative than random features. This also shows that in 3-D high-dimensional environments random features are salient enough when clustered to estimate pseudo-counts. 118
- 7.4 **Pre-trained DINO features excel with more complex observations:** When training on the Habitat environment we see that clustering with DINO embeddings is more effective than clustering with random features. Furthermore, only our method is able to leverage the pre-trained DINO features effectively. We argue that the priors present in the DINO embeddings help in aggregating the representations when building the episodic clusters, which ultimately determine the pseudo-counts. 119

- 7.5 **Ablation studies: We show the effectiveness of clustering, especially episodic clustering:** **(Left)** Increasing the cosine similarity threshold κ increases the number of global clusters in the cluster table. If the number of clusters in the cluster table are too granular, performance decreases as shown by lower visitation counts for higher values of κ . **(Right)** Episodic clustering helps whenever there is structure in the representations. Especially for DINO in Habitat performance drops significantly, if episodic clustering is removed. This can also be seen in VizDoom (left panels) where episodic clustering helps both DINO and random features. For Habitat (right panels) with random features, episodic clustering has no large effect since the random features are not expressive enough for the complex observations. 120
- 7.6 **Visualization of DINO clusters for observations with random noise:** We plot different samples from the cluster table for the noisy observations in Habitat. Each image represents the mean image of an episodic cluster that was aggregated to the cluster table. The clustering happens on the 384-dimensional embedding. Even with the concatenated random noise the embeddings are still clustered sensibly. 122
- 7.7 **We concatenate random noise to the observations:** To test the robustness of our method to observations with random noise, i.e, the noisy TV problem, we perform the same experiment as Saade et al. [2023] and concatenate a frame with random noise that changes after every step to the observation. 123
- 7.8 **Our method is robust to random noise in the observations:** We see that our method is robust to random noise in the observations. As a sanity check we also evaluate RECODE on noisy observations and confirm the results from Saade et al. [2023]. Surprisingly, for both methods the performance increases when adding random noise. We suspect the random noise has a regularising effect on the clustering. 123
- 8.1 **Oh-A-Dino learns a continuous latent space of fine-grained object features such as colour and material to enhance existing self-supervised learnt representations. The enhanced representations are effective at retrieving objects with multiple attributes in multi-object instance retrieval.** 126

- 8.2 **DINOv2 representations and slot-based representations struggle at multi-object instance retrieval, due to weak object-specific features (largely colour) in its embeddings. Our method is able to retrieve more relevant images by combining general scene and local representations.** DINOv2 representations excel at retrieving images where multiple attributes match such as shape and size, while retrieval degrades when adding colour. Slot-based representations lack specialisation, where retrievals sometimes match fine-grained features such as colour but often failing to retrieve a similar object altogether. Our method performs the best being able to augment the DINOv2 representation to mitigate its shortcomings. 128
- 8.3 **Object-level features improve prediction for all attributes, while also improving retrieval of multiple simultaneous attributes.** The main improvement is achieved on the colour(texture) attribute, which indicates that our method mitigates for the lacking learning signal in the other methods' representations. Slot-based methods perform worse than the pretrained DINOv2 representations when predicting multiple attributes simultaneously. 131
- 8.4 **Our method leverages the implicit general object understanding of DINOv2 to extract object-level features for multi-object instance retrieval. B:** Traditional slot-based object-centric methods learn slot-representations via cross-attention which provide little inductive bias, while having to compress global and object-level features into a small set of latents. **A:** We propose combining general self-supervised features with learnt object-level features to obtain an improved latent representation. The object-level features are learnt from image patches making training efficient and the latent space expressive. 132
- 8.5 **Visualisations of retrievals for CLEVRTex, augmenting with VAE object level features allows for retrieval in complex multi-object scenes. Oh-Ah-Dino** consistently retrieves the object in question with the correct properties such as the shape, size and material. **DINOv2** on the other hand often retrieves the correct object but with the wrong texture. 138
- 8.6 **Oh-A-DINOv2 improves over SSL and slot-based features for multi-object instance retrieval.** Our method outperforms both self-supervised features and slot-based approaches. The table shows that Oh-A-DINOv2 achieves higher precision and lower error rates on CLEVR and CLEVRTex, indicating that combining DINOv2 with object-level VAE features addresses the limitations of pure SSL features and slot-attention methods. 138

- 8.7 **Object-level VAE features enhance color and multi-attribute retrieval.** While the standalone VAE improves fine-grained color retrieval on both CLEVR and CLEVRText, only Oh-A-DINOv2 effectively integrates this signal with DINOv2 scene-level features. This leads to substantial gains in multi-attribute retrieval, where pure VAE features or SSL features (DINOv2) alone fall short. 139
- 8.8 **Ablation on VAE augmentation and model capacity.** (top) Augmenting SSL and slot-based features with object-level VAE latents improves retrieval across methods, with the largest gains observed for Oh-A-DINOv2. (bottom) Scaling backbone capacity from DINOv2-S to DINOv2-L provides only minor improvements. 140
- 8.9 **Our method improves over baselines and leverages DINOv2’s general object understanding.** (left) Table: comparison against baselines on Stanford Cars shows that our method achieves lower distance to the colours of the reference image. (right) Oh-A-Dino can consistently retrieve cars which match orientation and shape while matching colour and material opposed to DINOv2. 140
- 1 Using PPO together with only the cyclophobic intrinsic reward also converges far quicker than C-BET. Moreover, we are able to converge in KeyCorridorS3R3 confirming our theory in Section 5.7.1. 164
- 2 An overview of the MiniGrid environments used in this paper. 164
- 3 An overview of the Minihack environments tested in this paper. The top row shows the *Navigation* tasks, while the bottom row shows the *Skill* tasks. 165
- 4 **Comparison of DINO clusters and Random clusters:** Observations from the Habitat environment visualized as samples from the cluster table for DINO (magenta) and random features (cyan). DINO clusters aggregate more dissimilar but semantically related observations, whereas random clusters are more granular and distortion-free. Random features still form aggregated clusters, but some counts are unusually high, suggesting less reliable matching of similar observations. 169
- 5 **High level overview of policy network:** The policy network receives the observations and outputs the logits for the action probabilities and the value function. 171
- 6 **High level overview of how we use DINO:** We use the pre-trained DINO model to extract embeddings from the observations. We then use the cosine similarity to cluster the embeddings. 172

7 Overview of the pipeline for active learning with synthetic data:

We start with a question q from the X-Stance test dataset and translate the question to English with a NLLB-330M model [NLLB Team et al., 2022]. Then we let the Mistral-7B model generate synthetic data, i.e., comments for the translated question. The generated comments are then translated back to German to be used for fine-tuning the model in our experiments. 180

List of Tables

6.1	Number of comments in our 10 selected test questions for our experiments. The numbers are split into (favor against) labels. We have a 60 – 40 (Test-Train, Test-Test) split.	97
6.2	Topic alignment is crucial for the synthetic data to be effective: To determine whether improvement with synthetic data is due to the structure or content of the synthetic data, we augment the stance detection model with misaligned synthetic data. That is, the synthetic data does not align with the question given to the stance detection model. We observe the model only performs meaningfully better, when the synthetic data aligns with the posed question.	102
1	PPO Hyperparameters for VizDoom	170
2	Cosine Similarity and Episodic Clustering Parameters	170
3	RECODE Parameters VizDoom - Bold value is used for experiments	170
4	English comments sampled from the interquartile ranges of the entropy distribution of the synthetic data.	174
5	Corresponding German comments sampled from the interquartile ranges of the entropy distribution of the synthetic data.	175
6	Corresponding German comments sampled from the interquartile ranges of the entropy distribution of the real data.	176
7	Chosen questions for stance detection in German and their English translation	177
8	Sample of translated comments from comments generated by the LLM used for fine-tuning the stance detection model.	178
9	Sample of comments generated by the LLM used for fine-tuning the stance detection model.	179

- 10 Dataset characteristics for CLEVR and CLEVRText. The CLEVRText dataset has additional textures for the objects and backgrounds compared to CLEVR. It also has no color information for the objects, which is replaced by the material information. In general, instance retrieval with the CLEVRText dataset is much harder due to the increased number of possible object configurations. 181
- 11 **Data splits for Stanford Cars dataset and CLEVR and CLEVRText datasets.** The train set is used to train the VAE, while the validation set is used to select query images for the retrieval task. The test set contains the candidate images for the retrieval task. We only evaluate on the Stanford Cars Dataset where we take a subset of 5000 candidate images and 175 query images split into groups of 25. 182
- r.7 dedication

As is well known, no great feat is achieved on its own. I therefore take this section to acknowledge all the people who have helped me complete this journey.

I first want to thank my mother Cila who has always pushed me to pursue and value education. Without her support and unconditional love none of this would have been possible. While my late father Sylvius is not around to see my achievements, his infectious curiosity for all types of things and loving nature were certainly the kickstarters for my research career even though at the time we did not know this was going to be my path. In the same sense, I want to thank my brother Adrian who for my fifth birthday gifted me my first computer and had a big influence on me becoming a computer scientist.

I want to thank my partner Saeromi who has been with me throughout this time through thick and thin. She was always there when I needed her and this finished body of work is as much hers as it is mine. Finally, I want to thank my mentor Ralph who i've known for most of my adult life and taught me everything I know about life besides research.

I want to thank Prof. Dr. Stefan Harmeling for being a great PhD supervisor from whom I learned how to become a researcher. His knack for teaching and enthusiasm for the field always motivated me to go beyond what I thought was possible without feeling overwhelmed. I think this quality is incredibly rare and I feel fortunate to have been able to work in his group.

I want to thank my colleagues: Maike, Jan, Marc, Sebastian, Tobias who made the whole experience fun even during stressful times. Thanks to you I always felt equally comfortable as well as challenged. I want to thank the UPEKI team as well (Carina, Mira, Marc), I really enjoyed our cooperation and work together.

I also want to thank the people I met at conferences with whom I shared these special experiences: Amaya, Andrew, Timo I will never forget those times spent together.

Finally I also want to thank UPEKI and the Lllamar Insitute for providing the resources to perform this research.

Usage of Large Language Models (LLMs) This thesis was written with the help of LLMs. Specifically, the LLM was used to improve readability and clarity of text. At times it was also used to improve the structure of some the sections. The ideas and statements made however are of my own.

Introduction

The success of deep learning has fundamentally transformed how we approach complex pattern recognition and decision-making tasks. At the heart of this transformation lies a crucial but often implicit design choice: the inductive biases we embed in our models. These biases, that is the assumptions about what constitutes a good solution, determine not just what our models can learn, but how efficiently they can learn it.

Historically, the field has evolved through increasingly sophisticated ways of encoding these biases. Early neural networks relied primarily on the universal approximation capabilities of fully connected architectures, learning representations through supervised training on large labeled datasets. While powerful, this approach proved inefficient, requiring massive amounts of annotated data and often failing to capture the underlying structure of complex real-world problems.

The introduction of specialised architectures marked a shift toward more structured inductive biases. Convolutional Neural Networks (CNNs) embedded assumptions about spatial locality and translation invariance, while Transformers incorporated attention mechanisms that capture long-range dependencies. These *model-based inductive biases* encoded in architectural choices, loss functions, and regularisation techniques have enabled remarkable progress across domains from computer vision to natural language processing.

More recently, self-supervised learning has demonstrated that powerful representations can emerge by leveraging patterns inherent in the data itself. Methods like contrastive learning and masked language modeling exploit data augmentations and structural invariances to learn without explicit supervision. This success hints at a largely underexplored dimension: *data-based inductive biases* that derive learning signals directly from the structure of the data.

In this work, we explore the role of inductive biases in deep learning models and how they can be systematically leveraged to enhance learning across diverse domains. We formalise a distinction between

two complementary types of inductive biases: (i) model-based inductive biases, which are embedded through architectural choices, loss functions, and regularization techniques, and (ii) data-based inductive biases, which exploit the inherent structure present in the data itself to inform the learning process. While model-based inductive biases have been extensively studied and refined, data-based inductive biases remain underexplored despite their potential to capture task-specific patterns that would be difficult to encode through model design alone.

Our central contribution is demonstrating that these two types of inductive biases are complementary approaches that can be systematically combined to achieve superior learning performance. We develop a principled framework for extracting data-based inductive biases and show how they can augment existing model-based approaches across multiple domains, including reinforcement learning, natural language processing, and computer vision.

1.1 *Research Questions*

How are model-based and data-based inductive biases defined? We aim to distinguish model-based inductive biases (embedded in architectures, loss functions, and regularization) from data-based inductive biases (derived from data structure itself), providing the conceptual foundation for understanding how different bias types influence learning.

How do model-based inductive biases learn structured representations? While model-based inductive biases are well-established, we characterise these by giving an overview of architectural choices, learning objectives and regularisation techniques and analyse their role in learning representations.

Which approaches are effective at encoding and integrating data-based inductive biases? This work explores various approaches to encode data-based inductive biases, i.e., to leverage structure in the data itself. From handcrafted data augmentations to latent-variable models that automatically discover structure, the thesis explores multiple approaches to leverage patterns in data as learning signals.

How can we leverage data-based inductive biases to enhance model-based inductive biases? We define a framework that elegantly combines model-based and data-based inductive biases. This framework proposes simple building blocks for leveraging structure from data and to create a learning signal to enhance model learning. We show that popular representation learning methods such as self-distillation in self-supervised learning are special instantiations of this framework.

How can model-based and data-based inductive biases be effectively combined to enhance learning performance across diverse domains?

Building on this framework, we investigate how it can augment approaches across reinforcement learning, natural language processing, and computer vision tasks, seeking to understand when data-based inductive biases can be effectively integrated.

1.2 Contributions

Following from the research questions outlined above, the contributions of this work can be summarized as follows:

1. **We establish a simple framework distinguishing model-based and data-based inductive biases in deep learning.** We formalise how model-based inductive biases (embedded in architectures, loss functions, and regularization) can be systematically combined with data-based inductive biases (derived from data structure itself) to enhance learning performance across diverse domains.
2. **We present latent variable modeling as a principled approach for extracting data-based inductive biases from structured representations.** When pre-trained or learned representations are available, we show that latent variable models can automatically discover latent structures from which a learning signal can be leveraged to enhance performance on different downstream tasks.
3. **We demonstrate how pre-trained representations can be enhanced through data-based inductive bias signals.** By augmenting existing pre-trained models with structured information derived from data, we show that the limitations of foundation models can be mitigated without requiring full retraining, leading to improved performance on tasks requiring fine-grained understanding.
4. **We demonstrate the effectiveness of data-based inductive biases when data structure reveals task-relevant information.** Through systematic application across reinforcement learning, natural language processing, and computer vision, we show how leveraging inherent data patterns can provide learning signals that complement traditional model-based approaches.

1.3 Thesis Outline

To address the research questions and contributions outlined above, the thesis is structured as follows:

Chapter 2 discusses the role of inductive biases in deep learning models, presenting both model-based and data-based inductive biases.

We formalise the distinction between these two types of biases and establish the theoretical framework that underpins the subsequent chapters.

Chapter 3 explores the role of learning representations in deep learning models, which primarily falls under model-based inductive biases. We discuss how architectural choices, loss functions, and regularization techniques shape learned representations. The chapter concludes with a walkthrough of Variational Autoencoders which make use of several model-based inductive biases to learn a Gaussian latent space.

Chapter 4 introduces latent variable modeling as a systematic approach for extracting data-based inductive biases through latent spaces. We discuss how latent spaces can induce structure over data and how this structure can be leveraged into learning signals. Additionally, we explore how learned representations themselves contain compressed information that can inform data-based inductive biases through latent variable modeling. The chapter concludes with an introduction to self-supervised learning, which demonstrates how model-based inductive biases can be combined with simple data-based inductive biases.

Chapter 5 presents cyclophobic reinforcement learning, which leverages a hand-crafted data-based inductive bias by defining a latent space over hierarchical views of the environment. This approach improves exploration in reinforcement learning by incorporating cycle-avoiding intrinsic rewards that exploit the structural information present in different environmental perspectives.

Chapter 6 explores stance detection in online political discussions, where synthetic data generated by large language models creates a data-based inductive bias. We show how this approach can identify ambiguous samples in real data and improve classification performance while reducing manual annotation requirements.

Chapter 7 demonstrates how pre-trained representations can serve as the foundation for data-based inductive biases in reinforcement learning exploration. We show how clustering DINO representations with Gaussian mixture models creates effective pseudo-count estimates for exploration, particularly addressing the transition saliency problem in 3D environments.

Chapter 8 combines model-based and data-based inductive biases for multi-object instance retrieval. We augment pre-trained DINOv2 representations with object-centric features learned through a Variational Autoencoder trained on image patches, demonstrating how data-based inductive biases can address limitations in foundation models.

Finally, Chapter 9 concludes the thesis by synthesizing the key insights, discussing the broader implications of combining different types of inductive biases, and outlining promising directions for future research.

1.4 Notation

1.4.1 Sets and Spaces

We use calligraphic uppercase letters to denote mathematical spaces and sets throughout this work. These include input spaces, output spaces, latent spaces, representation spaces, embedding spaces, state and action spaces, observation spaces, and datasets (e.g., $\mathcal{X}, \mathcal{Y}, \mathcal{Z}, \mathcal{H}, \mathcal{S}, \mathcal{A}$).

1.4.2 Functions

Functions are typically denoted using lowercase letters (e.g., f, g, h) or in some cases by uppercase greek letters (e.g., Φ, Ψ) when discussing more abstract mappings. We make two special distinctions for the uppercase letters T and C , which are reserved for the structure inducing function T and the learning component function C , respectively, which play central roles in our framework for combining model-based and data-based inductive biases.

1.4.3 Variables and Vectors

Specific Instances. Lowercase letters represent specific instances, realizations, or particular values of variables. This includes data points, outputs, targets, latent variable instances, learned representations, and other vector quantities (e.g., x, y, z, h).

Random Variables. Random variables are denoted using bold lowercase letters, distinguishing them from their specific realizations (e.g., $\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{h}$).

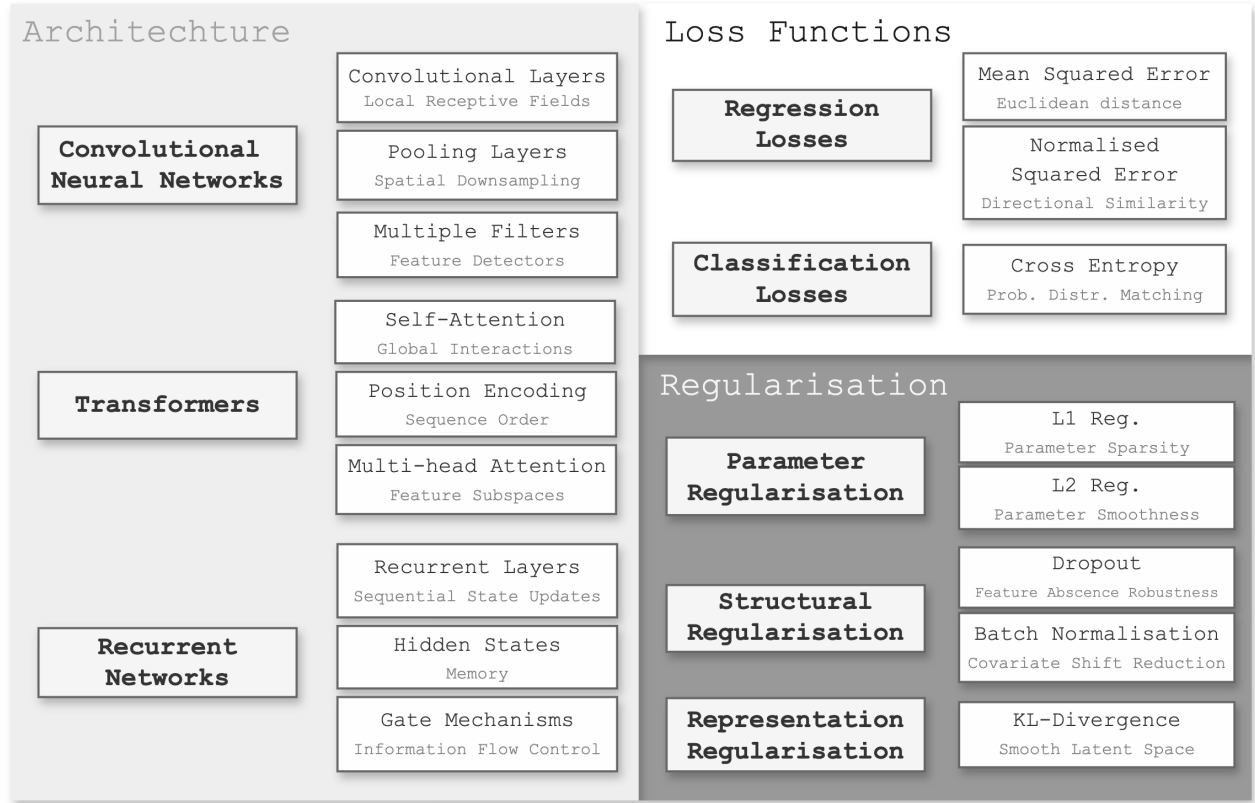
Parameters. Model parameters and hyperparameters are represented using Greek letters (e.g., $\theta, \phi, \mu, \sigma$). Subscripts and superscripts provide additional indexing when needed to distinguish between different model components, layers, or time steps.

Inductive Biases in Deep Learning

One of the central aspects of statistical learning theory is the concept of *inductive bias*. It is the set of assumptions that a learning algorithm makes in order to learn from data. The inductive bias is crucial for the success of a learning algorithm, as it defines the space of hypotheses that the learning algorithm can assume.

In the context of deep learning, inductive biases are numerous: the strongest inductive biases are arguably determined by the components of the learning architecture. Such as the architecture of the neural network itself, the chosen loss function and regularisation. We call such inductive biases *model-based inductive biases* since they are learning biases that are induced by the choice of model components and are in principle agnostic to data. Another type of inductive bias can be defined on specific properties of the data itself. We call this a *data-based inductive bias*. The goal of data-based inductive biases is to extract specific patterns in the data that model-based inductive biases are not able to capture. The idea is to learn a latent space that reflects specific patterns in the data which can then be leveraged to create a learning signal. Although data-based inductive biases have been implicitly present in various deep learning approaches, they have not been explicitly studied and characterised. As we will see throughout this work, data-based inductive biases arise naturally in reinforcement learning and more recently in the context of self-supervised learning.

For instance, self-supervised learning has shown that patterns and invariances in the data can be incorporated into the model by performing image transformations on the input data. By projecting these transformations through a fixed parameter network, we obtain latent representations of these transformations that the model can use as reference to learn invariances for various transformations, i.e., augmentations of the data. These invariances would be hard to embed with model-based inductive biases in an end-to-end manner. In reinforcement learning, an example of data-based inductive biases are auxiliary signals such as intrinsic rewards which guide exploration



when only sparse rewards are available. The intrinsic reward is then usually generated by quantifying the occurrence of certain patterns in the observations or in a latent space projection of the observations. This work shows that data-based inductive biases can formalise the procedure of leveraging specific patterns in the data to inform representation learning.

We thus define a framework built around model-based and data-based inductive biases, which allows for enhancing model-based inductive biases with data-based inductive biases. We show that self-supervised learning is a special case of this framework and that many different methods presented in this work can be unified under this framework to enhance model learning for downstream tasks such as reinforcement learning, representation learning and language modelling.

2.1 Model-based Inductive Biases

Model-based inductive biases can be understood as a learning bias that is introduced based on the components of the model. We illustrate this in Figure 2.1. For instance, convolutional neural networks (CNNs) in-

Figure 2.1: **Practical examples of model-based inductive biases:** We give an overview of components that induce model-based inductive biases together with their corresponding inductive biases. These components can be categorised into three distinct components: architecture, loss functions and regularisation.

corporate spatial locality and translational invariance via convolutional and pooling layers, making them well-suited for image data. Recurrent neural networks (RNNs) are designed for sequential data: they process inputs step by step in a fixed order and maintain a hidden state to capture temporal dependencies. This introduces an inherent causal and temporal inductive bias that later elements depend on earlier ones in a linear sequence. Transformers can be even more general: they can operate on sets, while sequences can be formed from sets through positional encodings.

Further inductive biases are determined by restricting the hypothesis space of the neural network via the nature of the learning objective. While a loss function such as the mean squared error penalises large deviations, the normalised squared error based on the cosine similarity encourages the model to learn representations that are similar in direction to the reference output. This is often complemented by a regularisation term that includes some prior knowledge about the data, such as sparsity or smoothness.

More formally, a model-based inductive bias can be understood as a prior belief about the model's parameters, informed by assumptions about the generative process underlying the data. Bayesian learning provides a formal way to incorporate these assumptions by defining a prior distribution over the model parameters.

Given the space of observed data $\mathcal{X}$, the posterior distribution over the parameters θ for a model is given by Bayes' theorem:

$$p(\theta|\mathcal{X}) = \frac{p(\mathcal{X}|\theta)p(\theta)}{p(\mathcal{X})}. \quad (2.1)$$

In classical Bayesian inference, the prior $p(\theta)$ explicitly encodes beliefs about parameter plausibility. In deep learning, however, such priors are often implicit: rather than specifying a probability distribution over θ , we impose structure through a deterministic mapping $\Phi(\theta)$ that reflects architectural choices, optimization dynamics, and different loss functions. This function does not define a probability density in the traditional sense, but nonetheless biases the learning process analogously to a prior, favoring certain parameter configurations over others.

We can thus rewrite the inference of the posterior as

$$p(\theta|\mathcal{X}) \propto p(\mathcal{X}|\theta) \cdot \exp(-\Phi(\theta)), \quad (2.2)$$

where $\Phi(\theta)$ is a function that acts as an unnormalised prior reflecting architectural choices, optimization dynamics and different loss functions. Note that since this term does not arise from a proper distribution, the resulting posterior can only be defined up to a proportional constant.

For this to constitute a valid unnormalized probability distribution, the expression must be non-negative and integrable. Non-negativity

is automatically satisfied since $p(\mathcal{X}|\theta) \geq 0$ and $\exp(\cdot) > 0$ for all real arguments. Integrability requires that

$$\int p(\mathcal{X}|\theta) \cdot \exp(-\Phi(\theta)) d\theta < \infty, \quad (2.3)$$

which is ensured when $\Phi(\theta)$ grows sufficiently fast relative to the likelihood as $\|\theta\|$ becomes large.

The prior function $\Phi(\theta)$ thus biases the learning process analogously to a prior, assigning lower values to more plausible parameters. This notation also allows for including additional priors on the parameters θ as will be done in the next section with data-based inductive biases. We will define model-based inductive biases more precisely in Chapter 3.

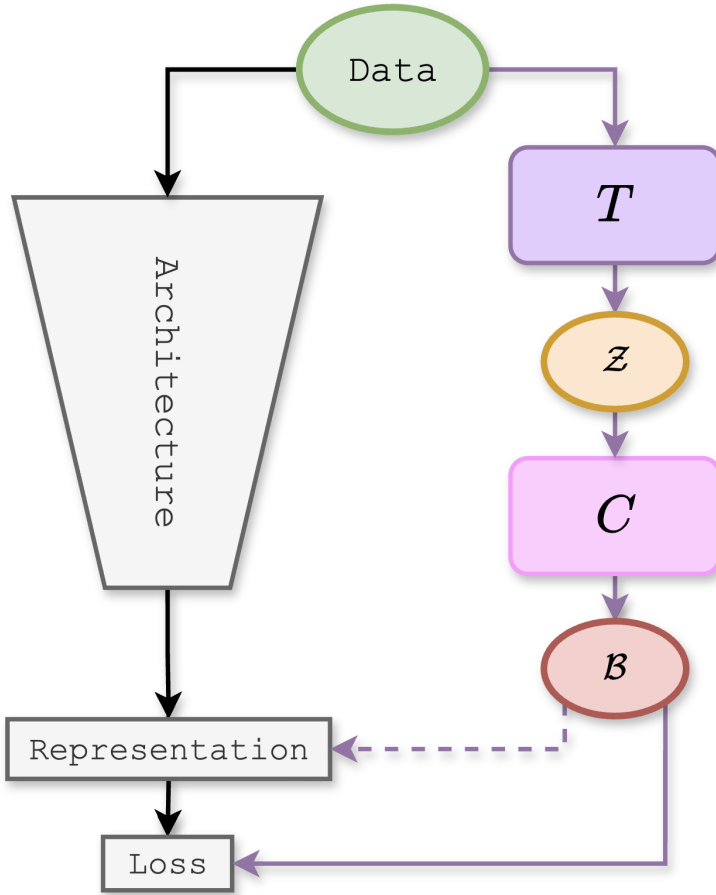
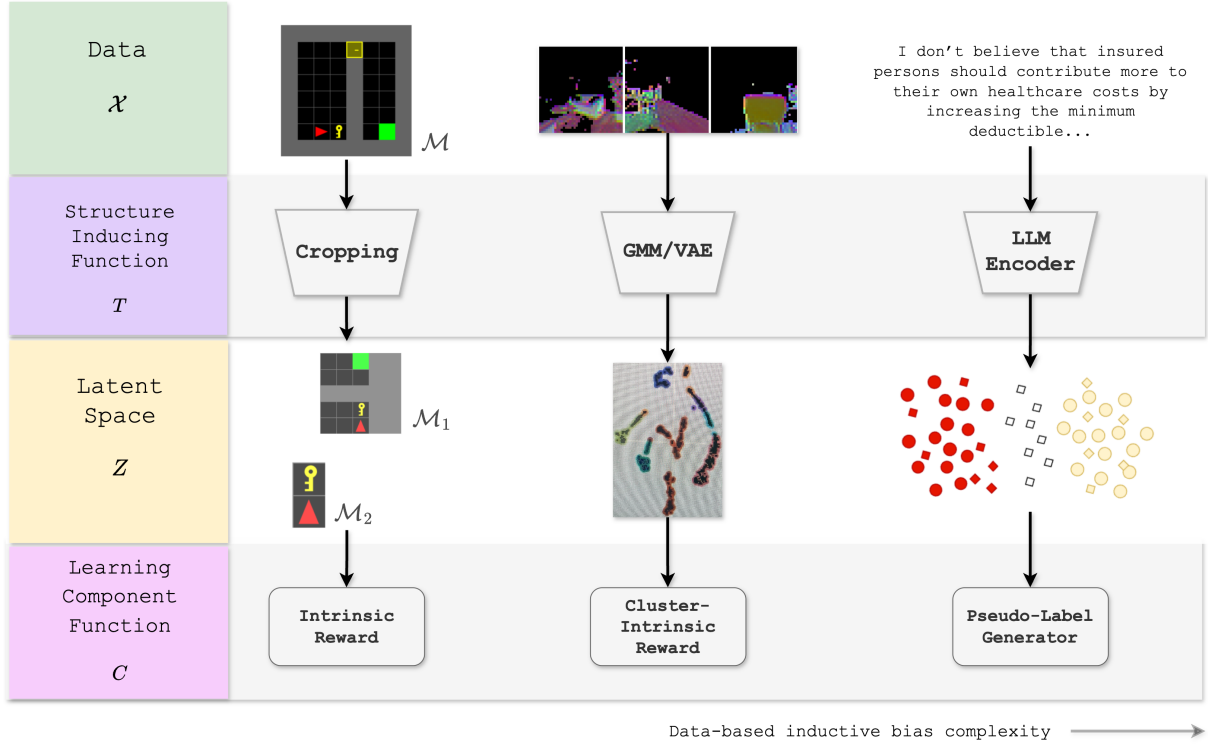


Figure 2.2: **A framework for model- and data-based inductive biases:** The data-based inductive bias defines a learning signal (component) that encodes specific patterns in the data extracted by the structure inducing function T . This learning signal can then be used to guide representation learning in a deep learning model.

2.2 Data-based Inductive Biases

While model-based inductive biases are often the focus of deep learning research, data-based inductive biases have the potential to substantially



improve deep learning models by incorporating specific patterns in the data that are difficult to embed with more general model-based inductive biases. If the data is structured in a way such that clusters or other structures emerge, this can be leveraged to learn more expressive representations. We define data-based inductive biases to consist of two processes:

1. Some structure in the data is induced, e.g., by applying transformations on the data, such as data augmentations or by learning a latent space over the data.
2. The induced structure is leveraged into a learning signal that biases the model according to the latent structure.

Thus given a dataset $\mathcal{X}$, we can define these two processes as the following functions (see also Figure 2.2):

1. A **structure inducing function** $T : \mathcal{X} \rightarrow \mathcal{Z}$ that induces some latent structure over the data resulting in a structured latent space.
2. A **learning component function** $C : \mathcal{Z} \rightarrow \mathcal{B}$ that leverages this latent space $\mathcal{Z}$ over the data to produce a learning component $\mathcal{B}$ that incorporates the data-based inductive bias into the model.

Figure 2.3: **Examples of data-based inductive biases.** We show examples of data-based inductive biases (increasing representation complexity) with their corresponding structure inducing function T and the resulting latent space. A simple data-based inductive bias is based on simply cropping the observations, which modifies the underlying environment transition dynamics. A more complex structure inducing function is a Gaussian Mixture Model which explicitly models a latent space over the input data. The structure inducing function T can also contain model-based inductive biases such as LLM encoder, in order to extract a latent space for the data-based inductive bias.

One of the central aspects of the methods discussed in this work is leveraging the interplay between learning the functions T and C to induce a data-based inductive bias into the model.

Figure 2.3 shows examples of data-based inductive biases in increasing order of representation complexity. A simple data-based inductive bias can be based on simply cropping the observations. In a reinforcement learning setting this modifies the underlying environment transition dynamics essentially inducing different Markov Decision Processes $\mathcal{M}$ (MDPs) for each cropping. This latent structure can then be leveraged into a learning signal by defining intrinsic rewards over the multiple induced MDPs. A more complex structure inducing function can be applied in a similar reinforcement learning setting by using a Gaussian Mixture Model to explicitly model a latent space over the observations. The clusters in latent space can then be leveraged into modelling a coarser and less noisy reward signal. The structure inducing function T can also consist of model-based inductive biases such as those in a pretrained LLM encoder. The condition here is that the pre-trained representations follow a latent structure that can be leveraged into a learning signal. In this sense, a way to differentiate model-based and data-based inductive bias comes down to the modelling approach of the inductive bias. The data-based inductive bias is defined a priori through the latent space induced by the structure inducing function T . The model-based inductive bias on the other hand defines no structure on the learned representations a priori and thus learns the representations iteratively through the defined learning components.

As with the model-based inductive bias, we can formalise this notion by adapting Equation 2.2 to include both the structure inducing function T and the learning component function C in the prior

$$p(\theta|\mathcal{X}) \propto p(\mathcal{X}|\theta) \cdot \exp\left(-\Phi(\theta) - \Psi(\theta, C(T(\mathcal{X})))\right). \quad (2.4)$$

In the case of the data-based inductive bias, Ψ can be understood as representing the influence of the choice of structure inducing function T and learning component function C on the model parameters θ through the data $\mathcal{X}$. Note that the model-based inductive bias Φ and the data-based inductive bias Ψ are not explicit functions that exist in reality, but rather represent the implicit assumptions on the model parameters introduced by the model-based and data-based inductive biases as shown by the examples in Figures 2.1 and 2.3.

2.3 A Framework for Model- and Data-based Inductive Biases

Having defined model-based and data-based inductive biases, we now turn to how these two forms of bias jointly shape learning in deep

learning systems. We provide a high-level synthesis of how they interact and set the stage for the more detailed treatment in the following chapters.

Figure 2.2 illustrates the overall framework for combining model-based and data-based inductive biases. The process begins by applying a structure inducing function T to the data $\mathcal{X}$, which extracts salient patterns or invariances to yield a structured latent space $\mathcal{Z}$. This latent structure captures information about the data that may not be easily expressible through architectural choices or loss function design alone.

The structured latent space $\mathcal{Z}$ is then processed by a learning component function C to produce a learning component $\mathcal{B}$ that can be systematically integrated with the model's existing inductive biases. This integration typically occurs through one of two pathways: either by incorporating the learning component as an auxiliary signal in the model's loss function, or by using the latent representations derived from $\mathcal{Z}$ directly to enhance the model on a representational level.

In this sense, data-based inductive biases do not replace model-based biases but rather complement and enrich them. By leveraging both the structural assumptions embedded in the model through model-based inductive biases and the patterns discovered in the data through data-based inductive biases, we are able to learn better representations that are tailored to the structure and invariances needed for a specific task. Chapter 4 will introduce different types of structure inducing functions, learning component functions and the different ways they can be incorporated into a deep learning model. In the following Chapter 3, we first introduce model-based inductive biases, which make up most of the known deep learning literature.

3

Learning Representations through Model-based Inductive Biases

This chapter introduces the concept of learning representations in deep learning models. As discussed in the previous chapter, a big part of learning representations is essentially determined by model-based inductive biases. That is, the choice of the different components to optimise a deep learning model such as the architecture of the neural network, the loss function and regularisation terms. We first discuss the role of representations in deep learning models and define all components that are needed to train a deep learning model. We then take a closer look at defining a deep learning model that is centered around learning representations by presenting the vanilla autoencoder. Finally, the variational autoencoder is discussed, which makes use of different types of model-based inductive biases to learn representations.

3.1 Representations in Deep Learning

The success of modern deep learning largely stems from neural networks' ability to automatically learn useful representations of data. Unlike traditional machine learning approaches that rely on hand-crafted features, deep learning models discover hierarchical representations that capture increasingly abstract patterns.

In a deep learning model, the input data is transformed into a sequence of intermediate representations that are progressively refined to support the final prediction task. These learned representations serve as the foundation for downstream tasks such as classification, regression, clustering, and anomaly detection. Early deep learning methods focused primarily on end-to-end optimisation for specific tasks, treating representation learning as an implicit byproduct of the training process. However, more recent approaches fundamentally attempt to learn good representations leading to more modular architectures that explicitly optimise for representation quality and transferability across

tasks. As such, one of the central parts of deep learning is to define the components that give origin to learning representations from data.

Let $g_\theta : \mathcal{X} \rightarrow \mathcal{H}$ represent a function parameterized by θ , where $\mathcal{X}$ is the input space and $\mathcal{H}$ is some embedding space. For concreteness, in the following we consider the case where inputs and embeddings are d -dimensional vectors $x \in \mathbb{R}^d$ and $h \in \mathbb{R}^d$. In a deep learning model, the function g_θ is a neural network that transforms the input data into a sequence of intermediate representations before producing the final output.

A neural network with L layers thus defines a sequence of intermediate mappings:

$$h^{(l)} = g_\theta^{(l)}(h^{(l-1)}), \quad l = 1, \dots, L \quad (3.1)$$

where $h^{(l)}$ is the representation at layer l and $g_\theta^{(l)}$ is the mapping defined by the l -th layer of the neural network, where we have $h^{(0)} = x \in \mathbb{R}^{d_0}$ for the first layer of the neural network. The intermediate activations $h^{(l)}$ are then the *learnt representations*.

There are a number of different architectures that can be chosen for g_θ , such as Convolutional Neural Networks [LeCun et al., 2015] (CNNs) for image data, Recurrent Neural Networks [Sherstinsky, 2020] (RNNs) for sequential data, or Transformers [Vaswani et al., 2017b] for language data. The conceptually simplest architecture is the feedforward neural network, which is a sequence of affine transformations followed by a nonlinearity

$$h^{(l)} = \sigma(W_\theta^{(l)} h^{(l-1)} + b_\theta^{(l)}), \quad (3.2)$$

where:

- $W_\theta^{(l)} \in \mathbb{R}^{d_l \times d_{l-1}}$ is the weight matrix of the l -th layer,
- $b_\theta^{(l)} \in \mathbb{R}^{d_l}$ is the bias vector of the l -th layer,
- $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ is the activation function applied element-wise.

While the previously mentioned more complex architectures have additional layers that are designed to capture specific invariances in the data, they follow the same principle of applying a sequence of affine transformations followed by a nonlinearity.

More abstractly, the learned representations $h^{(l)}$ are the result of a sequence of transformations that map the input data into a high-dimensional space. This high-dimensional space then depending on the task allows for linear separability or the emergence of clusters in the data. Formally, we can define this mapping as

$$g : \mathcal{X} \rightarrow \mathcal{H}, \quad (3.3)$$

where $\mathcal{H}$ is the high-dimensional space that the input data is mapped to and g is the mapping that defines the neural network. This view of learned representations as a mapping from the input space to an embedding space is the basis for many unsupervised and self-supervised learning methods that learn representations and allows to draw a connection between learned representations and latent spaces. This will be discussed in more detail in the following sections.

3.2 *Training a Deep Learning Model*

So far, we have discussed the role of representations in deep learning models. To build the foundations to discuss more complex deep learning methods, we will briefly discuss the components of training neural networks, such as how to get a prediction from a neural network and then optimising the network via a loss function and optimisation algorithm. The loss function defines the objective that the neural network is trained to minimise and as such defines the optimisation landscape that the neural network is trained on. The optimisation algorithm then defines how the neural network is updated to minimise the loss function, i.e., navigate the optimisation landscape to find a hopefully acceptable optimum. While the latter is only marginally relevant for the discussion of learned representations, the loss function is crucial as it defines the inductive bias that the neural network is trained on, which is one of the central topics of this work.

3.2.1 *Getting Predictions From A Neural Network*

From Equation 3.1, we see that the output of the neural network is the representation at the last layer $h^{(L)} = g_\theta(x)$. This representation is then mapped to *logits* via an additional affine transformation

$$z = W_{\text{out}_\theta} h^{(L)} + b_{\text{out}_\theta}. \quad (3.4)$$

Hence, if we define f to be the affine transformation of the last layer, for a given input sample x we have

$$z = f_\theta(g_\theta(x)). \quad (3.5)$$

For regression tasks, the logits can be used as is to obtain the final prediction of the neural network. To obtain probabilities from the logits, e.g., for classification tasks, the logits are passed through a softmax function

$$y_i = p(y = i | x) = \text{softmax}(z)_i = \frac{\exp(z_i)}{\sum_{j=1}^C \exp(z_j)}, \quad (3.6)$$

where C is the number of classes in the classification task. For notational simplicity, we refer to the whole neural network including the affine transformation f and the softmax operation as the neural network $\mathcal{F}_\theta$.

3.2.2 Loss Functions

With the predictions from the neural network, we can now define a loss function that quantifies the difference between the predictions and the true output, i.e., a desired output. For simplicity, in the following we consider two vectors $v \in \mathbb{R}^n$ and $w \in \mathbb{R}^n$. Depending on the use case v and w can be the desired logits z and predicted logits $\hat{z}$ (for regression) or the true output y and the predicted output $\hat{y}$ (for classification) respectively. We then define the following loss functions which are commonly used in deep learning models:

- **Mean Squared Error (MSE):** is commonly used for regression tasks and computes the squared difference between the true and predicted output

$$\mathcal{L}_{\text{MSE}}(v, w) = \|v - w\|_2^2 \quad (3.7)$$

$$= (v - w)^T (v - w) \quad (3.8)$$

$$= \frac{1}{n} \sum_{i=1}^n (v_i - w_i)^2, \quad (3.9)$$

where v_i and w_i are the i -th elements of z and $\hat{z}$, respectively. MSE minimizes the Euclidean distance between predictions and true outputs, penalizing large deviations quadratically.

- **Normalised Squared Error (NSE):** The NSE is a normalised version of the MSE and is defined as

$$\mathcal{L}_{\text{NSE}}(v, w) = \left\| \frac{v}{\|v\|_2} - \frac{w}{\|w\|_2} \right\|_2^2 \quad (3.10)$$

$$= 2 - 2 \frac{v^T w}{\|v\|_2 \|w\|_2} \quad (3.11)$$

$$= 2 - 2 \cos(v, w), \quad (3.12)$$

where it can be written in terms of the cosine similarity between v and w . That is, the NSE encourages the model to learn representations that are similar in direction to the true output.

- **Cross-Entropy (CE):** The cross-entropy loss is used for classification tasks. In this case, the difference is computed between two probability distributions

$$\mathcal{L}_{\text{CE}}(v, w) = - \sum_{i=1}^n v_i \log(w_i). \quad (3.13)$$

That is, between the true class v_i encoded as a one-hot vector, with a one entry at the true class and zeros elsewhere, and the predicted probability vector w_i .

3.2.3 Optimisation Algorithms

While the architecture and loss function define the core model-based inductive biases that shape representation learning, the optimization algorithm provides the mechanism through which these representations are actually learned. Although optimization methods are not the primary focus of this work, understanding the basic principles of gradient-based learning is essential for the methods presented in subsequent chapters.

We define a dataset $\mathcal{D}$ consisting of data points $\{x_i, y_i\}_{i=1}^N$, a loss function $\mathcal{L}$, and a neural network F_θ with trainable parameters θ . The goal is to optimize the parameters by minimizing the empirical risk through gradient-based methods that iteratively update parameters in the direction of steepest descent

$$\theta^* = \arg \min_{\theta} \frac{1}{|\mathcal{D}|} \sum_{(x,y) \in \mathcal{D}} \mathcal{L}(\mathcal{F}_\theta(x), y). \quad (3.14)$$

Different optimization algorithms implement this principle with varying degrees of sophistication, incorporating mechanisms such as adaptive learning rates and momentum to improve convergence in the non-convex loss landscapes typical of deep learning.

For completeness, we briefly review the most commonly used optimization algorithms, progressing from simple gradient descent to the adaptive methods employed throughout this work.

- **Vanilla Gradient Descent:** The simplest form of gradient descent is the *vanilla gradient descent* algorithm, which updates the parameters θ via the following update rule,

$$g_t = \frac{1}{|\mathcal{D}|} \sum_{(x,y) \in \mathcal{D}} \nabla_{\theta} \mathcal{L}(\mathcal{F}_\theta(x), y) \Big|_{\theta=\theta_{t-1}} \quad (3.15)$$

$$\theta_t = \theta_{t-1} - \alpha_t g_t \quad (3.16)$$

where $\alpha_t > 0$ is the learning rate and g_t is the gradient of the loss function with respect to the parameters of the neural network at iteration t . The new value of the parameters θ_t is then obtained by subtracting the gradient scaled by the learning rate from the current value of the parameters θ_{t-1} .

- **Stochastic Gradient Descent:** Since the vanilla gradient descent algorithm is computationally expensive for large datasets, in practice

stochastic gradient descent (SGD) is used, which computes the gradient on a mini-batch of the dataset. The update rule for SGD is then

$$g_t = \frac{1}{|\mathcal{B}_t|} \sum_{(x,y) \in \mathcal{B}_t} \nabla_{\theta} \mathcal{L}(\mathcal{F}_{\theta}(x), y) \Big|_{\theta=\theta_{t-1}} \quad (3.17)$$

$$\theta_t = \theta_{t-1} - \alpha_t g_t \quad (3.18)$$

where $\mathcal{B}$ is a mini-batch of the dataset, that is $\mathcal{B} \subset \mathcal{D}$, a subset of the dataset.

While stochastic gradient descent is conceptually simple, the fixed learning rate α does not adapt to the curvature of the loss function, which can lead to slow convergence or divergence, thus requiring careful hyperparameter tuning. Another issue is that the optimisation landscape of deep learning models is noisy and non-convex, which can lead to oscillations and reduced convergence speed. To improve this, a range of methods have been proposed that smoothen the parameter updates via a moving average and adapt the learning rate to consider the loss function curvature. We will discuss these methods next.

- **Momentum:** is a method that accelerates the convergence of the gradient descent algorithm by adding a fraction β of the previous update to the current update. The idea is that the momentum term helps to overcome local minima by smoothing the parameter updates, thus preventing oscillations and speeding up convergence. The update rule for the momentum algorithm is defined as follows,

$$\mu_t = \beta \mu_{t-1} + \frac{1}{|\mathcal{B}_t|} \sum_{(x,y) \in \mathcal{B}_t} \nabla_{\theta} \mathcal{L}(\mathcal{F}_{\theta}, y) \Big|_{\theta=\theta_{t-1}} \quad (3.19)$$

$$\theta_t = \theta_{t-1} - \alpha_t \mu_t \quad (3.20)$$

where v_t is the momentum term and $1 > \beta \geq 0$ is the momentum hyperparameter and μ_0 is initialised to 0. The larger the value of β , the more influence the previous updates have on the current update. Note that if β is 0 the momentum algorithm reduces to stochastic gradient descent.

- **RMSProp:** introduces an adaptive learning rate that scales the gradient by the square root of a moving average of the squared gradients. The idea is that the learning rate is adapted to the curvature of the loss function, such that the learning rate is reduced for steep directions and increased for flat directions. The update rule for RMSProp

is defined as follows,

$$g_t = \frac{1}{|\mathcal{B}_t|} \sum_{(x,y) \in \mathcal{B}_t} \nabla_{\theta} \mathcal{L}(\mathcal{F}_{\theta}(x), y) \Big|_{\theta=\theta_{t-1}} \quad (3.21)$$

$$v_t = \beta v_{t-1} + (1 - \beta) g_t^2 \quad (3.22)$$

$$\theta_t = \theta_{t-1} - \alpha_t \frac{g_t}{\sqrt{v_t + \epsilon}} \quad (3.23)$$

where v_t is the moving average of the squared gradients, β controls the influence of more recent squared gradients, and ϵ is a small constant to prevent division by zero. The momentum term for the squared gradient is initialised to zero, i.e., $v_0 = 0$. While RMSProp is already a powerful optimisation algorithm, it can be further improved by the Adam algorithm, which combines the momentum and RMSProp algorithms.

- **Adam [Kingma, 2014]:** combines the momentum and RMSProp algorithms by adding a moving average of the gradients and the squared gradients as follows,

$$g_t = \frac{1}{|\mathcal{B}_t|} \sum_{(x,y) \in \mathcal{B}_t} \nabla_{\theta} \mathcal{L}(f_{\theta}(x), y) \Big|_{\theta=\theta_{t-1}} \quad (3.24)$$

$$\mu_t = \beta_1 \mu_{t-1} + (1 - \beta_1) g_t \quad (3.25)$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \quad (3.26)$$

$$\theta_t = \theta_{t-1} - \alpha_t \frac{\mu_t}{\sqrt{v_t + \epsilon}} \quad (3.27)$$

where μ_t is the moving average of the gradients, v_t is the moving average of the squared gradients, β_1 and β_2 control the influence of more recent gradients and squared gradients, respectively, and ϵ is a small constant to prevent division by zero. The momentum terms for the gradients and squared gradients are initialised to zero, i.e., $\mu_0 = 0$ and $v_0 = 0$. The Adam algorithm is one of the most widely used optimisation algorithms in deep learning and is known for its robustness and fast convergence. In this work all proposed methods are trained with the Adam optimisation algorithm.

The evolution from vanilla gradient descent to adaptive methods like Adam reflects the development of increasingly sophisticated mechanisms for navigating the complex loss landscapes of deep neural networks. While simple gradient descent treats all parameters uniformly, momentum smoothes parameter updates by incorporating gradient history. RMSProp adapts learning rates to local curvature, and Adam combines both mechanisms to create a robust optimizer that works well across diverse architectures and tasks. These optimization algorithms provide the computational foundation for training the neural networks

used throughout this thesis. The choice of Adam as our standard optimizer ensures consistent and reliable parameter updates across all experiments.

3.2.4 Regularisation

Regularisation usually describes a set of techniques that are used to avoid the model from fitting its parameters too closely to the training data, i.e. to the noise in the training data. Such regularisation techniques can either be applied directly to the model parameters or to the loss function. For clarity, in the following we define the output of the network as y and the true output as y' , while z is the intermediate mapping from Equations 3.6 and 3.4 respectively. Then the general structure of such regularisation techniques can be written in terms of the loss function,

$$\mathcal{L}_{\text{reg}}(y, y', \theta) = \mathcal{L}(y, y') + \lambda \mathcal{R}(\theta), \quad (3.28)$$

where $\mathcal{L}$ is the loss function, $\mathcal{R}$ is a regularisation function $\mathcal{R} : \mathbb{R}^n \rightarrow \mathbb{R}$ that takes the flattened model's parameters as input and outputs a scalar. Then, λ is a scalar that controls the regularisation strength. There are several examples of this type of regularisation, such as L1 and L2 regularisation, where $\mathcal{R}$ is given by the L1 or L2 norm of the parameters, respectively.

A more recent approach is to use regularisation techniques that operate on the learnt representations themselves, rather than just the model parameters

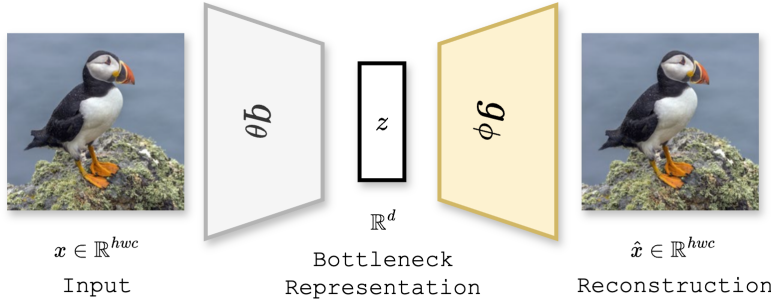
$$\mathcal{L}_{\text{reg}}(y, y', z, z') = \mathcal{L}(y, y') + \lambda \mathcal{R}(z, z'). \quad (3.29)$$

Here, $\mathcal{R}$ is a function that compares the intermediate representation z to some reference representation z' . This reference may come from the same model with frozen parameters (e.g., a teacher model) or from a reference distribution serving as prior. Instead of preventing overfitting through parameter constraints, this approach discourages the model from learning trivial or uninformative representations.

This representation-based regularisation framework naturally connects to data-based inductive biases. However, rather than using heuristic regularisation terms, structure discovered from the data itself can guide representation learning. As we will see in Chapter 4, self-supervised learning methods exemplify this approach by using data transformations to create reference representations that encode meaningful invariances, effectively combining model-based regularisation with data-based inductive biases.

3.3 The (Variational) Autoencoders Strong Model-based Inductive Biases

Autoencoder



Variational Autoencoder

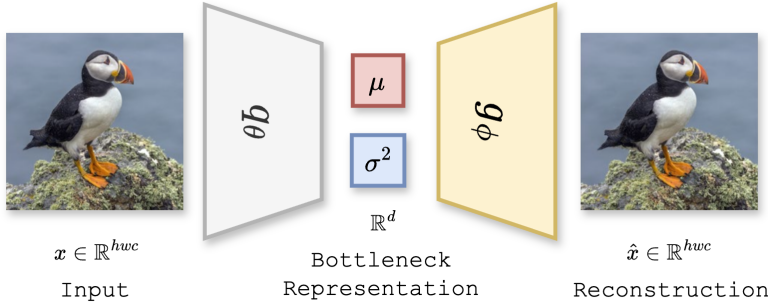


Figure 3.1: Autoencoder vs Variational Autoencoder architecture: The autoencoder is a simple deep learning model that learns representations by reconstructing the input data. The variational autoencoder is an extension of the autoencoder that learns representations by reconstructing the input data and assumes the data can be modelled under a Gaussian distribution. The variational autoencoder makes use of different model-based inductive biases to learn a Gaussian latent space.

Having discussed the foundations of training neural networks, we can now introduce deep learning models that focus on learning representations more explicitly by incorporating specific model-based inductive biases to learn representations. We will first discuss the autoencoder, which is a simple deep learning model that learns representations by reconstructing the input data. The autoencoder is a great example of using model-based inductive biases to restrict the space of functions a model can represent. In the case of the traditional autoencoder this is restricted by the reconstruction objective, which enforces that the model must reconstruct its input. An extension of the autoencoder, the variational autoencoder includes architectural and regularisation constraints that assume the data can be modelled under a Gaussian distribution. In the next chapter it will be shown that the variational autoencoder is a powerful latent variable model which can extract structure from high-dimensional data such as images.

3.3.1 The Autoencoder: Towards Inducing Structure in Representations

The autoencoder is a simple deep learning model that learns representations through a bottleneck in the network, i.e., a layer with fewer units than the input and output layers, which forces the network to learn a compressed representation of the input data. The autoencoder is trained by minimising the difference between the input data and the reconstructed data, which is called the reconstruction loss, where the reconstruction loss is a measure of how well the autoencoder can reconstruct the input data from the learned representation.

More precisely, as shown in Figure 3.1, the autoencoder consists of two parts: an *encoder* and a *decoder*.

- The **encoder** q_θ maps the input x to a compressed representation $z = q_\theta(x)$.
- The **decoder** g_ϕ maps the compressed representation back to the input data by reconstructing the input $\hat{x} = g_\phi(z)$.

The encoder can be seen as the mapping Φ from Equation 3.3 that maps the input data into a representation space that is in this case lower-dimensional than the input data. The autoencoder is then trained by minimising the difference between the input data and the reconstructed data,

$$\mathcal{L}_{\text{AE}}^{\theta,\phi}(x_i, \hat{x}_i) = \mathcal{L}_{\text{MSE}}^{\theta,\phi}(x_i, \hat{x}_i) = \|x_i - \hat{x}_i\|_2^2, \quad (3.30)$$

where $\mathcal{L}_{\text{MSE}}$ is the mean squared error loss function.

After training, the bottleneck representation y encodes a low-dimensional representation of the input data x . The bottleneck, together with the reconstruction loss, encourages the autoencoder to learn a representation that groups similar data points together and separates dissimilar ones. However, in the case of a simple autoencoder, the learnt representation space is not explicitly regularised and can lack smoothness and structure. This may result in regions of the latent space that do not correspond to “valid” data, making these representations less suitable for downstream tasks.

Next, we present the variational autoencoder, which extends the traditional autoencoder into a latent variable model through model-based inductive biases. This allows the autoencoder to learn a structured, smooth, and continuous representation space that generalises better and enables meaningful interpolation.

3.3.2 The Variational Autoencoder: From a Model-based Inductive Bias Perspective

The Variational Autoencoder is actually a latent variable model which explicitly models the latent space as a Gaussian distribution which is a common assumption for latent variable models. We will explore the components of the variational autoencoder from a latent variable modelling perspective in Chapter 4. However, in order to learn a latent variable model that assumes a Gaussian distribution, the variational autoencoder makes use of several model-based inductive biases to project the high-dimensional input data into a Gaussian latent space. In the following, we will explore three different model-based inductive biases that are incorporated into the architecture and learning objective to achieve this.

1. Gaussian bottleneck. In a variational autoencoder the encoder f_θ explicitly models the parameters of a Gaussian distribution, μ and σ^2 , as shown in Figure 3.1 (bottom). The encoder output is then used to parametrise a Gaussian distribution,

$$q_\theta(z|x) = \mathcal{N}(z|\mu, \sigma^2). \quad (3.31)$$

which is an approximation of the posterior distribution over the latent variables z . More precisely, let h_μ and h_{σ^2} be two final layers after the encoder f_θ , then the parameters of the Gaussian distribution are given by the outputs of the encoder,

$$\mu = h_\mu(q_\theta(x)), \quad \sigma^2 = h_{\sigma^2}(q_\theta(x)). \quad (3.32)$$

We will explain the mathematical details of this in Chapter 4 in the context of latent variable models. For now, we see that there is an explicit model-based inductive bias where a neural network is used to explicitly model the parameters of a Gaussian distribution.

2. Gaussian prior. Another strong model-based inductive bias is the regularisation of the model parameters with a standard normal Gaussian prior. The loss function of the variational autoencoder is given by

$$\mathbb{E}_{z \sim f_\theta(z|x)} \log g_\phi(x|z) - \beta D_{KL}(q_\theta(z|x) \| p(z)), \quad (3.33)$$

where $\log g_\phi(x|z)$ is the likelihood of the data given the latent variables and D_{KL} is the Kullback-Leibler divergence. The first term can be interpreted as a reconstruction loss, similar in spirit to the objective in a vanilla autoencoder. However, in the Variational Autoencoder, the likelihood term typically assumes a Gaussian distribution over the data. Moreover, the reconstruction term is expressed as an expectation over

the approximate posterior, which is not present in vanilla autoencoders. In practice, this expectation is usually approximated using a single sample from the posterior, made tractable via the reparameterisation trick.

The second term regularises the approximate posterior $q_\theta(z|x)$ to stay close to the prior distribution $p(z)$, typically chosen as a standard normal $\mathcal{N}(0, I)$. This encourages the latent space to be smooth and continuous, and enables meaningful sampling and interpolation in the latent space. The regularisation term is weighted by β , hence controlling the influence of the latent space to be close to the prior distribution.

3. Reparameterisation trick. Training a variational autoencoder involves learning a distribution over the latent variables z , typically modeled as a Gaussian. However, directly sampling from this distribution introduces a non-differentiable operation, which prevents the use of standard backpropagation. To address this, the *reparameterisation trick* is employed, which expresses the sampling operation in a differentiable way by rewriting the latent variable as:

$$z = \mu + \sigma^2 \odot \epsilon, \quad (3.34)$$

where $\epsilon \sim \mathcal{N}(0, I)$, and μ and σ^2 are the outputs of the encoder network that parameterise the approximate posterior distribution $q_\theta(z|x)$.

This formulation isolates the stochasticity in ϵ , while allowing gradients to propagate through μ and σ^2 . In essence, we transform a sample from the standard normal distribution by scaling and shifting it according to the learned parameters. This enables efficient and differentiable end-to-end training of the VAE using gradient descent.

These three model-based inductive biases, the Gaussian bottleneck, the Gaussian prior and the reparameterisation trick, allow the variational autoencoder to learn a continuous and smooth latent space over the input data, modelled as a Gaussian distribution as opposed to the discrete latent spaces of the vanilla autoencoder. The variational autoencoder is thus a good example of how model-based inductive biases can be used to embed the assumption of Gaussianity into a deep learning model.

In the next chapter, we will introduce the concept of data-based inductive biases and discuss how model-based inductive biases can be augmented with data-based inductive biases to introduce more information into the model. For this purpose, we will discuss the role of latent variable models as structure inducing functions T and how they can be used to extract information from the data that otherwise cannot be learnt by the more general model-based inductive biases. Here, variational autoencoders will be presented again from a latent variable modelling perspective.

4

Leveraging Latent Spaces for Data-based Inductive Biases

In the previous chapter, we explored how model-based inductive biases enable representation learning in deep neural networks. While these types of biases are well-established in the literature, data-based inductive biases offer a complementary approach to guiding model learning.

The key insight underlying data-based inductive biases is that meaningful structure exists within data, and this structure can be systematically extracted and transformed into learning signals that enhance model performance. This structure can be leveraged in two primary ways: through expert knowledge when domain expertise is available or through systematic discovery using latent variable modeling techniques that automatically uncover structure from the data.

When working with raw data, extracting meaningful structure can be challenging and may require preprocessing or dimensionality reduction. However, when pre-trained representations are available, their inherent structure provides a rich foundation for defining latent spaces through latent variable modeling approaches.

Transforming discovered latent structure into actionable learning signals presents its own challenges. These signals can be incorporated into the learning process in multiple ways: they may augment existing model-based inductive biases within the learning objective, or they may be directly integrated into the model's learned representations.

This chapter provides a comprehensive framework for understanding and implementing data-based inductive biases. We begin by introducing the fundamental concepts of latent variables and latent spaces, establishing the mathematical foundation for structure discovery. We then examine latent variable modeling techniques, including Gaussian mixture models and Variational Autoencoders, demonstrating how these approaches can automatically learn meaningful latent representations. The subsequent sections detail how to transform latent structure into effective learning components and integrate these components into model training. Finally, we explore how self-supervised learning

methods exemplify the combination of model-based and data-based inductive biases.

4.1 Latent Variables and Latent Spaces

The first step to introducing data-based inductive biases is to define the structure inducing function

$$T : \mathcal{X} \rightarrow \mathcal{Z}, \quad (4.1)$$

which maps the data space $\mathcal{X}$ to a latent space $\mathcal{Z}$. For this, we define the concept of a latent space and introduce latent variable models. We show that latent variable models allow for learning latent spaces in an unsupervised manner with minimal assumptions on the data.

In general, learning latent variables assumes that the data is generated by some underlying process that is not directly observable. The idea is that instead of generating the data from known causes, we observe the effects and aim to infer the hidden causes. These hidden causes are called *latent variables* and can be thought of as the underlying factors that generate the data.

More precisely, given that we assume observed data is generated by a set of latent variables $\mathbf{z} \in \mathbb{R}^d$ sampled from a latent space $\mathcal{Z}$, we can define a generative model that describes the process of generating data $x \in \mathbb{R}^{\mathcal{D}}$ from the latent variables where $d \ll \mathcal{D}$

$$\mathbf{z} \sim p(\mathbf{z}), \quad x \sim p(\mathbf{x}|\mathbf{z}). \quad (4.2)$$

Then using Bayes' theorem, we can define a posterior distribution over the latent variables given the data

$$p(\mathbf{z}|\mathbf{x}) = \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{p(\mathbf{x})}, \quad (4.3)$$

where $p(\mathbf{z})$ is a prior distribution over the latent variables, $p(\mathbf{x}|\mathbf{z})$ is the likelihood of the data given the latent variables and $p(\mathbf{x})$ is the marginal likelihood of the data.

In practice, $p(\mathbf{z})$ can be chosen to be a simple distribution, such as a categorical or Gaussian distribution, while $p(\mathbf{x}|\mathbf{z})$ models the data generation process conditioned on the latent variables, and is typically selected based on the nature of the data. For example, a Gaussian distribution for continuous data, a Bernoulli for binary data, or a neural network decoder in deep generative models.

There are two quantities of interest in the generative model: the marginal distribution $p(\mathbf{x})$ and the posterior distribution $p(\mathbf{z}|\mathbf{x})$. The marginal distribution $p(\mathbf{x})$ describes the probability of observing any data point under our model. However, for the structure inducing

function T , our primary interest is the posterior distribution $p(\mathbf{z}|\mathbf{x})$, which requires computing $p(\mathbf{x})$ via Bayes' theorem.

Determining $p(\mathbf{x})$ is straightforward if the latents are assumed to be discrete (e.g., categorical), however, if the prior is continuous (e.g., Gaussian), computing $p(\mathbf{x})$ becomes intractable, as it involves integrating over all possible values of $\mathbf{z}$, which often cannot be done analytically. In such cases, the posterior distribution $p(\mathbf{z}|\mathbf{x})$ is approximated using variational inference techniques optimizing a lower bound on the log marginal likelihood $\log p(\mathbf{x})$. This will be discussed in more detail in the next sections.

One of the important design decisions with regards to the structure inducing function T is whether we want to model a **discrete** or **continuous** latent space. We will see that this impacts the nature of the learning component function which transforms the latent space into an actionable learning signal. In the following, we present two latent variable models: the Gaussian Mixture Model which models a discrete latent space and the Variational Autoencoder which models a continuous latent space.

4.2 Learning Latent Spaces with Latent Variable Models

Latent variable models learn latent spaces by defining a generative model over the data, which assumes an underlying distribution over the latent variables. This allows us to systematically extract structure from data in an unsupervised manner, making latent variable models a natural choice as structure inducing function T to generate data-based inductive biases.

In this sense, latent variable models attempt to model each component of the generative model (see Equation 4.2) to approximate two key quantities of interest: the *posterior distribution* $p(\mathbf{z}|\mathbf{x})$ (see Equation 4.3), i.e., the distribution of the latent variables given the data, and the *marginal distribution* $p(\mathbf{x})$ (see Equation 4.4), which describes the data distribution. Modelling both the posterior distribution $p(\mathbf{z}|\mathbf{x})$ and the marginal distribution $p(\mathbf{x})$ can be challenging, depending on the choice of the prior $p(\mathbf{z})$ and the likelihood $p(\mathbf{x}|\mathbf{z})$, since the marginal likelihood $p(\mathbf{x})$

$$p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z})p(\mathbf{z}) d\mathbf{z}, \quad (4.4)$$

is intractable for most choices of continuous distributions for $p(\mathbf{z})$ and $p(\mathbf{x}|\mathbf{z})$. In this work, we focus on Gaussian mixture models (GMMs) and the previously introduced Variational Autoencoder (VAE), since they allow for learning latent spaces from complex data.

GMMs assume the generative process is a mixture of Gaussians, where the prior $p(\mathbf{z})$ is a categorical distribution and the likelihood

$p(\mathbf{x}|\mathbf{z})$ is a Gaussian distribution. Since the parameters of the GMM describing the latent space are discrete, the marginal distribution $p(\mathbf{x})$ is tractable. It is thus straightforward to learn a latent space even on relatively complex data. VAEs on the other hand are suited for learning latent spaces over high-dimensional data, however, the prior $p(\mathbf{z})$ is a continuous Gaussian distribution and the likelihood $p(\mathbf{x}|\mathbf{z})$ is parameterized by a neural network, which makes $p(\mathbf{x})$ intractable. A solution to this in practice, is to calculate an approximation of the marginal likelihood $p(\mathbf{x})$ using a variational lower bound, which implicitly approximates the posterior distribution $p(\mathbf{z}|\mathbf{x})$ via a surrogate distribution $q(\mathbf{z}|\mathbf{x})$. We examine both approaches in detail below, beginning with GMMs where exact inference is possible, then exploring how VAEs handle the intractable case through variational approximation.

4.2.1 Gaussian Mixture Models

A Gaussian mixture model (GMM) is a latent variable model that assumes a categorical prior $p(\mathbf{z})$ and a Gaussian likelihood $p(\mathbf{x}|\mathbf{z})$ for its generative process. The prior is defined over so called components, where each component is sampled from a categorical distribution. More precisely, for a GMM with K components, we have

- K : the number of components,
- $z \in \{1, \dots, K\}$: is a latent variable representing the component index.

Furthermore, each component is described by a multivariate Gaussian distribution,

- $\mu_k \in \mathbb{R}^d$: the mean of the k -th component,
- $\Sigma_k \in \mathbb{R}^{d \times d}$: the covariance matrix of the k -th component,
- $\pi_k \in \mathbb{R}$: the mixing coefficient of the k -th component.

The marginal distribution that describes the data is then given by

$$p(\mathbf{x}) = \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}|\mu_k, \Sigma_k), \quad (4.5)$$

where we average over all components weighted by the mixing coefficients π_k , essentially a mixture of Gaussians. Note that this is the discrete case of the marginal distribution in Equation 4.4, where $\pi_k = p(\mathbf{z} = k)$ and $\mathcal{N}(\mathbf{x}|\mu_k, \Sigma_k) = p(\mathbf{x}|\mathbf{z} = k)$. The GMM then optimises the maximum likelihood estimate of the parameters $\theta = \{\pi_k, \mu_k, \Sigma_k\}_{k=1}^K$ to maximise the log-likelihood of the data

$$\log p(\mathbf{x}) = \log \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}|\mu_k, \Sigma_k). \quad (4.6)$$

Since this log-likelihood can be computed analytically, an EM algorithm can be used to learn the parameters of the GMM that maximise the log-likelihood of the data. This entails updating the parameters of the components and the mixing coefficients iteratively.

EM algorithm for GMMs. Given a dataset $\mathcal{D} = \{x_i\}_{i=1}^N$, we want to learn the parameters $\theta = \{\pi_k, \mu_k, \Sigma_k\}_{k=1}^K$ of the GMM. We do this with the Expectation-Maximisation (EM) algorithm, which alternates between an E-step and an M-step.

E-step: The E-step computes the posterior probabilities $p(\mathbf{z} = k|x_i)$ for each data point x_i using the current parameters θ

$$\gamma_{ik} = p(\mathbf{z} = k|x_i) = \frac{\pi_k \mathcal{N}(x_i|\mu_k, \Sigma_k)}{\sum_{j=1}^K \pi_j \mathcal{N}(x_i|\mu_j, \Sigma_j)}, \quad (4.7)$$

where $\gamma_{ik} \in [0, 1]$ and $\sum_{k=1}^K \gamma_{ik} = 1$. This is equivalent to evaluating the posterior distribution $p(\mathbf{z}|x_i)$ in Equation 4.3 for each data point x_i and component k . With these posteriors over the latent variables, we can now update the parameters θ in the M-step.

M-step: The M-step updates the parameters θ using the current posterior probabilities $p(\mathbf{z} = k|x_i)$.

1. Calculate the total probability mass assigned to each component k :

$$N_k = \sum_{i=1}^N \gamma_{ik}. \quad (4.8)$$

2. Update the mixing coefficients π_k based on the total probability mass:

$$\pi_k = \frac{N_k}{N}. \quad (4.9)$$

3. Update the means μ_k :

$$\mu_k = \frac{1}{N_k} \sum_{i=1}^N \gamma_{ik} x_i. \quad (4.10)$$

4. Update the covariance matrices Σ_k :

$$\Sigma_k = \frac{1}{N_k} \sum_{i=1}^N \gamma_{ik} (x_i - \mu_k)(x_i - \mu_k)^T. \quad (4.11)$$

The process is repeated until the log-likelihood of the data converges, i.e., stops improving according to some threshold.

The posteriors γ_{ik} represent the responsibility of each component k for generating data point x_i , effectively defining the latent space of the GMM. Through this mechanism, the data is modeled as a mixture of

Gaussians, where each data point can belong to multiple components with varying probabilities.

GMMs exemplify how latent variable models can serve as structure inducing functions T for data-based inductive biases. They are particularly effective when data lies on a low-dimensional manifold with cluster-like structure. In Chapter 6, we demonstrate this by using GMMs to learn latent spaces over pre-trained representations for enhanced exploration in reinforcement learning.

4.2.2 Variational Autoencoders (Revisited)

In Chapter 3, we introduced the variational autoencoder (VAE) as a deep learning model that enforces Gaussian structure on learned representations. Here, we revisit VAEs from the latent variable modeling perspective, highlighting how they handle the intractable inference problem that arises with continuous latent variables.

Unlike GMMs, where the discrete latent variables make exact inference tractable, VAEs face a fundamental challenge: the likelihood $p(\mathbf{x}|\mathbf{z})$ is parameterized by a neural network, making the marginal distribution $p(\mathbf{x})$ intractable to compute. This means we cannot directly evaluate the posterior distribution $p(\mathbf{z}|\mathbf{x})$ using Bayes' theorem.

The VAE's solution is elegant: instead of computing the true posterior $p(\mathbf{z}|\mathbf{x})$ directly, it learns an approximate posterior $q(\mathbf{z}|\mathbf{x})$ that is both tractable and close to the true posterior. This approximation is achieved by optimizing a lower bound on the log marginal likelihood, known as the Evidence Lower Bound (ELBO).

Deriving the evidence lower bound (ELBO). The key insight is to introduce the approximate posterior $q(\mathbf{z}|\mathbf{x})$ and use it to rewrite the intractable log marginal likelihood in a tractable form. We begin with the log marginal likelihood of the data:

$$\log p(\mathbf{x}) = \log \int p(\mathbf{x}, \mathbf{z}) d\mathbf{z} = \log \int p(\mathbf{x}|\mathbf{z}) p(\mathbf{z}) d\mathbf{z}, \quad (4.12)$$

which as discussed previously is intractable to compute because $p(\mathbf{x}|\mathbf{z})$ is parameterized by a neural network. We thus introduce an approximate posterior distribution $q(\mathbf{z}|\mathbf{x})$ and rewrite the log marginal likelihood as:

$$\log p(\mathbf{x}) = \log \int q(\mathbf{z}|\mathbf{x}) \frac{p(\mathbf{x}, \mathbf{z})}{q(\mathbf{z}|\mathbf{x})} d\mathbf{z} = \log \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})} \left[\frac{p(\mathbf{x}, \mathbf{z})}{q(\mathbf{z}|\mathbf{x})} \right]. \quad (4.13)$$

We now apply Jensen's inequality to move the logarithm inside the expectation, which yields:

$$\log p(\mathbf{x}) = \log \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})} \left[\frac{p(\mathbf{x}, \mathbf{z})}{q(\mathbf{z}|\mathbf{x})} \right] \geq \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})} \left[\log \frac{p(\mathbf{x}, \mathbf{z})}{q(\mathbf{z}|\mathbf{x})} \right]. \quad (4.14)$$

This is the so called Evidence Lower Bound (ELBO), which is a lower bound on the log marginal likelihood. Now we can rewrite the ELBO in a way to compute its components, we can write out the joint $p(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}|\mathbf{z})p(\mathbf{z})$ and thus get:

$$\text{ELBO}(\mathbf{x}) = \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})} [\log p(\mathbf{x}|\mathbf{z}) + \log p(\mathbf{z}) - \log q(\mathbf{z}|\mathbf{x})] \quad (4.15)$$

$$= \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})} [\log p(\mathbf{x}|\mathbf{z})] - \text{KL}(q(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z})). \quad (4.16)$$

The first term is the expected log-likelihood of the data under the approximate posterior distribution $q(\mathbf{z}|\mathbf{x})$, and is usually called the reconstruction term. The second term is the KL divergence between the approximate posterior distribution $q(\mathbf{z}|\mathbf{x})$ and the prior distribution $p(\mathbf{z})$, acting as a regularization term. This is the same Equation 3.33 presented in Chapter 3.

Why maximizing the ELBO works. While we now know that the ELBO is a lower bound on the log marginal likelihood, we can also show that maximizing the ELBO is equivalent to minimizing the KL divergence between the approximate posterior distribution $q(\mathbf{z}|\mathbf{x})$ and the true posterior distribution $p(\mathbf{z}|\mathbf{x})$.

For this purpose, we define the KL-divergence between the true posterior distribution $p(\mathbf{z}|\mathbf{x})$ and the approximate posterior distribution $q(\mathbf{z}|\mathbf{x})$ as:

$$\text{KL}(q(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z}|\mathbf{x})) = \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})} \left[\log \frac{q(\mathbf{z}|\mathbf{x})}{p(\mathbf{z}|\mathbf{x})} \right]. \quad (4.17)$$

We then substitute Bayes' rule into the denominator of the right-hand side to obtain:

$$\text{KL}(q(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z}|\mathbf{x})) = \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})} \left[\log \frac{q(\mathbf{z}|\mathbf{x})}{p(\mathbf{z}|\mathbf{x})} \right] \quad (4.18)$$

$$= \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})} \left[\log \frac{q(\mathbf{z}|\mathbf{x})}{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})/p(\mathbf{x})} \right] \quad (4.19)$$

$$= \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})} \left[\log \frac{q(\mathbf{z}|\mathbf{x})}{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})} + \log p(\mathbf{x}) \right] \quad (4.20)$$

$$= \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})} [\log q(\mathbf{z}|\mathbf{x}) - \log p(\mathbf{x}|\mathbf{z}) - \log p(\mathbf{z})] + \log p(\mathbf{x}). \quad (4.21)$$

We can now group the expression in terms of the KL-divergence:

$$\text{KL}(q(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z}|\mathbf{x})) = -\mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})} [\log p(\mathbf{x}|\mathbf{z})] + \text{KL}(q(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z})) + \log p(\mathbf{x}). \quad (4.22)$$

Then isolating $\log p(\mathbf{x})$ on the left-hand side, we obtain:

$$\log p(\mathbf{x}) = \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})} [\log p(\mathbf{x}|\mathbf{z})] - \text{KL}(q(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z})) + \text{KL}(q(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z}|\mathbf{x})), \quad (4.23)$$

which is the exact decomposition of the log marginal likelihood into the Evidence Lower Bound (ELBO) and the KL-divergence between the approximate posterior distribution $q(\mathbf{z}|\mathbf{x})$ and the true posterior distribution $p(\mathbf{z}|\mathbf{x})$:

$$\log p(\mathbf{x}) = \text{ELBO}(x) + \text{KL}(q(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}|\mathbf{x})). \quad (4.24)$$

Since the KL-divergence is non-negative the above constitutes a lower bound as shown in Equation 4.14. Thus, the ELBO is exactly the log marginal likelihood of the data if p and q are equal. In the same sense, by rearranging the expression in terms of the KL-divergence

$$\text{KL}(q(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}|\mathbf{x})) = \log p(\mathbf{x}) - \text{ELBO}(x), \quad (4.25)$$

we see that maximizing the ELBO is equivalent to minimizing the KL-divergence between the approximate posterior distribution $q(\mathbf{z}|\mathbf{x})$ and the true posterior distribution $p(\mathbf{z}|\mathbf{x})$.

Connection to the Latent Variable Framework. Despite the complexity of the variational approximation, VAEs follow the same fundamental latent variable modeling framework as GMMs. Both methods:

1. Define a generative model $p(\mathbf{x}|\mathbf{z})p(\mathbf{z})$
2. Aim to learn the posterior distribution $p(\mathbf{z}|\mathbf{x})$
3. Maximize the marginal likelihood $p(\mathbf{x})$ (exactly for GMMs, approximately for VAEs)

The key difference is in tractability: GMMs can compute everything exactly due to discrete latents, while VAEs require variational approximation for continuous latents but can handle much more complex, high-dimensional data.

Summary. Latent variable models provide a unified framework for learning structured representations in an unsupervised manner. The choice between GMMs and VAEs depends on whether the latent space should be discrete or continuous:

- **GMMs:** Best for lower-dimensional data with clear cluster structure, where exact inference is possible.
- **VAEs:** Essential for high-dimensional data (like images) where the flexibility of neural networks is needed.

Both serve effectively as structure inducing functions T for data-based inductive biases.

4.3 Inducing Latent Spaces with Transformations On Data

While latent variable models provide a principled approach to learning structured latent spaces directly from data, in many practical scenarios, data transformations can also serve as an effective way to induce latent structure. These transformations can either enhance the learning process of latent variable models or directly define the latent structure themselves, with a neural network projecting the transformed data into a structured latent space.

The structure inducing function T can leverage data transformations to expose or create underlying structure. This can be formalized as:

$$T(\mathcal{X}) = \mathcal{F}_\theta(t(\mathcal{X})), \quad (4.26)$$

where t represents a data transformation and $\mathcal{F}_\theta$ is the model inducing a latent space. Note that $\mathcal{F}_\theta$ in this case can be any function such as a neural network or a latent variable model.

Data transformations can take various forms, from simple preprocessing steps to sophisticated domain-specific operations. The key insight is that these transformations can expose structure that might be difficult to discover directly from raw data, or they can explicitly encode domain knowledge about relevant invariances and relationships.

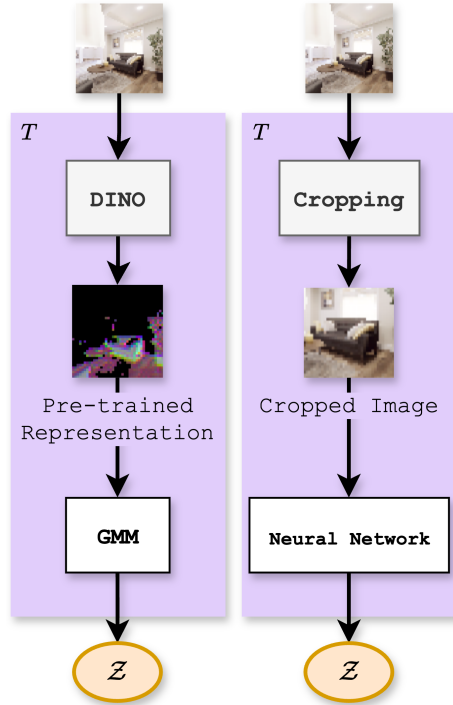


Figure 4.1: **Examples of data transformations in practice:** These examples show how data transformations can help shape the structure inducing function T . Pre-trained representations can be used to extract features from raw data, which can then be leveraged into a latent space with a latent variable model such as a GMM. More simple transformations such as data augmentations can be used to expose invariances in the data, after which a (pretrained) neural network can be used to embed the transformed data into a latent space. Such combinations arise often in practice.

Examples of data transformations in practice. The methods presented in this thesis demonstrate key ways in which data transformations can induce structure:

- **Pre-trained representations:** Using pre-trained models to extract features from raw data serves as a transformation that exposes semantic structure. For instance, extracting representations from pre-trained vision models provides a structured feature space that can be directly used or further processed by latent variable models to model a discrete latent space.
- **Data augmentations:** Data augmentations serve as transformations that expose invariances in the data, after which neural networks or latent variable models can learn representations that capture semantic relationships across different views of the same underlying content.

While data transformations can directly induce structure, they rely on data transformations that incorporate expert knowledge about relevant structure and invariances, with subsequent models primarily serving to embed the transformed data into a suitable representation space. Compared to latent variable modelling this approach still has its merits: transformation-based approaches can be highly effective when domain knowledge about relevant structure is available and can be encoded through appropriate transformations.

4.4 Leveraging Latent Spaces Into a Learning Component

The structure induced by the latent space can now be used to create a learning component for the data-based inductive bias via the learning component function

$$C : \mathcal{Z} \rightarrow \mathcal{B}, \quad (4.27)$$

where $\mathcal{B}$ defines the space of learning components. A learning component $b \in \mathcal{B}$ represents the transformation of the latent structure $\mathcal{Z}$, specifically $z \in \mathcal{Z}$, into a form that can be effectively used to enhance the model. Integration strategies for this learning component into the model then largely depend on whether the induced latent space by the structure inducing function T is discrete or continuous.

4.4.1 Discrete learning components

We define the simplest learning component as a discrete assignment

$$C : \{1, \dots, K\} \rightarrow \mathbb{R}^K,$$

where $b = C(z)$ returns a vector of counts for each discrete latent cluster k . This discrete index exposes the cluster structure of the latent space.

The resulting cluster labels can then be used as weights for counting occurrences or masking. More generally, such an integration can be realized as an additive term to the overall loss function:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{task}} + \lambda b, \quad (4.28)$$

where $\mathcal{L}_{\text{task}}$ is the primary task-specific loss, λ is a hyperparameter controlling the influence of the inductive bias, and $b = C(z)$ is the scalar learning component applied to the latent representation z .

In our work on *cyclophobic reinforcement learning* (Chapter 5), the learning component function C produces an intrinsic reward by combining hierarchical views of the environment together with a cycle-avoidance signal that penalises state revisitations. Formally, C aggregates structural information over multiple hierarchical observations z to produce a scalar reward signal that guides exploration toward unexplored regions

$$r_{\text{total}}(s, a) = r_{\text{extrinsic}}(s, a) + \lambda C(s, a, z), \quad (4.29)$$

where $r_{\text{extrinsic}}$ is the environment-provided reward, and $C(s, a, z) = r_{\text{cyclophobic}}(s, a, z)$ quantifies the cycle-avoidance penalty with respect to the hierarchical views z .

Similarly, in our exploration-via-clustering approach (Chapter 6), C corresponds to a pseudo-count-based intrinsic reward derived from the cluster of the GMM-induced latent space.

Another approach is to derive a learning component

$$C : \{1, \dots, K\} \rightarrow \{0, 1\}^K$$

to weight or mask the learning signal

$$\mathcal{L}_{\text{total}} = b \cdot \mathcal{L}_{\text{task}}. \quad (4.30)$$

In our work surrounding stance detection (Chapter 7), the embeddings of LLM-generated synthetic data acts as a reference distribution when projected through a pre-trained BERT model. The labels of these embeddings define a discrete latent structure that is used to mask out ambiguous samples that potentially hurt classification performance of a BERT-based classifier

$$\mathcal{L}_{\text{total}} = b \cdot \mathcal{L}_{\text{CE}}(y', y), \quad (4.31)$$

where y' is the true label and b is a weight that can be derived from the latent structure, essentially filtering out ambiguous samples based on the canonical structure of the synthetic data.

4.4.2 Continuous learning components

In many cases latent structure may encode multi-modal aspects of the data that are hard to discretise. In such cases, the latent structure may

be used as is and we define the learning component to be a latent vector sampled from the latent space. As opposed to discrete learning components, the role of the learning component function C becomes secondary for a continuous latent space, where its main purpose is to normalise or regularise the latent vector. That is

$$C : \mathbb{R}^d \rightarrow \mathbb{R}^k,$$

simply maps the latent vector into a different vector space. Thus, when the latent space is continuous, integrating the resulting learning component becomes more important.

This integration involves either augmentation or alignment of the model's existing representations. Given a continuous learning component $b = C(z) \in \mathbb{R}^k$, we define these strategies as follows:

1. **Representational augmentation:** The learning component b is concatenated with the model's existing representation to form an enhanced representation that captures both the model-based and data-based perspectives:

$$\hat{h} = [h, b], \quad \mathcal{L}_{\text{downstream}} = f_{\theta}(\hat{h}), \quad (4.32)$$

where h is the original representation, $\hat{h}$ is the augmented representation, and f_{θ} is a downstream task-specific function. This approach effectively expands the representation space to include the information encoded in the data-based inductive bias.

2. **Representational alignment:** The learning component serves as a reference or target representation to which the model's representation should align:

$$\mathcal{L}_{\text{align}} = d(h, b), \quad (4.33)$$

where d is a suitable distance or divergence measure, such as cosine distance, Euclidean distance, or KL-divergence depending on the nature of the representations.

In our multi-object instance retrieval work (Chapter 8), we employ the representation augmentation approach, where latent vectors learned by a VAE from object image patches are concatenated with global DINOv2 representations

$$v = [e, z] \in \mathbb{R}^{p \times (n_g + n_z)}, \quad (4.34)$$

where e represents the global features from DINOv2 and z contains the object-level features from the VAE. In this case, the learning component

function C is essentially the identity, only normalising z to a comparable norm as e . This augmented representation v is then used directly for similarity computation in the retrieval task.

Self-supervised learning methods like DINO, BYOL [Grill et al., 2020] or SimSiam exemplify the representation alignment approach, where the representations from a teacher model serve as a target for the student model

$$\mathcal{L} = d(b^{\text{student}}, b^{\text{teacher}}), \quad (4.35)$$

where d maximises the cosine similarity between the student's and teacher's representation. In DINO's case for instance, the learning component function $b = C(z)$ consists of a softmax operation that normalises and regularises the latent representations z into a vector of discrete bins.

Summary. In this section, we defined the learning component $\mathcal{B}$ and its several possible instantiations via the learning component function C . We then discussed how to integrate the learning component into the model, which can usually be done as an auxiliary term to the loss function if the learning component is a scalar. When the learning component is a vector, i.e., the representation itself, it can be used to augment the existing representation of the model in order to learn better representations or integrated into the loss function directly as a reference representation. In general, the data-based inductive bias can be seen as a reference or regularisation term towards which the model-based inductive biases should be aligned to.

In the next section, we will introduce the concept of self-supervised learning and how it can be seen as a special implementation of our model-based and data-based inductive biases framework.

4.5 Self-Supervised Learning Integrates Model- and Data-based Inductive Biases

Having defined the components of a data-based inductive bias in the previous sections, we now demonstrate how self-supervised learning exemplifies this framework in practice. Self-supervised learning methods are based on pretext tasks, which are supervised learning objectives derived directly from the data itself. This objective enables models to discover and exploit underlying structure without external labels.

The key insight is that pretext tasks fundamentally identify invariances or transformations in the data which then can be used to create learning signals. For instance, contrastive methods learn representations by distinguishing between different augmentations of the same

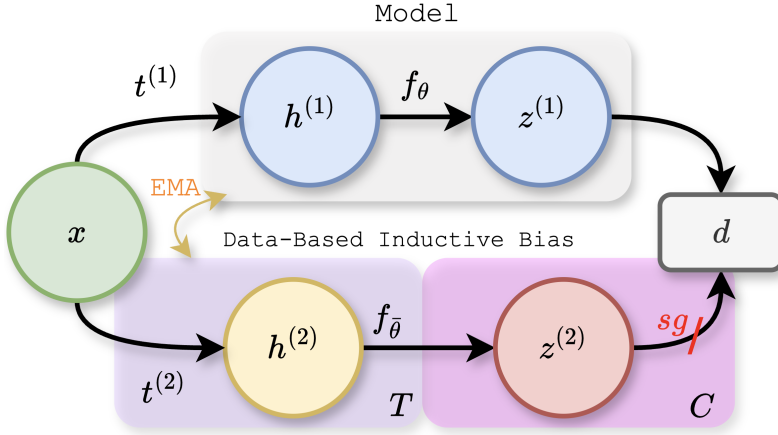


Figure 4.2: **Self-supervised learning under the model-based, data-based inductive bias view:** Self-supervised learning consists of model-based and data-based inductive bias paths. The data-based path’s parameters are fixed and only updated periodically serving as a reference to the model-based path. The structure inducing function T makes use of data augmentations and a neural network to project the invariances defined by the data augmentations to a latent space. The latent representation from the data-based path is then used to update the model-based path.

input, while self-distillation methods employ teacher-student architectures to learn consistent representations across different views of the same data. In each case, the method first induces structure in the data (through augmentations, view generation, or other transformations) and then leverages this structure to create a learning objective that biases the model toward more expressive representations.

This directly fits into our framework for data-based inductive biases: the structure inducing function T corresponds to the data transformations that define the pretext task, while the learning component function C and the corresponding learning component B transform the latent structure into a learning signal to guide representation learning.

As a concrete example and to prepare for the methods presented in this paper, we examine the data-based inductive bias underlying self-distillation methods, which represent one of the most successful paradigms in self-supervised learning and serve as the foundation for influential approaches such as BYOL and DINO.

4.5.1 The Inductive Biases of Self-Distillation Methods

In Figure 4.2, we present the general structure of self-distillation methods such as BYOL and DINO in terms of the components of our framework. Self-distillation methods rely on a siamese architecture consisting of of a *student* and *teacher* path. These two paths can be directly translated to our framework where we call the *student* path, the *model* path and the *teacher* path, the *data-based inductive bias* path. The *model* path has learnable parameters and encodes the model-based inductive biases, while the *data-based inductive bias* path has fixed parameters that get updated periodically (via an exponential moving average), thus serving as a fixed learning component for the *model* path. In the following, we will look at each part of the data-based inductive bias

path in detail and how it interacts with *model* path.

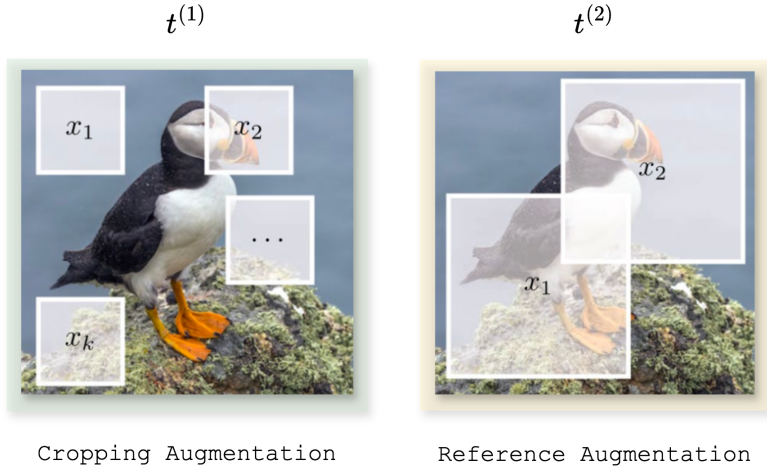


Figure 4.3: Data augmentations

Structure inducing function: data augmentations. In self-distillation methods, the structure inducing function T combines data augmentations with a neural network based backbone f . This is a similar setup as the one described in Section 4.4. The data augmentations serve as pre-processing transformations that reveal invariant structure in the data, while the function f projects these views into a latent representation space $\mathcal{Z}$ where the relationships can be systematically exploited.

Common data augmentations and their corresponding desired invariances are:

- **Horizontal Flipping:** The model should be invariant to horizontal flips, as the semantics of most natural images remain unchanged (e.g., a car is still a car if mirrored). In contrast, vertical flipping is often avoided because it can alter spatial cues (e.g., the sky appearing below).
- **Blurring:** Blurring, such as with a Gaussian filter, reduces high-frequency details but retains the overall structure and semantics of the image. The learned representation should be invariant to moderate blurring, preserving object identity even with slightly reduced detail.
- **Cropping and resizing:** Invariance to spatial transformations like random cropping ensures that the model can identify objects regardless of partial occlusion or slight changes in scale and location.
- **Color jittering:** The model should learn to ignore variations in brightness, contrast, saturation, and hue that do not affect object identity.

A widely used augmentation is random cropping, as illustrated in Figure 4.3. Typically, two different augmentations of the same type are applied to the same original image: one is input to the *model* path, and the other to the *data-based inductive bias* path.

As shown in Figure 4.2, we obtain two different augmented samples $h^{(1)} = t^{(1)}(x)$ and $h^{(2)} = t^{(2)}(x)$. These augmented samples are then processed through the respective paths: the *model* path produces representation $z^{(1)} = f_{\theta}(x^{(1)})$ with learnable parameters θ , while the *data-based inductive bias* path produces representation $z^{(2)} = f_{\bar{\theta}}(x^{(2)})$ with fixed parameters $\bar{\theta}$ that are updated via exponential moving average. This is the structure inducing function T .

The learning component function: compression via softmax. To stabilise representation learning, self-distillation methods apply regularisation to the learnt latent representations. In the following, we describe this procedure for DINO: both $z^{(1)}$ and $z^{(2)}$ are passed through a softmax operator yielding $p^{(1)} = C(z^{(1)})$ and $p^{(2)} = C(z^{(2)})$. This maps the continuous representations to pseudo-discrete bins, compressing the representations and helping the extraction of salient parts.

Integrating the learning component via similarity loss. Integrating the learning component into the model then simply consists of minimising the distance between the teacher representation $p^{(2)}$, i.e. the learning component, and the representations from the student model, i.e., the model path. This is representational alignment as described in Section 4.4.2. Thus given a specific function $d : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ that measures the similarity between two projections, a loss function

$$\mathcal{L} = \sum_{i=1}^N d(p_i^{(1)}, c_i^{(2)}) + d(c_i^{(2)}, p_i^{(1)}), \quad (4.36)$$

measures the relationship between the *model* path representation $p_i^{(1)}$ and the *data-based inductive bias* path representation $p_i^{(2)}$. The loss is typically symmetric to ensure bidirectional learning.

Self-distillation methods in self-supervised learning exemplify the practical application of our data-based inductive bias framework by inducing and leveraging structure in the data. The siamese architecture separates model-based and data-based inductive biases into distinct computational paths: the *model* path contains learnable parameters that encode architectural biases, while the *data-based inductive bias* path employs fixed or slowly-updated parameters to serve as a stable structure inducing function T .

The data-based inductive bias emerges through a two-stage process. First, data augmentations expose semantic invariances by creating multiple views of the same underlying images. Second, the reference path

transforms these augmented views into structured latent representations through a fixed neural network backbone, ensuring consistent structural encoding.

The learning component function C consists of applying a softmax operator to the latent representations to obtain discrete logits. This regularised learning component then acts as a reference representation for the model-based path.

4.6 Summary

This chapter establishes a comprehensive framework for understanding and implementing data-based inductive biases through systematic exploitation of latent structure in data. Unlike model-based inductive biases that are embedded in network architectures and training procedures, data-based inductive biases derive learning signals directly from the inherent structure present within the data itself.

4.6.1 Core Framework Components

The framework consists of three fundamental components that work in sequence:

1. **Structure Inducing Function (T):** Maps data space $\mathcal{X}$ to a structured latent space $\mathcal{Z}$, extracting meaningful patterns and relationships from the data.
2. **Learning Component Function (C):** Transforms latent structure into actionable learning signals appropriate for integration into model training.
3. **Integration Strategy:** Incorporates the learning component into the model's objective function or architecture.

4.6.2 Key Approaches to Structure Discovery

Latent Variable Models. We examined two primary approaches for systematic structure discovery:

- **Gaussian Mixture Models (GMMs):** Ideal for discrete latent structures with clear cluster organization, enabling exact inference through the EM algorithm.
- **Variational Autoencoders (VAEs):** Essential for high-dimensional continuous data, using variational approximation to handle intractable posterior distributions.

Both methods follow the fundamental latent variable modeling framework of defining generative models $p(x|z)p(z)$ and learning posterior distributions $p(z|x)$ by maximizing marginal likelihood.

Data Transformations. Beyond latent variable models, we showed how data transformations can effectively induce structure through:

- **Pre-trained representations:** Leveraging existing models to extract semantic features.
- **Data augmentations:** Exposing invariances and relationships across different views of the same content.

4.6.3 *Learning Component Integration*

The framework accommodates different types of learning components based on their mathematical structure:

- **Discrete components:** Intrinsic rewards in reinforcement learning that guide exploration through single-valued signals.
- **Continuous components:** Rich representations that can augment existing features or serve as alignment targets.

4.6.4 *Self-Supervised Learning as Exemplar*

We demonstrated how self-supervised learning perfectly instantiates this framework, particularly through self-distillation methods like DINO and BYOL. These approaches:

- Use data augmentations as the structure inducing function T .
- Employ siamese architectures to separate model and data-based inductive bias paths.
- Create learning objectives that encourage invariance across different views of the same data.

4.6.5 *Connection to Thesis Methods*

This theoretical foundation directly enables the diverse applications presented in the subsequent chapters of this thesis. Each method demonstrates a specific instantiation of the data-based inductive bias framework:

Reinforcement Learning Applications (Chapters 5, 7): The structure inducing function T operates on state representations to discover

environmental patterns, while the learning component function C produces scalar intrinsic rewards that guide exploration toward novel or structurally interesting regions.

Natural Language Processing (Chapter 6): Large language models serve as the structure inducing function T by generating synthetic data with known latent structure, while the learning component function C extracts masks from this latent structure to improve stance detection through selective data augmentation.

Computer Vision (Chapter 8): Pre-trained vision models and variational autoencoders combine as structure inducing functions T to capture both global and object-level features, to enhance the fine-grained instance retrieval of self-supervised vision models.

Cyclophobic Reinforcement

Learning: A Cycle-Avoiding Data-Based Inductive Bias for Exploration in Reinforcement Learning

Corresponding publication. This chapter is based on the work:

Cyclophobic Reinforcement Learning

Stefan Sylvius Wagner, Peter Arndt, Jan Robine, Stefan Harmeling
Transactions on Machine Learning Research (TMLR)

Personal contributions. Stefan Sylvius Wagner did literature review in the area of exploration by intrinsic motivation in reinforcement learning. Together with Peter Arndt, Jan Robine and Stefan Harmeling the idea of avoiding cycles as inductive bias was proposed. Stefan Sylvius Wagner and Stefan Harmeling developed and formalised the method, while Stefan Sylvius Wagner then implemented the method, created the experimental setup and conducted the experiments. Stefan Sylvius Wagner wrote the initial draft of the document, which was then edited with the help of Peter Arndt and Stefan Harmeling.

Abstract. In environments with sparse rewards, finding a good inductive bias for exploration is crucial to the agent’s success. However, there are two competing goals: novelty search and systematic exploration. While existing approaches such as curiosity-driven exploration find novelty, they sometimes do not systematically explore the whole state space, akin to depth-first-search vs breadth-first-search. In this paper, we propose a new intrinsic reward that is cyclophobic, i.e., it does not reward novelty, but punishes redundancy by avoiding cycles. Augmenting the cyclophobic intrinsic reward with a sequence of hierarchical representations based on the agent’s cropped observations we are able to achieve excellent results in the MiniGrid and MiniHack environments. Both are particularly hard, as they require complex

interactions with different objects in order to be solved. Detailed comparisons with previous approaches and thorough ablation studies show that our newly proposed cyclophobic reinforcement learning is more sample efficient than other state of the art methods in a variety of tasks.

5.1 Preamble

The first method presented in this work introduces a “handcrafted” data-based inductive bias. We present an illustration of this data-based inductive bias in Figure 5.1. A latent structure over the observations is induced by taking different croppings of an agent’s observations. The croppings induce different Partially Observed Markov Decision Processes (POMDPs) for each view, where the POMDPs describe a latent structure over the full Markov Decision Process (MDP). This latent structure is leveraged by computing an intrinsic reward that avoids cycles over these different POMDPs thus leveraging the latent structure induced by the observation croppings.

This is a first example of a data-based inductive bias where the cropping transformation can be understood as structure inducing function T eliciting different POMDPs $\mathcal{Z}$ over the environment, while the cyclophobic intrinsic reward maps this latent structure as learning component function C to a learning component $\mathcal{B}$ that can be integrated into the model via the cyclophobic Q-function. We present this method in the following.

5.2 Introduction

Exploration is one of reinforcement learning’s most important problems. Learning success largely depends on whether an agent is able to explore its environment efficiently. Random exploration explores all possibilities but at great costs, since it revisits states very often. More efficient approaches use intrinsic rewards based on novelty [Ostrovski et al., 2017b, Bellemare et al., 2016b], where the occurrence of states is counted and some novelty metric is determined or be based on curiosity where novelty in the representations is determined [Pathak et al., 2017b, Burda et al., 2018b, Raileanu and Rocktäschel, 2020b, Zhang et al., 2021b]. This often leads to great results, but at the price of possibly not exploring all corners of the environment systematically. This can paradoxically be less sample efficient, when aleatoric uncertainty in the observations is not large enough to give an informative reward signal, such as environments which are challenging from a combinatorial standpoint, but where the observations are simple [Parisi et al., 2021]. Especially in environments where the task is complex we would ideally pursue both goals: novelty search and systematical exploration.

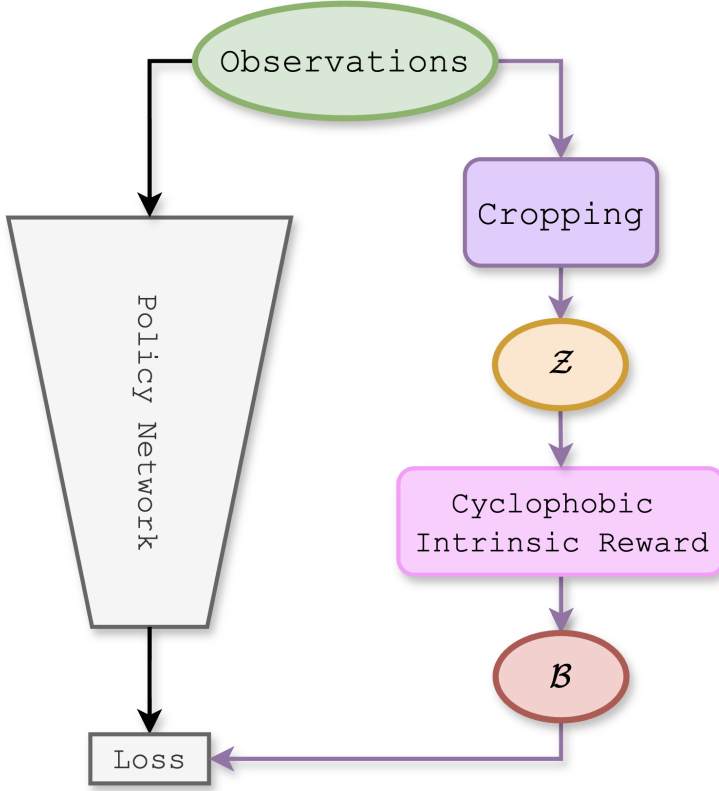


Figure 5.1: **Cropping and intrinsic rewards define the data-based inductive bias of Cyclophobic Reinforcement Learning**

How can we favor novelty while ensuring that the environment is systematically explored? To achieve this, we propose *cyclophobic reinforcement learning* which is based on the simple idea of avoiding cycles during exploration. More precisely, we define a negative intrinsic reward, i.e., the cyclophobic intrinsic reward that penalizes redundancy in the exploration history. With this we ensure two things: first, the agent receives immediate feedback about which state-action pair to discourage. Second, by maximizing the agent’s actions it is forced to take the next unexplored state-action pair since already explored state-action pairs will have a negative intrinsic reward. This approach is similar to optimistic initialization [Machado et al., 2014] where the value function is initialized with a high value and as the update rule is applied, the values will decrease for subsequent visits, thus forcing the agent to take the next unexplored action with highest remaining value. In this sense, the cyclophobic intrinsic reward can be seen as a count-based extension of optimistic initialization.

The cyclophobic intrinsic reward is further enhanced by applying it to several cropped observations of the environment which we call hierarchical state representations. Furthermore, we call the single cropped views, hierarchical views. The cropped observations allow

for masking out environmental noise by focusing on the events closer to the agent at different scales. This allows us to generalise across similar parts of the environment and to transfer knowledge to any other location of the environment. Similarly, cycles on a hierarchical level reveal structural properties about the environment. An interesting object such as a key, produces less cycles in a smaller view than some other object (since e.g. the key can be interacted with). Thus the probability of picking up the key increases, since other *less interesting* observations produce more cycles (e.g a wall).

Both the hierarchical views and cyclophobic intrinsic reward form a data-based inductive bias to the reinforcement learning agent. The hierarchical views effectively define a latent space where the structure inducing function is the cropping operation, generating different hierarchical views which each define a different POMDP of the environment. The learning component function C is then the cyclophobic intrinsic reward which leverages the structure from the latent space to generate an intrinsic reward that reflects the structure of each POMDP. The intrinsic reward is simply introduced into the policy model by mean of its objective function, i.e., update rule. We will describe each component in detail in the following.

5.3 Contributions

1. We introduce cyclophobic reinforcement learning for efficient exploration in environments with extremely sparse rewards and large action spaces. It is based on a cyclophobic intrinsic reward for systematic exploration applied to a hierarchy of views. Instead of rewarding novelty, we avoid redundancy by penalizing cycles, i.e., repeated state-action pairs in the exploration history. Our approach can be applied to any MDP for which a hierarchy of views can be defined.
2. In the sparse-reward setting of the MiniGrid and MiniHack environments we thoroughly evaluate cyclophobic reinforcement learning and can show that it achieves excellent results compared to existing methods, both for tabula-rasa and transfer learning.
3. In an ablation study we provide deeper insights into the interplay of the cyclophobic intrinsic reward and the hierarchical state representations, providing insight into the role they play in learning representations for exploration. Furthermore, we show that exploration and therefore learning success depend greatly on the function approximator learning appropriate invariances that reduce the complexity of the state space. Hence, in the former we analyse the effect of the data-based inductive biases,

while in the latter we explore the relevance of model-inductive biases.

5.4 *Related Work*

Learning representations of the environment that are reusable and can be transferred has been explored previously. Learning successor representations [Barreto et al., 2016] requires learning local environment dynamics that can be reused when the environment, i.e. the original distribution, is changed. [Schaul et al., 2015] use Singular Value Decomposition on the learned action-value function to obtain a canonical representation of the environment which can be transferred to other situations.

More recently, Parisi et al. [2021] learn an exploratory policy which is then combined with a task specific policy at transfer time. Our method likewise contains a task-agnostic component given by the hierarchical state representations and a task-specific component given by the cyclophobic intrinsic reward.

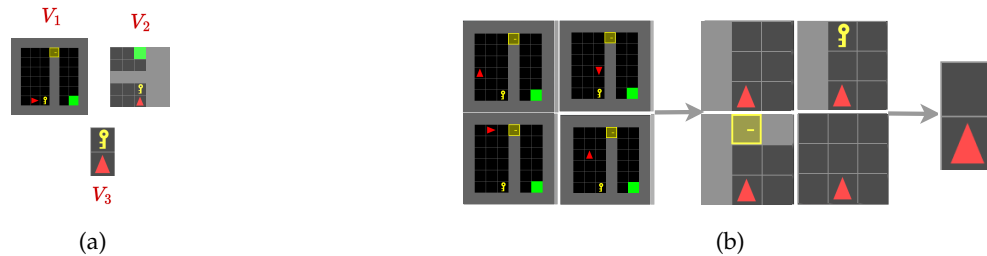
Mirowski et al. [2016] define loop closure as an auxiliary loss for exploration in 3D mazes in addition to a depth measure. The authors detect loop closure by checking whether the agent has been near a previously visited position. To get position information the authors integrate velocity information over time. Furthermore, NovelD [Zhang et al., 2021b] can also be seen as indirectly detecting loops as it part of its intrinsic reward is based on only rewarding states when they have not been visited before.

5.5 *Cyclophobic Reinforcement Learning*

We begin by first defining the structure inducing function T , which consist of the cropped hierarchical views and defines the latent space encoding structural invariances of the environment. The learning component function C then consists of defining a policy that leverages the cyclophobic intrinsic reward into providing a signal that encodes the structure of the several POMDPs induced by latent space defined by the cropped hierarchical views. Integrating the signal into the model is then simply achieved by adding this signal to the existing policy model reward.

5.5.1 *Hierarchical State Representations*

Besides penalizing cycles, the second key idea of this paper is to consider a hierarchy of state representations. For this, we repeatedly crop the agent’s observations to induce additional partially observ-



able Markov decision processes (POMDPs). In general, restricting the view leads to ignoring information about the environment, however in combination with the cyclophobic intrinsic reward we gain additional information about the structure of the environment as we get different Q-functions for different POMDP's. The relevant insight is that on limited views lower down the hierarchy, trajectories can contain cycles that have not been experienced on views higher up the hierarchy. These cycles in smaller views represent transferable knowledge about the structure of the environment. E.g. in the MiniGrid environment (see Section A.2) encountering a wall in the smallest view will cause a cycle, capturing that running into a wall is counterproductive. On larger views this knowledge is only available directly, if we tried all walls everywhere. So, the smaller views capture relevant invariances that also apply to the larger views. Hierarchical state representations also enable task-agnostic learning [Parisi et al., 2021]. That is, when learning an exploration policy we want to learn a policy that together with its representations can be transferred to other environments. By definition, the hierarchical state representations allow us to transfer learned invariances in smaller views to other environments by simply keeping these views for transfer learning. In general MDPs, it might not be obvious how to define hierarchical state representations. Ideally, the agent has some sort of sense of “location” to allow cropped views to be meaningful. In general, for grid world-like environments, the hierarchical state representations are easily definable, since the agent has a defined location that can be used to define smaller neighborhoods typically corresponding to limited views around the agent or from the agent's view.

To discuss the roles of the different views more precisely, let's consider three such cropped views which consist of the global view V_1 , the intermediate view V_2 and the smallest possible view V_3 (see Figure 5.2 (a)). Each view gives a new, typically more limited, perspective of the true state. As shown in Figure 5.2(b), each view induces a different set of cycles, for instance a sequence of actions which do not lead to a cycle in the full view V_1 , might lead to a cycle in a smaller view V_3 , because

Figure 5.2: **Hierarchical views allow us to transfer relevant invariances about the environment.** (a) The three different representations are obtained by cropping the observation for each state-action pair (s, a) . V_1 is the full view where the agent sees the whole environment or the largest portion of it. V_2 is an intermediate cropped representation of the agent's view that helps the agent generalize by reusing familiar observations. V_3 is the most restricted view where the agent only sees what is immediately in front of it. (b) Through the views V_1 to V_k the amount of cycles continuously increases as the observations in the higher views can be mapped multiple times to the same observation in the lower view. This naturally leads to each view being a separate POMDP describing the true MDP.

in the latter, more states are mapped to the same observation. Thus, the different views provide different types of information which are useful for learning and allow the agent to focus on different properties of the environment.

In general, we can have an arbitrary number of views. Each view V_i induces a POMDP

$$\mathcal{V}_i = (\mathcal{S}, \mathcal{A}, \mathcal{O}^{V_i}, \mathcal{P}, \mathcal{R}, \mathcal{Z}^{V_i}, \gamma) \quad (5.1)$$

each having their own set of observations $\mathcal{O}^{V_i}$ and observation function $\mathcal{Z}^{V_i} : \mathcal{S} \rightarrow \mathcal{O}^{V_i}$. All POMDP's $\mathcal{V}_1, \mathcal{V}_2, \dots$ operate on the same state space $\mathcal{S}$, however they have different sets of observations $\mathcal{O}$ and corresponding observation functions. General POMDPs can have a probabilistic observation function. In our case, the observations are deterministic functions of the full view, e.g. the observations for the i th POMDP of state s is

$$o^{V_i} = \mathcal{Z}^{V_i}(s) \quad (5.2)$$

which corresponds to the cropping for the view V_i . Hereby we create partial representations of the state space that allow us to identify invariances in the environment by looking at the same true state s through different perspectives.

Note that, POMDPs are normally used to model uncertain observations. That is, they are a generalization of MDPs where the true state is not observable. Here, the POMDP idea is used to model different views of a fully observable state space. Therefore, we do not seek to solve a POMDP problem, but rather we extend a regular MDP to get redundant hierarchical representations.

5.5.2 Cyclophobic Intrinsic Reward

For efficient exploration, redundancy must be avoided. A sign for redundancy is when states are repeatedly explored, with other words, when the agent encounters cycles in the state space instead of focusing on novel areas. To guide the exploration, we will penalize cycles using a cycle penalty, which we call cyclophobic intrinsic reward (a negative intrinsic reward). This avoids redundancy such that uninteresting parts of the state-action space are discarded quickly. For instance, if an agent gets stuck in some area, typically, it is facing numerous cycles. In such a situation we would like the agent to assign penalties to the repeating state-action pairs in order to focus on more promising parts of the state space that do not cause immediate repetition.

Formally, let us assume that we have per episode a history of previous state-action pairs $\mathcal{H}_{\text{episodic}} = \{(s_1, a_1), (s_2, a_2), \dots, (s_t, a_t)\}$ and we are currently at the state-action pair (s_{t+1}, a_{t+1}) . We say that we have

encountered a cycle if the state in the current state-action pair appeared already in the history, i.e. $(s_{t+1}, a_{t+1}) \in \mathcal{H}_{\text{episodic}}$. For its first repeated occurrence, we will penalize the state-action pair (s_t, a_t) (just before the cycle) by negative one,

$$r_{\text{cycle}}(s, a; \mathcal{H}_{\text{episodic}}) = -1. \quad (5.3)$$

Note that the cycle penalty as a function depends on the episodic history $\mathcal{H}_{\text{episodic}}$. If a cycle is encountered multiple times, e.g. l times, the overall cycle penalty for a state-action pair is $-l$. That is, during exploration the cycle penalty can penalize indefinitely. However, note that every single repeated encounter is penalized with -1 . For pairs (s, a) that have not created a cycle, $r_{\text{cycle}}(s, a; \mathcal{H}_{\text{episodic}}) = 0$.

Learning with cycle penalties. In principle, the cycle penalty can be combined with any reinforcement learning algorithm. However, we explain how it can be built into the SARSA update rule, since the latter will propagate the penalty across the trajectory. This is important as we want the cycle discouragement to happen on-policy, i.e., the state-action pair before the cycle should be penalized. Eventually, through the SARSA updates this information will propagate to other state-action pairs. To learn the Q-function, we are employing the standard SARSA update rule,

$$Q(s, a) \leftarrow (1 - \eta) Q(s, a) + \eta [r(s, a) + \gamma Q(s', a')] \quad (5.4)$$

with $(s, a, r_{\text{ex}}, s', a')$ being a transition, η being step size and where the total reward $r(s, a)$ for the state-action pair (s, a) is the weighted sum of the extrinsic reward from the environment r_{ex} and the cyclophobic intrinsic reward r_{cycle} defined above,

$$r(s, a) = \rho r_{\text{ex}}(s, a) + r_{\text{cycle}}(s, a; \mathcal{H}_{\text{episodic}}). \quad (5.5)$$

where ρ trades off extrinsic and intrinsic rewards.

5.5.3 A cyclophobic policy for hierarchical state representations

In the following, we describe how the cyclophobic intrinsic reward and the hierarchical state representations can be combined into a policy that exploits the cyclophobic inductive bias. For this, we define several Q-functions along the views, and combine them as a weighted sum. The weights are determined by counts of the observations in each observation set $\mathcal{O}^{V_i}$ which we explain next.

Mixing coefficients. Many strategies for defining mixing coefficients are possible. For concreteness, in this paper we follow a simple schema,

where we determine the weights from the observation counts, which are obtained from the history of states visited throughout training,

$$\mathcal{H}_{\text{all}} = \{s_1, s_2, \dots, s_T\}. \quad (5.6)$$

Note that $\mathcal{H}_{\text{all}}$ contains all states that have been visited in the training so far, which is different from the states in the episodic history $\mathcal{H}_{\text{episodic}}$ that was used in Sec. 5.5.2 to define cycles. Denoting the corresponding views of the history as $o_t^{V_i} = \mathcal{Z}^{V_i}(s_t)$, the counts for view V_i are

$$N(o_1^{V_i}), N(o_2^{V_i}), \dots, N(o_T^{V_i}). \quad (5.7)$$

where N counts the number of times the observation $o_t^{V_i}$ has been encountered. The raw counts are normalized by their maximum (maximum for simplicity, other normalizations are possible), because the smaller views have higher counts than the bigger views. The weights should be large for states that have been seen less often, so we subtract the normalized counts from one,

$$\alpha_*^{V_i}(o_t^{V_i}) = 1 - \frac{N(o_t^{V_i})}{\max(N(o_1^{V_i}), N(o_2^{V_i}), \dots, N(o_T^{V_i}))}. \quad (5.8)$$

The previous formula allows us to compute the weights for each respective view V_i for a single state s_t ,

$$\alpha_*(s_t) = [\alpha_*^{V_1}(\mathcal{Z}^{V_1}(s_t)), \dots, \alpha_*^{V_k}(\mathcal{Z}^{V_k}(s_t))] = [\alpha_*^{V_1}(o_t^{V_1}), \dots, \alpha_*^{V_k}(o_t^{V_k})] \quad (5.9)$$

where $\alpha_*(s_t)$ is a vector. Finally, the softmax operator turns these weights into a vector of probabilities,

$$\alpha(s_t) = \text{softmax}(\alpha_*(s_t)) = [\alpha_1(s_t), \dots, \alpha_k(s_t)]. \quad (5.10)$$

The entries of this vector are denoted by $\alpha_i(s_t)$ and are used to define the cyclophobic Q-function in the next section. The definition of α can be extended to all state s by setting it to zero for unseen states, i.e. $\alpha(s) = 0$ (zero-vector) for $s \notin \mathcal{H}_{\text{all}}$.

In general, larger views in the hierarchy will have bigger entries in $\alpha(s_t)$ as the observations repeat less often than in the smaller views. Thus $\alpha(s_t)$ gives more weight to the larger views than the smaller ones. However, the converse is true for the cyclophobic intrinsic reward, since in the smaller views the cyclophobic penalty is given far more often than in the larger views and thus the q-values in smaller views will be higher than in the larger views.

Cyclophobic Q-function. To combine all views into a single global policy we define a mixture over the different Q-functions learned with

the cyclophobic intrinsic reward as defined in Section 5.5.2. For view V_i , we define Q-function as

$$Q(o^{V_i}, a) \leftarrow (1 - \eta) Q(o^{V_i}, a) + \eta [r(o^{V_i}, a) + \gamma Q(o'^{V_i}, a')]. \quad (5.11)$$

This follows from our argumentation in Section 5.5.1, where we replace the state s in Equation 5.4 by the observations o^{V_i} in their respective views. Then we can define cyclophobic Q-function as the mixture of the Q-functions of each view,

$$Q_{\text{cycle}}(s, a) = \sum_i \alpha_i(s) Q(\mathcal{Z}^{V_i}(s), a) = \sum_i \alpha_i(s) Q(o^{V_i}, a). \quad (5.12)$$

Note that the mixing coefficients $\alpha_i(s_t)$ are only non-zero for states s_t that appeared in the global history $\mathcal{H}_{\text{all}}$. Thus the cyclophobic Q-function is zero for states $s \notin \mathcal{H}_{\text{all}}$ not encountered.

Cyclophobic policy. Finally, the cyclophobic policy is defined as the greedy action for the cyclophobic Q-function, i.e.

$$\pi(s) = \arg \max_a Q_{\text{cycle}}(s, a) = \arg \max_a \sum_i \alpha_i(s) Q(o^{V_i}, a). \quad (5.13)$$

Having normalized the counts within each view ensures comparability of the counts. This ensures that Q-values from rare i.e more salient observations have a larger effect on deciding the action for the policy π . In an ablation study in Section 7.6 we show that the combination of the cyclophobic intrinsic reward and hierarchical state representations is crucial to the method’s success.

5.6 Experiments

5.6.1 Experiment Setup

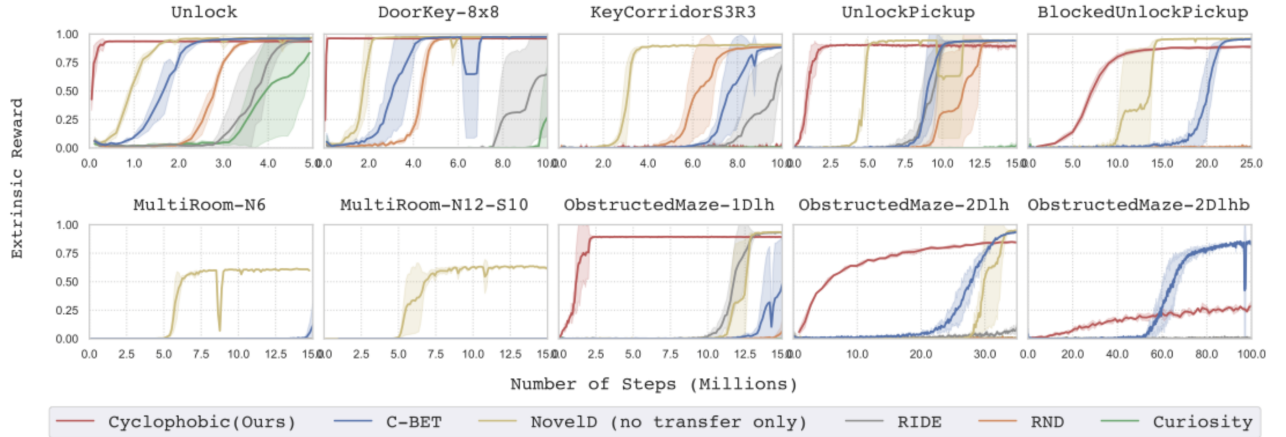
Our experiments are inspired by Parisi et al. [2021] and Samvelyan et al. [2021]. We test in environments where the causal structure is complex and the binding problem [Greff et al., 2020], [van Steenkiste et al., 2019a] arises. That is, where some form of disentangled representation of the environments plays an important role for efficiently finding solutions. This allows us to test the usefulness of the data-based inductive bias consisting of the hierarchical views and cyclophobic intrinsic reward.

Environments. We test our method on the MiniGrid and MiniHack environments:

- The MiniGrid environment [Chevalier-Boisvert et al., 2018] consists of a series of procedurally generated environments where

the agent has to interact with several objects to reach a specific goal. The MiniGrid environments pose currently a benchmark for the sparse reward problem, since a reward is only given when reaching the final goal state.

- The MiniHack environment [Samvelyan et al., 2021] is a graphical version of the NetHack environment [Küttler et al., 2020]. We select environments from the *Navigation* and *Skill* tasks. The MiniHack environment has a richer observation space by containing more symbols than the MiniGrid environments and a richer action space with up to 75 different actions. While not necessarily tailored to the sparse reward problem as the MiniGrid environment, the high state-action complexity makes it one of the most difficult environments for exploration.



Ablation Study. We also perform an ablation study where we ablate the components of the data-based inductive bias. For this, we analyse the effect of each component, i.e., the hierarchical views and cyclophobic intrinsic reward on the performance in the MiniGrid and MiniHack environments. Additionally, we also visualise the exploration exploration behaviour of each of the components.

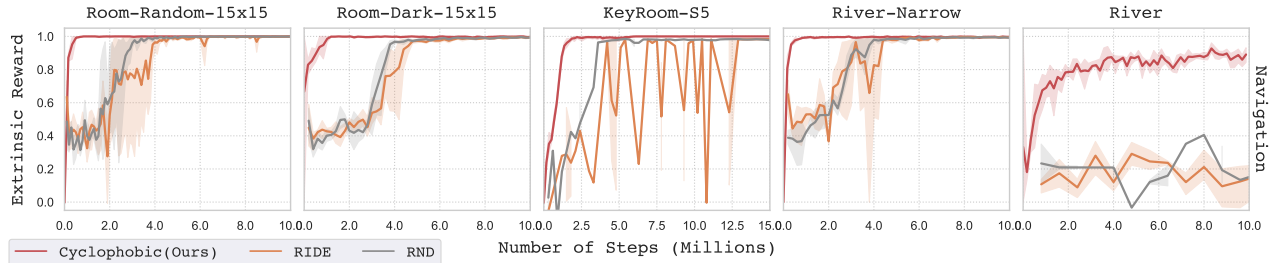
Apart from the data-based inductive bias, we also take a look at the influence of the model-based inductive biases in the policy model. The cyclophobic agent is tabular and thus show poor performance in environments, where properties such as color invariance are needed for efficiently solving the task. As an ablation, we thus modify the environment to contain objects with varying amounts of colour to study the influence of rising complexity with colour on the tabular

Figure 5.3: **Minigrid Exploration:** The hierarchical representations together with cyclophobic intrinsic reward allow for much quicker convergence than other methods. For the KeyCorridor and MultiRoom environments colour invariance is crucial to solving the environments which we analyse in our ablation study.

agent. Then we show that implementing a PPO based neural agent mitigates these issues through the model-based inductive bias defined by the CNN architecture.

Baselines. The baselines for the MiniGrid experiments are provided by Parisi et al. [2021] and allow us to compare our method to C-BET [Parisi et al., 2021], Random Network Distillation [Ostrovski et al., 2017b], RIDE [Raileanu and Rocktäschel, 2020b] and Curiosity-driven exploration [Pathak et al., 2017b]. Furthermore, we also compare to NovelD Zhang et al. [2021b]. For the MiniHack environment we compare our results to the baselines presented by Samvelyan et al. [2021] which include IMPALA [Espeholt et al., 2018b], RIDE [Raileanu and Rocktäschel, 2020b] and Random Network Distillation [Burda et al., 2018b]. For the skill tasks the only available baseline is IMPALA [Espeholt et al., 2018b].

Evaluation metric. While during learning the intrinsic reward based on cyclophobia plays the essential role, the ultimate goal is to maximize the extrinsic reward that is provided by the environment. Thus for comparison, we have to plot the extrinsic reward the agent receives for each episode. The reward ranges from 0 to 1.



5.6.2 Results

Minigrid. In three out of six environments, our proposed method converges much faster than the competitors, including C-BET [Parisi et al., 2021] and NovelD Zhang et al. [2021b]. Only “KeyCorridorS3R3”, “MultiRoom” and “ObstructedMaze-2Dlhb” pose significant challenges to our approach, because our method is tabular and thus cannot deal with too many object variations in the environment (e.g. random color changes). Furthermore, the “MultiRoom” environment proves challenging for all environments with only C-BET and NovelD managing to reach convergence, while we are able to fetch some rewards. This is due to the large amount of observations the different corridors produce.

Figure 5.4: **MiniHack Navigation:** The agent converges quicker in the *Navigation* task than the intrinsic curiosity base-lines such as RIDE [Raileanu and Rocktäschel, 2020b] and Random Network Distillation [Burda et al., 2018b]. This corroborates our hypothesis, that avoiding cycles is essential for quick exploration.

Note that we were not able to reach convergence in “ObstructedMaze-2Dlhb” with NovelD. In Figure 5.8 our approach also excels in the “KeyCorridorS3R3” and more difficult environments once we remove the colors, e.g. “KeyCorridorS4R3”, “KeyCorridorS5R3”, “KeyCorridorS6R3”.

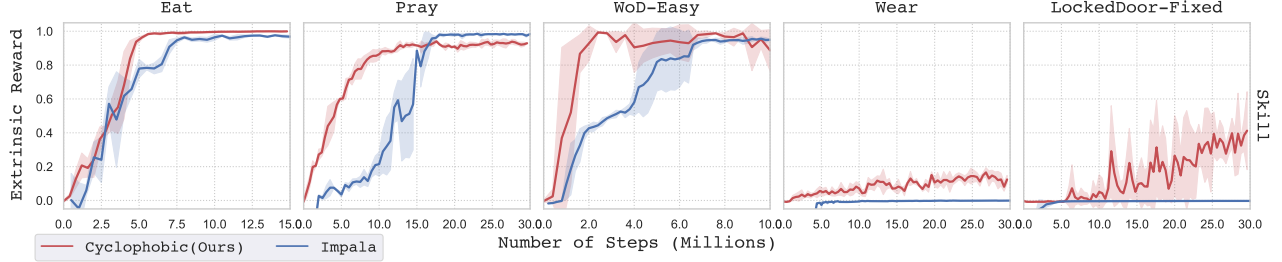


Figure 5.5: **MiniHack Skill:** We converge quicker than IMPALA [Espeholt et al., 2018b] in the *Skill* task. The *Skill* task defines over 75 different actions the agent must learn to use making it one of the most difficult sets of environments of the MiniHack suite.

MiniHack. To push our approach to its limits, we also tackle some of the MiniHack environments which requires the agent to learn a large skill set.

- **Navigation task:** For the simpler navigation tasks shown in Figure 5.4, our method converges quicker than intrinsic curiosity driven methods such as RIDE and RND. Especially, in the “River” environment, only our cyclophobic agent is able to converge and solve the environment. However, of course there are environments such as “RoomUltimate” where our approach fails due to its tabular style, which limits the complexity of the environment.
- **Skill task:** For the *Skill* task the only available baseline is IMPALA which is not based on intrinsic curiosity. Here we are also vastly superior, even collecting extrinsic rewards when the baseline can not (“Wear” and “LockedDoor-Fixed”).

5.6.3 Studying the Data- and Model-Based Inductive Biases

We analyse several aspects of cyclophobic reinforcement learning in regards to the model- and data-based inductive biases defined in this work. We first take a look at the impact of the hierarchical state representations and the cyclophobic intrinsic reward on the performance of cyclophobic reinforcement learning (see Figure 5.6), effectively measuring the effect of each of the components of the data-based inductive bias. We also analyse how efficiently cyclophobic reinforcement learning explores the environment by visualizing state visitation counts (see Figure 5.7).

Furthermore, we take a look at model-based inductive biases in the policy model, by studying how observational complexity affects exploration by ablating a reward-independent observational feature in the observations. For this, we reduce the number of different colors for objects in the “KeyCorridor” environment (see Figure 5.8).

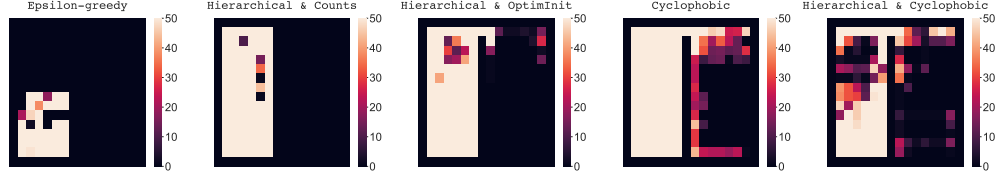


Impact of the data-based inductive bias. In Figure 5.6 we analyze the impact of the hierarchical state representations and the cyclophobic intrinsic reward on the agent’s performance. We ablate the hierarchical views in 3 ways.

1. In the first two figures we compare the cyclophobic intrinsic reward to optimistic initialisation (i.e., the learning components B) individually and additionally assess the effect of the hierarchical views (i.e., the structure inducing component T). We observe that the cyclophobic intrinsic reward is superior to optimistic initialisation. The hierarchical views improve performance for both the learning components, however only with cyclophobic intrinsic reward does performance improve substantially, showing that the hierarchical views and the cyclophobic intrinsic reward make up a good data-based inductive bias.
2. In the middle two figures we measure the effect of weighting the different views according to their visitation counts. This is part of the learning component function C leveraging the latent space of the hierarchical views into a learning component $\mathcal{B}$, i.e., the cyclophobic intrinsic reward. We see that for “River” the performance is improved while for “UnlockPickup” both weighted and unweighted variants perform the same. Overall the effect of weighting the different views is not very pronounced.
3. Finally in the last two figures we analyse the effect of using multiple croppings for the hierarchical views. We toggle the largest,

Figure 5.6: **Studying the data-based inductive bias in cyclophobic reinforcement learning:** Hierarchical state representations are crucial to performance in cyclophobic reinforcement learning. We observe that the hierarchical views improve performance for both the optimistic initialization and cyclophobic agent, while leveraging the largest performance boost with the cyclophobic intrinsic reward. Toggling between the largest, intermediate and smallest views. We see that all 3 types of views deliver the best performance.

intermediate and smallest views. Only using single views performs worst. Combining the largest and smallest views improves performance, especially for “River”. However, the best performance is achieved when combining the largest, intermediate and smallest views.



Exploration behaviour of the data-based inductive bias. Figure 5.7 shows state counts as a heat map after 10,000 steps of training. We distinguish four cases:

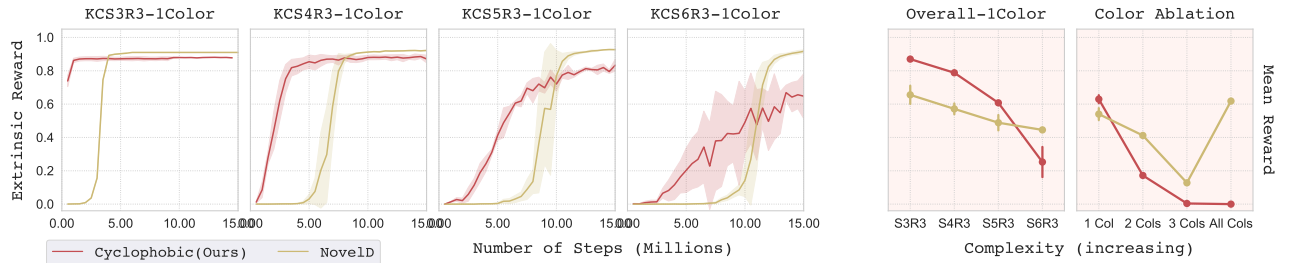
1. **Epsilon-greedy:** Plain epsilon greedy exploration fails to find the goal in the bottom right.
2. **Hierarchical & counts:** We replace the cyclophobic intrinsic reward in equation (3) with a count-based intrinsic reward similar to Strehl and Littman [2008b] defined by $N(o_T^{V_i})^{-\frac{1}{2}}$, for view V_i . This improves the results but still fails.
3. **Hierarchical & optimistic initialization:** We use optimistic initialization to let the agent try all possible actions. We initialize the q-table with a value of two. Optimistic initialization manages to enter the second room but just fails at getting a reward within 10,000 steps.
4. **Cyclophobic:** Having a cyclophobic intrinsic reward calculated only on the largest view finds the goal.
5. **Cyclophobic & hierarchical:** The combination of hierarchical views and cyclophobic intrinsic rewards (as explained in Sec. 5.5.3) explores even more efficiently as can be seen in the left room where fewer steps are needed to leave it.

Studying the model-based inductive biases. We demonstrated the impact of the data-based inductive bias through the hierarchical state representations and cyclophobic intrinsic reward. We now analyse the impact of the model-based inductive biases in the the function approximator, which learns observational invariances through its CNN

Figure 5.7: **Exploration behaviour(visitation counts for “DoorKey-16x16”):** The cyclophobic Q-function explores the environment more efficiently than the optimistic initialization, count and epsilon-greedy based counterparts. We record visitation counts for several variations of intrinsic reward and hierarchical state representations. Hierarchical state representations together with the cyclophobic intrinsic reward are the most efficient.

architecture. For this, we first show the impact of colour on the performance of the tabular agent and then show how using a neural PPO agent which is a CNN policy network mitigates the performance issues in an environment that benefits from invariance to colour for sample efficiency.

The tabular agent struggles when colour invariance is needed. In Figure 5.8 (white), we fix the color of objects in the environment to only one color and compare our method to NovelD [Zhang et al., 2021b], since it is able to solve all MiniGrid environments. We see that in “KeyCorridorS4R3” and “KeyCorridorS5R3” our method is more sample efficient than NovelD. Only in the “KeyCorridorS6R3” the large state space reduces performance (see Figure 5.8 (pink left)). In general, we see that when the environment is observationally less complex (e.g by removing colors (see Figure 5.8 (pink right))), the cyclophobic agent is more sample efficient than NovelD. However, as soon as observational complexity increases NovelD takes the upper hand.



The neural agent profits from colour invariance. We train a PPO agent using only the cyclophobic intrinsic reward. Figure 1 shows that for the given environments the cyclophobic PPO agent is able to perform well in the “KeyCorridorS3R3” environment. The colour invariance learnt from the model-based inductive bias introduced by the CNN architecture of the model thus enables the agent to solve the “KeyCorridorS3R3” environment.

5.7 Discussion

5.7.1 Limitations

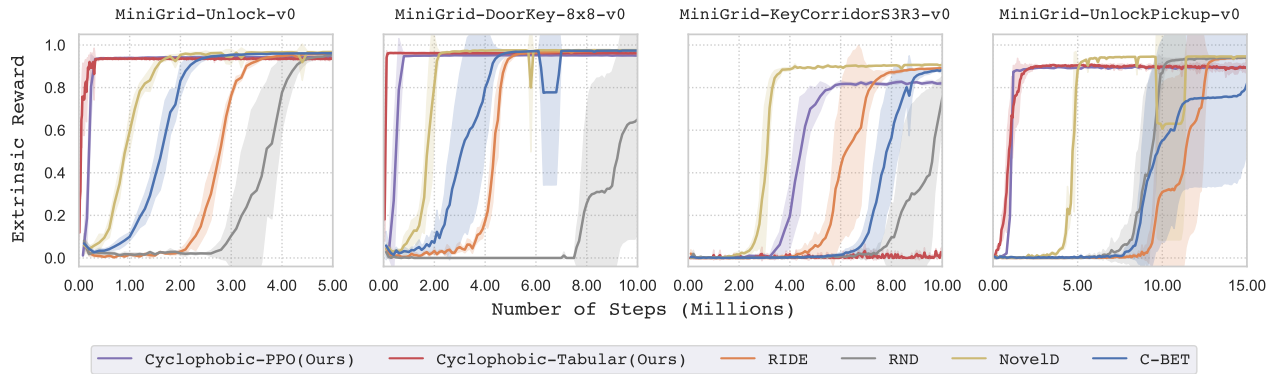
In some environments with a big number of different objects our method struggles to converge. The tabular agent does not have the observational invariances that methods based on function approximation

Figure 5.8: **Observational complexity ablation study (reducing colors):** The cyclophobic agent can gather rewards successfully in large “KeyCorridor” environments when reducing colors, showing that the exploration mechanism is very strong. Generally, the more distinct i.e. complex the observations in the environment are, the more the learned invariances from the neural network matter.

learn and exploit (e.g. with CNNs). For instance, for these environments learned invariances allow them to handle differently colored objects. While the tabular agent is able to avoid redundancy through the cyclophobic intrinsic reward and generalize this to other parts of the environment through the hierarchical state representations, it does not learn the kind of observational invariances neural networks do, which are sometimes necessary. However, note that we can show that reducing the number of colors for the “KeyCorridor” environment improves our performance dramatically (see Figure 5.8). Also, preliminary results show a cyclophobic PPO agent (see Appendix A.1) solving “KeyCorridor”, further supporting our theory. We explore this in our ablation study. An additional limitation that results from our tabular approach is that the entire observation history $\mathcal{H}_{\text{all}}$ has to be recorded. This however can be mitigated by using an unweighted sum of the q-values for all views as we show in the ablation study.

5.7.2 Conclusion

This first method introduces cyclophobic reinforcement learning as a data-based inductive bias to improve exploration in reinforcement learning. The structure inducing function T crops the global view of the agent into multiple local views which induce a latent space consisting of separate POMDPs for each view. This latent space is then leveraged through the learning component function C which takes the hierarchical views and computes the cyclophobic intrinsic reward for each view. This hierarchical cyclophobic intrinsic reward is then integrated into the model through a mixture of Q-functions over the different views by adding the intrinsic reward to the extrinsic reward in each view. This allows us to learn a policy that outperforms state of the art methods in several environments.



We also studied the importance of the data-based inductive bias, by

Figure 5.9: Using PPO together with only the cyclophobic intrinsic reward also converges far quicker than C-BET. Moreover, we are able to converge in KeyCorridorS3R3 confirming our theory in Section 5.7.1.

ablating the performance of the hierarchical views and the cyclophobic intrinsic reward separately. That is, we tested the influence of the latent space $\mathcal{Z}$ on the learning component $\mathcal{B}$. We show that only with both parts of the data-based inductive bias we achieve superior performance. Incidentally, we noticed that for some environments colour invariance is crucial to solving some environments. We show that in this case the model-based inductive bias of neural policy network is able to leverage this property compared to a tabular agent.

6

Leveraging an LLM as a Data-Based Inductive Bias to Find Ambiguous Samples in Latent Space To Improve Stance Detection

Corresponding publication. This chapter is based on the work:

**The Power of LLM-Generated Synthetic Data
For Online Political Discussions**

Stefan Sylvius Wagner, Maike Behrendt, Marc Ziegele, Stefan Harmeling
International Conference On Learning Representations (ICLR) 2025
Spotlight Award

Personal contributions. Stefan Sylvius Wagner, Maike Behrendt and Marc Ziegele did literature review in the area of stance detection in online political discussions. Maike Behrendt developed the original BERT fine-tuning pipeline, which Stefan Sylvius Wagner then used to develop the proposed method. Stefan Sylvius Wagner implemented the method and conducted the experiments, while the experimental setup was developed under the advise of Stefan Harmeling. Stefan Sylvius Wagner wrote the initial draft of the document, which was then edited together with Maike Behrendt and Stefan Harmeling.

Abstract. Stance detection holds great potential to improve online political discussions through its deployment in discussion platforms for purposes such as content moderation, topic summarization or to facilitate more balanced discussions. Typically, transformer-based models are employed directly for stance detection, requiring vast amounts of data. However, the wide variety of debate topics in online political discussions makes data collection particularly challenging. LLMs have revived stance detection, but their online deployment in online political discussions faces challenges like inconsistent outputs, biases, and vulnerability to adversarial attacks. We show how LLM-generated synthetic data can improve stance detection for online political discussions by using reliable traditional stance detection models for online deployment, while leveraging the text generation capabilities of LLMs for synthetic data generation in a secure offline environment. To achieve this, (i) we generate synthetic data for specific debate questions by prompting a Mistral-7B model and show that fine-tuning with the generated synthetic data can substantially improve the performance of stance detection, while remaining interpretable and aligned with real world data. (ii) Using the synthetic data as a reference, we can improve performance even further by identifying the most informative samples in an unlabelled dataset, i.e., those samples which the stance detection model is most uncertain about and can benefit from the most. By fine-tuning with both synthetic data and the most informative samples, we surpass the performance of the baseline model that is fine-tuned on all true labels, while labelling considerably less data.

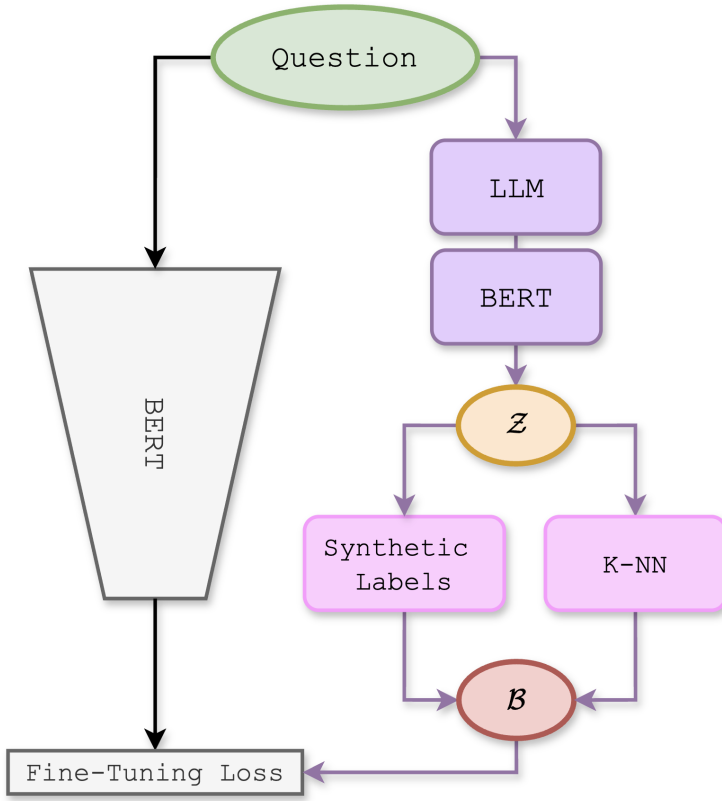


Figure 6.1: LLM-generated synthetic data together with pre-trained language representations make up a distinct latent space that can be leveraged for improved stance detection.

6.1 Preamble

The next method presented in this work defines a data-based inductive bias to improve a stance detection classifier. The data-based inductive bias allows for leveraging synthetic data generated by a large language model (LLM) to augment the stance detection classifier. The following method shows that this works best when the synthetic data induces a specific latent structure that is compatible with the stance detection classifier, i.e., extends its decision boundary sensibly. Incidentally, this induced latent structure can also be used to filter out low-quality (ambiguous) samples that hurt classifier performance, which can then be given to a human expert for proper labelling.

Figure 6.1 shows that both the structure inducing function T and the learning component function C can consist of multiple modules that work sequentially or in parallel. To induce the latent structure Z , the LLM first generates synthetic data from both classes which is then passed through a pre-trained BERT model. The BERT global embeddings then capture the latent structure present in the generated synthetic data. This latent structure can then be leveraged into masking labels that help filter out ambiguous samples based on the distance to

known synthetic neighbours. An added bonus of the induced latent structure is, that it can be used to either analyse whether the synthetic data aligns with the stance detection topic by comparing synthetic and real labels or the latent structure can serve as reference distribution to filter out ambiguous samples via k-nearest neighbours.

6.2 Introduction

With the recent advent of powerful generative Large Language Models (LLMs) such as ChatGPT, Llama [Touvron et al., 2023] and Mistral [Jiang et al., 2023], new ways of performing stance detection have opened up via zero-shot or chain-of-thought prompting. This is especially important in the area of online political discussion where topics are complex and labelled data is hard to come by. At the same time, an ever important use case in online political discussions is being able to use stance detection for an ongoing discussion to, e.g., suggest suitable comments for engagement between participants [Küçük and Can, 2020, Behrendt et al., 2024]. In the case of LLMs, while strong at analysing complex topics and at open-ended text generation, explicit classification can be inconsistent [Cruickshank and Xian Ng, 2023], they are prone to biases [Ziems et al., 2023] and open to adversarial attacks [Greshake et al., 2023]. More traditional stance detection models based on, e.g., BERT [Devlin et al., 2019] are task-specific and therefore consistent in their output, however they need large amounts of labelled data [Mehrafarin et al., 2022, Vamvas and Sennrich, 2020] to perform well.

In this work, we combine both traditional stance detection and LLMs to get the best of both worlds. For stance detection, we use BERT as a lightweight stance detection model that produces fast and consistent output given the data it has been fine-tuned on. To address the issue of needing large amounts of data and to improve stance detection performance, we define generate synthetic with an LLM that induces a specific latent space. We show the LLM is able to generate high quality samples which fall distinctly into the respective stance classes, thus giving us synthetic labels by construction. This allows us to improve the lightweight stance detection model in a number of different ways:

- (i) We can analyse the data for potential biases by comparing the alignment of the synthetic data distribution to the real world data distribution, allowing us to generate data data for augmentation that aligns the model to a specific topic, essentially making the model more robust.
- (ii) We can tackle the issue of ambiguous samples in stance detection that are difficult for the model to classify and therefore deteriorate its

performance. Since the synthetic data partitions the space between both classes, ambiguous samples that the stance detection model cannot classify properly can be identified as lying in between the two classes. These samples can then be labelled manually. Then, fine-tuning with these additional samples outperforms the model that is fine-tuned on all true labels alone, while only having labelled a subset of samples in the unlabelled data pool. We illustrate our method in Figure 8.2.

6.3 Contributions

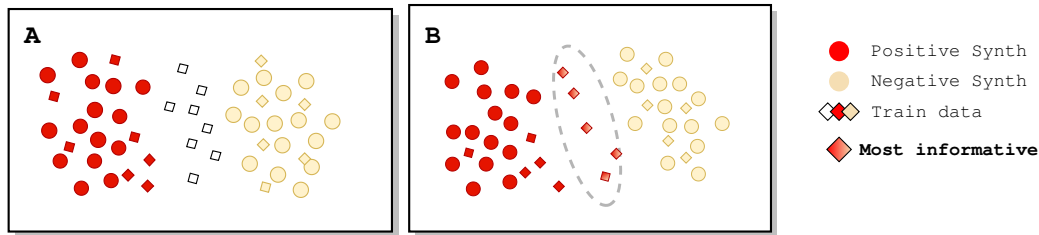
(Q1) Does fine-tuning a stance detection model with synthetic data improve stance detection performance? We first analyse whether fine-tuning the BERT model with synthetic data improves stance detection and show that this approach almost reaches the model trained with all true labels and is superior to using zero-shot Mistral-7B for the complex topics in online political discussions. *We reveal that a stance detection model can be tailored to a certain topic with only synthetic data.*

(Q2) How does the synthetic data improve performance and does it align with real world data? Our second question analyses the generated synthetic data. We analyse how well it aligns with the real world training data by visualising the T-SNE projected embeddings of the stance detection model and by comparing the entropy distribution of the synthetic data to the real data. *We find that the synthetic data aligns well with the real data, indicating that the LLM is able to generate comments for both stances while introducing minimal further bias.*

(Q3) Can we further improve the model by using the synthetic data as labelled reference distribution for active learning? The synthetic data allows us to identify unlabelled real data samples that improve the model even further through active learning. Due to the canonical nature of the synthetic data, we are able to extract real word samples for human labelling that are difficult (ambiguous) for the model to classify. We do this, by determining the k -nearest synthetic neighbours of the real data. *The stance detection model is fine-tuned jointly with these samples and the synthetic data, where we surpass the baseline model even when it is fine-tuned on all true labels, while labelling considerably less data manually.*

6.4 Related Work

Stance Detection for Online Political Discussions. Stance detection, a sub-task of sentiment analysis [Romberg and Escher, 2023] and opinion mining [ALDayel and Magdy, 2021], aims to automatically identify



an author’s stance (*favor*, *against*, or *neutral*) towards a discussed issue or target. In online political discussions, this involves determining if the contribution in question is *for* or *against* a topic like tax increases. Stance detection has been identified as an important task for improving discussion summarization [Chowanda et al., 2017], detecting misinformation [Hardalov et al., 2022], and evaluating opinion distributions in online political discussion and participation processes [Romberg and Escher, 2023]. Stance detection is also used in recommender systems and discussion platforms [Küçük and Can, 2020]. Still, due to its dependency on context, stance detection is a highly challenging task. Identifying stance requires understanding both the question and the contributor’s position, complicated by users often deviating from the original question and discussing multiple topics in the same thread [Ziegele et al., 2014], leading to little usable training data. Some works in stance detection use graph convolutional networks to learn more out of the present data [Zhang et al., 2022, Li and Goldwasser, 2019]. Recently, fine-tuning transformer-based models [Vaswani et al., 2017a, Liu et al., 2022] to solve stance detection is a common practice, but training these models requires a large amount of annotated data, which for the large variety of questions in online political discussions is unfeasible to acquire.

Active Learning. The aim of *active learning* is to minimize the effort of labelling data, while simultaneously maximizing the model’s performance. This is achieved by selecting a *query strategy* that chooses the most interesting samples from a set of unlabelled data points, which we refer to as *most informative* samples. These samples are then passed to, e.g., a human annotator for labelling. There exist many different query strategies such as Query By Committee (QBC, [Seung et al., 1992]), Minimum Expected Entropy (MEE, Holub et al. [2008] or Contrastive Active Learning (CAL, Margatina et al. [2021]). By actively choosing samples and asking for the correct labelling, the model is able to learn from few labelled data points, which is advantageous especially when annotated datasets are not available. Within the domain of political text analysis, many different tasks lack large amounts of annotated data. It has been already shown in the past that these tasks can benefit

Figure 6.2: We investigate the use of LLM-generated synthetic data for stance detection in online political discussions. (A) We generate synthetic data ●○ for specific questions using a Mistral-7B model. The synthetic data is then used to fine-tune the stance detection model. We show that fine-tuning with synthetic data improves the performance of the model, since the synthetic data is roughly faithful to the real data’s ◆◇ underlying distribution. However, some real world samples ◇ cannot be captured by the synthetic data. (B) We therefore use the synthetic data to identify the most informative samples ◆ in the unlabelled real data pool, which are better off labelled by human experts. Combining the synthetic data with the manually labelled most informative samples improves the performance of the model even further.

from the active learning: e.g., stance detection [Kucher et al., 2017], topic modeling [Romberg and Escher, 2022], speech act classification [Schmidt et al., 2023] or toxic comment classification [Miller et al., 2020]. In this work, we examine how LLM-generated synthetic data can be used instead of real labelled data to select the most informative samples to be manually labelled.

Stance detection and synthetic data generation with LLMs. Recent work has shown that synthetic data generated from LLMs can be used to improve the performance of a model on downstream tasks. Møller et al. [2023] showed that synthetic data can be used to improve the performance of a model on downstream classification tasks by comparing the performance of a model finetuned on LLM-generated data to crowd annotated data. In many cases the model finetuned on LLM-generated data outperforms the model finetuned on crowd annotated data. Mahmoudi et al. [2024] study the use of synthetic data for data augmentation in stance-detection. The authors use GPT-3 to generate synthetic data for a specific topic with mixed results due to the inability of GPT-3 to generate good data samples. In our work, we use a newer LLM model, Mistral-7B, which generates better synthetic data samples and show that we can generate synthetic data that matches the real data distribution. Veselovsky et al. [2023] analyse in which ways synthetic data is best generated for tasks like sarcasm detection and sentiment analysis. The authors reach the conclusion that grounding the prompts to generate the synthetic data to real samples helps improve the quality of the synthetic data. Similarly, Li et al. [2023] argue that subjectivity in the classification task determines whether synthetic data can be used effectively. It has been shown that LLMs can be used directly for stance detection such as [Cruickshank and Xian Ng, 2023], [Burnham, 2023] [Ziems et al., 2023]. However, the general conclusion of these studies is that while LLMs are competitive with other transformer models such as BERT, especially for edge cases, they exhibit replication issues. [Burnham, 2023] also discuss the possibility of pre-training models on more specific data to improve the generalisation capability of the model. Ziems et al. [2023] highlight the potential biases that can emerge in open ended generation tasks and classification performance varies depending on how representative the training data is. We therefore focus on using LLMs to generate synthetic data to solve key challenges in stance detection such as the lack of available data for specific topics and labelling large amounts of data, rather than using LLMs directly for the task.

6.5 Using Synthetic Data to Improve Stance Detection

In line with our declared contributions, we present the LLM-based data-based inductive bias to improve the performance of the stance detection model:

(i) To fine-tune the model with synthetic data, we first define the synthetic dataset and show our pipeline to generate it. The stance detection model is then further fine-tuned on the synthetic data to align the model to a specific topic.

(ii) We then present our synthetic data-based approach to identify the most informative samples from a batch of unlabelled data, where we propose a synthetic extension to the QBC (Query by Committee, [Seung et al., 1992]) method, where the synthetic data act as an ensemble of experts. As described in Section 8.2 and Figure 8.2, we use the synthetic data as reference distribution to identify the most informative samples in the unlabelled data pool. The idea is that for ambiguous samples the k -synthetic nearest neighbours are split in their labels and therefore lie on the decision boundary of the model.

Preliminaries. Political discussions are typically centered around questions $q \in \mathcal{Q}$ (sometimes also called issues or targets). For stance detection, we usually have for each of these questions q a set of labelled data $\mathcal{D}^{(q)} = \{(x^{(i)}, y^{(i)})\}_{i=1}^I$ where $x^{(i)} \in \mathcal{X}$ is a statement (or comment) and $y^{(i)}$ is the stance of the statement, with $y^{(i)} \in \{0, 1\} = \mathcal{Y}$. Note, that we use the notation $\mathcal{D}^{(q)}$ for labelled and for unlabelled datasets (then the labels are ignored). We view the stance detection model as a binary classification function $f : \mathcal{Q} \times \mathcal{X} \rightarrow \mathcal{Y}$, where we included the question as input to provide context. The stance detection model such as BERT [Devlin et al., 2019] is *fine-tuned* by minimizing the cross-entropy loss between the predicted labels $\hat{y}^{(i)} = f(q, x^{(i)})$ and the actual labels $y^{(i)}$.

6.5.1 Generating Synthetic Data for Stance Detection

To generate synthetic samples, we employ a quantized version of the Mistral-7B-instruct-v0.1 model to generate comments on a specific question q . These synthetic samples induce a specific latent structure when passed through a BERT model that allows us to augment the stance detection classifier to improve its stance detection performance. We use the following prompt to generate the synthetic data:

A user in a discussion forum is debating other users about the following question: [q] The person is in favor about the topic in question. What would the person write? Write from the person's first person perspective.

where "[q]" must be replaced with the question q . Similarly, to generate a negative sample, we replace "is in favor" with "is not in favor". As in the X-Stance dataset [Vamvas and Sennrich, 2020], we assign the two labels 0 and 1. We denote the question-specific synthetic dataset as:

$$\mathcal{D}_{\text{synth}}^{(q)} = \{(x_{\text{synth}}^{(m)}, 1)\}_{m=1}^{M/2} \cup \{(x_{\text{synth}}^{(m)}, 0)\}_{m=1+M/2}^M \quad (6.1)$$

where half of the M synthetic data samples have *positive* labels, i.e., are comments in *favor* for the posed question, while the other half is *against*. Since the dataset is in German, we translate the questions q with a "NLLB-300M" [NLLB Team et al., 2022] translation model. The English answers from the Mistral-7B model are then translated back to German using the translation model. We also tried other similar sized open source LLMs (Llama, Openassistant, Falcon), but found that only the Mistral-7B model produced sensible comments.

Coming back to Figure 6.1, the LLM together with the pre-trained BERT model act as structure inducing function T . By projecting the synthetic data $\mathcal{D}_{\text{synth}}^{(q)}$ to an embedding space with the pre-trained BERT model as in Equation 6.3, we get a latent space over the synthetic data that clearly separates the synthetic data into the two stance classes (see Figure 6.4). For data-augmentation (Q1), this latent structure gives us the learning component $\mathcal{B}_{\text{aug}}$ by construction, which are simply the labels of the generated syntethic data used for fine-tuning the stance detection model. Figure 6.4 (left), shows that this latent structure allows us to check alignment with the stance detection model's learned biases, which as shown in Section 6.7.2 is crucial for the synthetic data to be at effective at augmenting the stance detection model.

In the next section, we will define an additional learning component from the latent structure $\mathcal{Z}$ to detect the most informative samples in the unlabelled data pool.

6.5.2 Getting the Most Informative Samples: Synthetic Query By Committee

As discussed before, we can use the data-based inductive bias to generate an additional learning component that helps detecting ambiguous samples in real data samples, which can then be classified by a human expert or oracle. To identify the ambiguous (most informative) samples as described in (Q3) we take from two active learning methods: Query by Committee (QBC) [Seung et al., 1992] and Contrastive Active Learning (CAL, Margatina et al. [2021]). Instead of using QBC's ensemble of experts and the KL-divergence based information score in CAL, we directly use BERT embeddings of the synthetic data and its labels to identify ambiguous samples using k -nearest neighbors, thus exploiting the inherent latent structure of the synthetic data. The most informa-

tive samples are then the data points with the most indecisive scores. Synthetic Query by Comitte (SQBC) consists of three steps:

(1) Generate the embeddings. Given some embedding function $g : \mathcal{Q} \times \mathcal{X} \rightarrow \mathcal{E}$, with $\mathcal{E} \in \mathbb{R}^d$, we generate embeddings,

$$E = \{e^{(i)}\}_{i=1}^I = \{g(q, x^{(i)})\}_{i=1}^I, \quad (6.2)$$

$$E_{\text{synth}} = \{e_{\text{synth}}^{(m)}\}_{m=1}^M = \{g(q, x_{\text{synth}}^{(m)})\}_{m=1}^M \quad (6.3)$$

for the unlabelled data and the labelled synthetic data respectively. Note that q is the question for which we generate the synthetic data and for which we want to detect the most informative samples. If obvious from the context, we omit the superscript (q) .

(2) Using the synthetic nearest neighbours as oracles to score the unlabelled data. For the i -th unlabelled embedding $e^{(i)}$ let $\text{NN}(i)$ be the set of indices of the k nearest neighbours (wrt. to the embeddings using the cosine similarity) among the labelled embeddings E_{synth} . The score for each unlabelled data point counts the number of labels $y_{\text{synth}}^{(m)} = 1$ among the nearest neighbours, i.e.,

$$s(i) = \sum_{m \in \text{NN}(i)} y_{\text{synth}}^{(m)} \in \{0, \dots, k\}. \quad (6.4)$$

For our experiments, we choose $k = M/2$ which worked well across all experiments (other values for k are possible, but did not lead to significantly better results).

(3) Choosing the most informative samples. The scores take values between 0 and k . For 0, the synthetic nearest neighbours all have labels $y_{\text{synth}}^{(m)} = 0$, for value k , all have labels $y_{\text{synth}}^{(m)} = 1$. The *most informative* samples have a score around $k/2$. We thus adjust the range of the scores so that values in the middle range have the smallest scores (close to 0). We do this by subtracting $k/2$ from the score and taking the absolute value,

$$s'(i) = |s(i) - k/2|. \quad (6.5)$$

The J most informative samples $\mathcal{D}_{\text{MInf}}^{(q)} \subset \mathcal{D}^{(q)}$ among the unlabelled samples are the J samples with the smallest scores. In the experiments we vary J to study the impact of manually labelled most informative samples. Finally, the most informative samples are labelled by a human expert.

In summary, we use the syntethic embeddings E_{synth} , which follow the latent structure $\mathcal{Z}$ as reference distribution to filter out ambiguous samples (most informative samples). These can then be labelled by

a human expert yielding a learning component $\mathcal{B}_{\text{Minf}}$, which labels the most informative samples. Both learning components can now be combined to improve stance detection performance further.

6.6 Experiments

6.6.1 Datasets

X-Stance dataset. We evaluate on the German dataset of the X-Stance dataset [Vamvas and Sennrich, 2020], which contains 48,600 annotated comments on many policy-related questions (140 topics), answered by Swiss election candidates. We chose the German X-Stance dataset because it to our knowledge the most comprehensive stance detection dataset with a variety of topics and comments, while also being focused on online political discussions in governmental participation processes. Known english datasets such as SemEval-2016 [Mohammad et al., 2017] or P-Stance [Küçük and Can, 2020] offer less variety and are more focused on social media discussions. The comments are labelled either as being in *favor* (positive) or *against* (negative) the posed question. The dataset is split in training and testing questions, i.e., a question in the training dataset does not appear in the test dataset. Furthermore, for each question q from the training data, there are several annotated comments, which form the dataset $\mathcal{D}_{\text{train}}^{(q)}$. Analogously, for the test data we have a set of annotated comments written as $\mathcal{D}_{\text{test}}^{(q)}$. To refer to the whole training dataset we write $\mathcal{D}_{\text{train}} = \cup_{q \in \mathcal{Q}} \mathcal{D}_{\text{train}}^{(q)}$.

For our experiments, we fine-tune all stance detection models for each question separately allowing for better performance since the data distributions can vary greatly between questions (also a common scenario in online political discussions). To limit computation time, we selected 10 questions from the test dataset to evaluate our method, that best reflect the variability of the data. We split the datasets of these questions into Test-Train and Test-Test to fine-tune the BERT model on synthetic data and to perform active learning. That is, we use the Test-Train dataset to fine-tune the model and the Test-Test dataset to evaluate the model. The number of comments in the 10 selected test questions is shown in Table 7.

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
Total-Test	233 267	29 77	55 141	102 79	181 88	216 181	281 97	178 166	49 130	169 259
Test-Train	146 154	19 44	34 83	68 40	111 50	130 108	163 63	100 106	29 78	98 158
Test-Test	87 113	10 33	21 58	34 39	70 38	86 73	118 34	78 60	20 52	71 101

Synthetic dataset. For synthetic data-augmentation and active learning based on SQBC (see Section 6.5.2) we generate synthetic datasets

Table 6.1: **Number of comments in our 10 selected test questions for our experiments.** The numbers are split into (favor | against) labels. We have a 60 – 40 (Test-Train, Test-Test) split.

of varying sizes $M = \{200, 500, 1000\}$ for each of the 10 questions. The synthetic data follows the same structure as the data from the X-stance dataset, where for a specific question q we have M comments and M labels. Each set contains $M/2$ positive labels and $M/2$ negative labels, i.e., the synthetic data is balanced. We show samples of the synthetic data in Appendix C.1.

6.6.2 Experimental Setup

General setup. For all experiments, we start with a pre-trained BERT base model and adapt to the stance detection task by fine-tuning on the X-Stance training dataset $\mathcal{D}_{\text{train}}$ (all questions). We call this the **Baseline** since it is the vanilla BERT-based stance detection (e.g., Vamvas and Sennrich [2020]).

We evaluate our methods along the lines of the questions proposed in Section 8.2: **(Q1)**: we analyse the effect of fine-tuning **Baseline** with synthetic data and compare it to the **Baseline** that was only fine-tuned on $\mathcal{D}_{\text{train}}$. **(Q3)**: we fine-tune **Baseline** with synthetic data and the most informative samples. We present the baselines and our methods in the following:

Baseline methods.

- **Baseline**: the default BERT model fine-tuned only on $\mathcal{D}_{\text{train}}$, (e.g., Vamvas and Sennrich [2020]).
- **True Labels**: we fine-tune **Baseline** on the true labels of $\mathcal{D}_{\text{test}}^{(q)}$.
- **Random+Synth**, **CAL+Synth**: we use the active learning approaches to get the most informative samples $\mathcal{D}_{\text{MInf}}^{(q)}$.

Our methods.

- **Baseline+Synth**: we fine-tune the **Baseline** on the synthetic data $\mathcal{D}_{\text{synth}}^{(q)}$.
- **True Labels+Synth**: we fine-tune **True Labels** additionally on the synthetic data $\mathcal{D}_{\text{synth}}^{(q)}$.
- **SQBC+Synth**: we apply our active learning approach to get the most informative samples $\mathcal{D}_{\text{MInf}}^{(q)}$.

For fine-tuning with synthetic data only **(Q1)**, we compare the performance of our approaches to the baselines without synthetic data. For the active learning methods **(Q3)**, we compare to the non-active learning and active learning baselines. For further experimental details we refer to Appendix C.2.

Analysing the synthetic data. To analyse the synthetic data **(Q2)**, we visualize the BERT embeddings of the synthetic data together with

the embeddings of the real world data (see Figure 6.4). We use T-SNE [van der Maaten and Hinton, 2008] to project the embeddings, i.e., the output CLS token of the BERT model to a two-dimensional space. To assess whether the synthetic data captures the overall characteristics of the real world data and shares similar labels, we plot the individual embeddings of the synthetic data together with the means of the embeddings of the real world data. Additionally, we plot their corresponding labels. Finally, we also visualize the most informative samples selected by the different active learning methods.

6.6.3 Results: Evaluating the effectiveness of the synthetic data

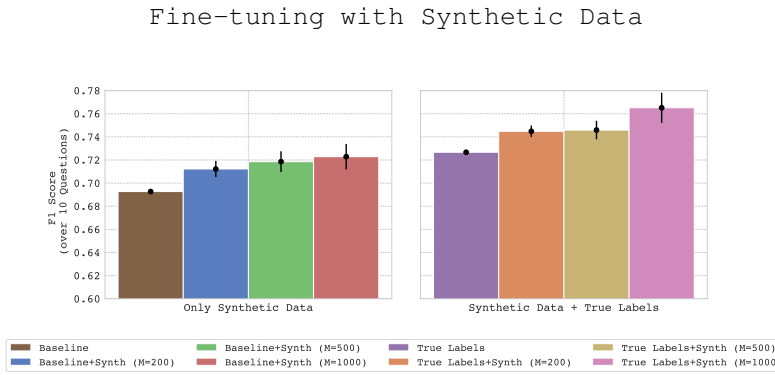
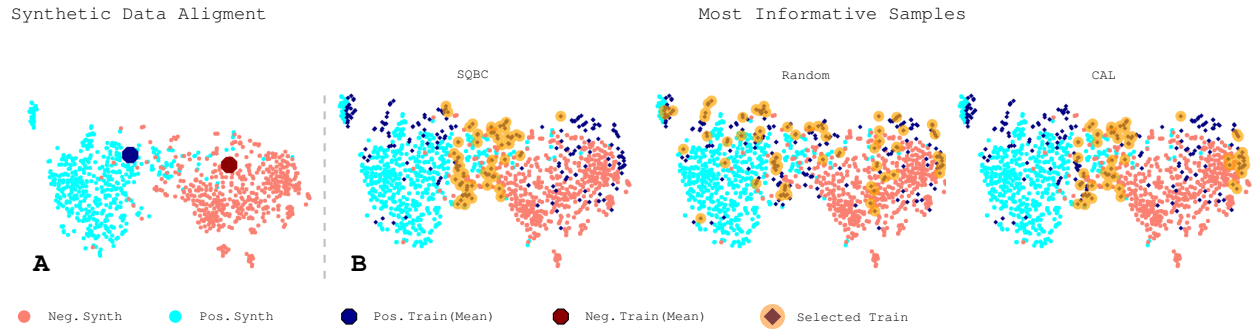


Figure 6.3: **Q1: Fine-tuning the model with synthetic data improves performance for increasing dataset size:** Shown are the F1-Score of fine-tuning with **Only Synthetic Data** (left) and **Synthetic Data + True Labels** (right) for increasing synthetic dataset size. Even if a dataset has been fully labelled, augmenting it with synthetic data proves equally as effective.

(Q1) LLM-generated synthetic data substantially improves stance detection. We show in Figure 6.3, that fine-tuning with only synthetic data improves the stance detection model. For $M = 1000$ the performance almost reaches the **True Labels** model, indicating that we can tailor a model to a certain topic without having any data for it. Furthermore, we can still improve the model consistently when new labelled data is available as seen by the **True Labels+Synth** models. In Section 6.7.1, we also analyse stance detection with zero-shot and fine-tuning approaches on Mistral-7B, but show that these are far less effective on the X-Stance dataset than our fine-tuned BERT model. In the following, we attempt to provide an understanding as to why the synthetic data is so effective.

(Q2) The synthetic data aligns well with real world data. We compare the T-SNE projected embeddings of the synthetic and real data in Figure 6.4(A). The synthetic data aligns well with the real world dataset, since the means of the training data are close and in the direction of the synthetic data. We further analyse the synthetic data in Section 6.7.2: the synthetic data is generally of high quality which validates the notion



that it can serve as a good reference distribution for the model. From a statistical learning point of view the synthetic data can be thought of as smoothening the decision boundary of the model. This also explains why fine-tuning with real samples is also very effective. Furthermore, we show that it is crucial to generate synthetic data that aligns with the given topic. The insight that the synthetic data provides a good prior, serves as motivation to use the synthetic data to identify ambiguous samples that are the most informative to the model. We elaborate on this in the following section.

(Q3) Synthetic data aids in finding unlabelled samples that further improve the stance detection model by extending its decision boundary.

We show the results of combining the most informative samples and synthetic data in Figure 6.5. Combining both, we outperform **True Labels** while using *only* 25% of the labelled data. We compare the selection strategy of the methods in Figure 6.4(B): Due to the k-nearest neighbours objective of **SQBC**, the model selects samples that are in between the two classes, which proves superior to **CAL** and to **Random** for smaller synthetic data sizes. **CAL** performs the worst across the board: it assumes that similar embeddings that have different outputs are ambiguous, which makes it prone to outliers in the real data, e.g., when the stance detection model misclassifies a sample. Therefore, **CAL** often selects samples from only one class which worsens performance. Interestingly for $M = 1000$, **Random** outperforms both active learning methods **SQBC** and **CAL**. **Random** selects similar samples to **SQBC**, but also uniformly samples from outliers from both classes, extending the decision boundary of the model. We argue this is especially effective for larger synthetic dataset sizes where the synthetic data smoothenes the decision boundary and thus mitigates the high variance introduced by the most informative samples. Thus, the model remains robust while extending the decision boundary. However, as we show in Section 6.7.2, the real data is quite homogenous. Therefore, with severe outliers present, **Random** could select these and worsen performance. This

Figure 6.4: **(Q2) Analysing the synthetic data ($M=1000$):** The synthetic data aligns well with the real data, which is crucial for improving stance detection performance and to check for potential biases introduced by the synthetic data. **SQBC** selects the samples that are in between the two classes, i.e., that are the most ambiguous and informative for the model.

would not happen with **SQBC** due to its k-nearest neighbour objective.

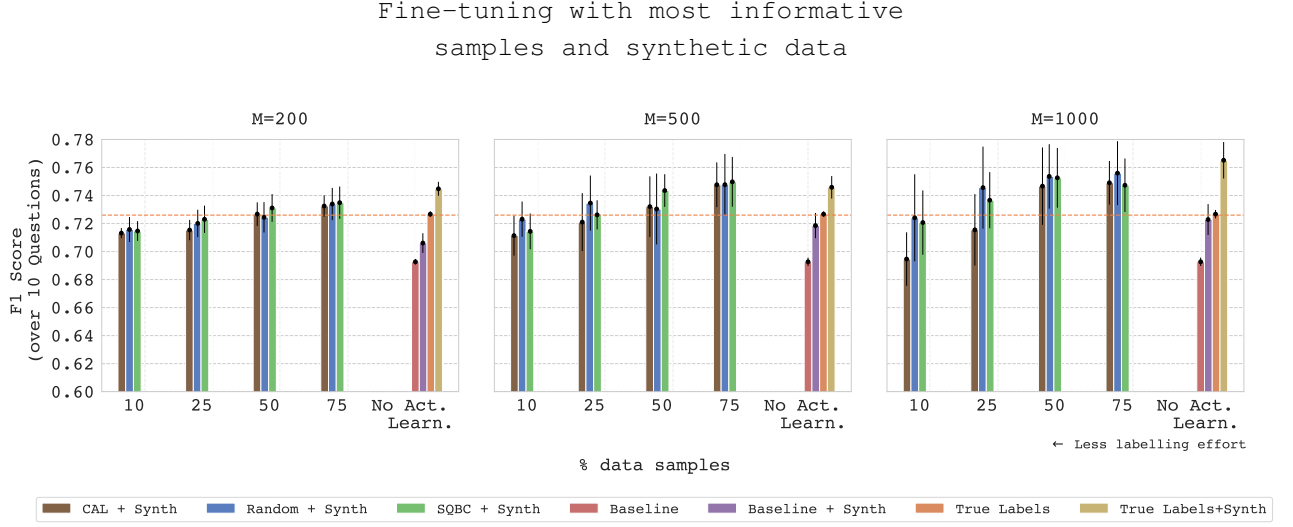


Figure 6.5: **(Q3) Fine-tuning with synthetic data improves stance detection, while combining most informative samples and synthetic data surpasses the baseline model fine-tuned with all true labels (above dashed line ---) using less manually labelled data:** The reason for the performance increase can be attributed to two phenomena: (i) the synthetic data smoothens the decision boundary of the model making it more robust to outliers. (ii) The most informative samples improve the model where the synthetic data distribution is not expressive enough.

6.7 Ablations

We investigate different aspects of using synthetic data for online political discussions. First, we study how well LLMs perform on classifying stance on the X-Stance dataset directly. Our assumption behind this is that while LLMs are strong at generating open ended text, they seem to have more difficulties when conditioned on a specific task combined with a narrow dataset. We also study the properties of the synthetic data by calculating the per comment entropy distribution together with the comment length. We then compare to the real data. Furthermore, to determine whether fine-tuning on a per topic basis is sensible, we study if the observed performance improvement is related to the content or the structure of the synthetic dataset.

6.7.1 Using LLMs directly for X-Stance

Cruickshank and Xian Ng [2023] and Gül et al. [2024] have shown promising results using zero-shot stance detection and fine-tuning various LLMs on common stance detection datasets such as SemEval-2016 [Mohammad et al., 2017] and P-Stance [Li et al., 2021]. However, both these datasets have relatively little variety in topics, with the topics addressing more popular discussion topics such as general US-Politics. For this reason, we evaluate Mistral-7B on D_{train} of X-Stance, which has a larger number of niche topics. We adopt the prompt and fine-tuning scenario (fine-tuning over 4 epochs with LoRA [Hu et al., 2021]) as in

Gül et al. [2024] and use our Mistral-7B model for both zero-shot stance detection and fine-tuned stance detection.

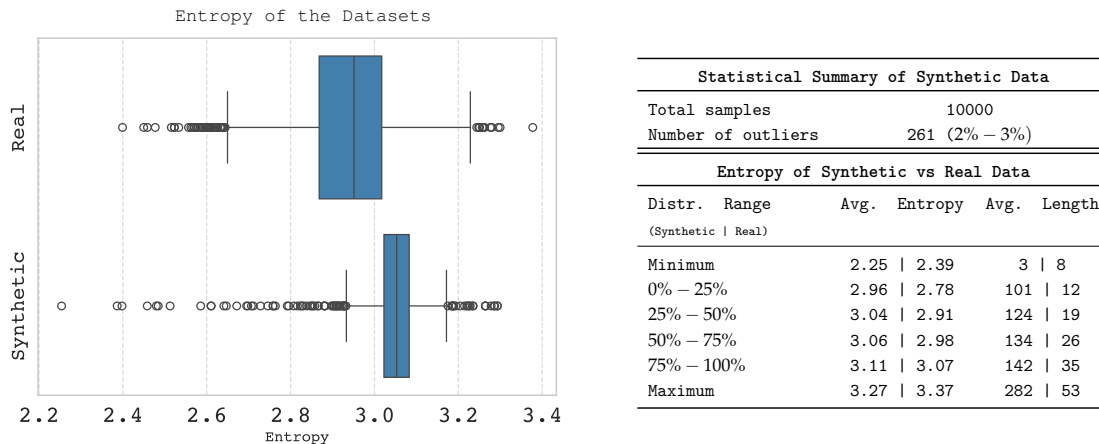
F1 Score (avg. 10 Questions)		F1 Score (avg. 10 Questions)			
		M	200	500	1000
Fine-tuned LLM	0.182	Baseline	0.693	0.693	0.693
Zero-shot LLM	0.419	Baseline+Synth	0.711	0.717	0.723
Baseline	0.693	Baseline+Synth			
Baseline+Synth (M=1000)	0.723	(Misaligned)	0.699	0.704	0.694
SQBC+Synth (M=1000)	0.754				

Table 6.2 shows that zero-shot stance detection barely reaches the performance of the pre-trained BERT baseline. Surprisingly, fine-tuning the Mistral-7B model with $\mathcal{D}_{\text{train}}$ worsened performance even further. We tried various hyperparameter settings and fine-tuned for up to 10 epochs, more than the 4 used in Gül et al. [2024]. We believe the poor performance can be attributed to a few reasons: (i) The topics in the X-Stance dataset are likely not present in the training sets of the Mistral 7-B model compared to SemEval and P-Stance which contain social media comments. (ii) Due to the smaller parameter count, the model may struggle to capture both the topic and the given comment for stance prediction. This often showed with the Mistral-7B model not giving consistent classification outputs or it would often refuse to predict stance. (iii) Furthermore, the model is not tailored to give single line response. In fact, the responses were often verbose so we also accepted answers that contained the words "favor" or "against". Better prompting strategies could improve performance, however with our findings we believe that as of now using LLMs for open-ended text generation is more effective than conditioning them to give a specific output for stance detection. We also tried other similar sized open source LLMs (Llama, Openassistant, Falcon) and found that they struggled similarly in producing consistent zero-shot classification.

Table 6.2: **Topic alignment is crucial for the synthetic data to be effective:** To determine whether improvement with synthetic data is due to the structure or content of the synthetic data, we augment the stance detection model with misaligned synthetic data. That is, the synthetic data does not align with the question given to the stance detection model. We observe the model only performs meaningfully better, when the synthetic data aligns with the posed question.

6.7.2 Studying the Synthetic Data

Properties of the synthetic data. To determine the quality and diversity of the synthetic data, we calculate the entropy of each comment (entropy over words) for both the synthetic and real data and compare the interquartile ranges of the corresponding entropy distributions in Figure 6.6. We also determine the average comment length for the interquartile ranges. The entropy reveals information about the content of both datasets, while the length acts a surrogate for structure. We observe, that the entropies between the real and synthetic data are similar, while the average comment length of the synthetic data is longer than the real data's. Looking at the samples in Appendix C.1, we observe that the real comments have a concise (even emotional) writing style, which is common in online political discussions. The synthetic



data comments are verbose and more reserved. Thus the difference in entropy could be attributed to the synthetic data containing less “emotional” comments compared to the real data. Nonetheless, we feel the synthetic samples manage to capture the content of the real data. Interestingly, the difference in length does not seem to affect the synthetic data alignment as can be seen from Figure 6.4. It is clear that the synthetic data is well aligned with the real data, even though the comment length between both sets of data differs. That is, the BERT model is largely invariant to the comment length and is more sensitive to the content of the comments rather than their structure.

Content vs structure: Is per topic fine-tuning necessary? To further validate our approach of fine-tuning the model per topic, we determine whether the improved performance comes from the discussion oriented structure of the data or from the content of the data. We fine-tune the model with misaligned questions and synthetic datasets, since we believe data across different topics does not necessarily share the same representation, justifying the need for per topic fine-tuning. Table 6.2(right) shows that fine-tuning with the synthetic data is only effective when the synthetic dataset is aligned with the posed question as can be seen by **Baseline+Synth**. Fine-tuning with synthetic data from a different topic **Baseline+Synth (Misaligned)** provides little improvement, delivering performance close to the **Baseline**. This validates the usefulness of the generated synthetic data for the stance detection model.

6.8 Discussion

Potential impact of this work. An apparent advantage of our approach is the possibility of dividing the synthetic data generation and the

Figure 6.6: **Synthetic data properties:** (Left) The per comment entropy of the synthetic data is similar to the entropy of the real data, where the synthetic data has a higher mean entropy than the real data, while the real data has higher variance. (Right) The generated synthetic data is of high quality, with only a few outliers. Interestingly, synthetic comments are longer than real ones, making real comments denser and synthetic ones more verbose. We argue that since the projected embeddings in Figure 6.4 show alignment between both datasets, BERT appears to be invariant to comment length.

stance model fine-tuning. The former can be outsourced to dedicated infrastructure, while with the latter fine-tuning and inference is accessible even for smaller organisations with fewer resources. Considering the large amount of topics in participation processes, the ability to generate synthetic data or to reduce labelling effort with SQBC further increases the benefits for smaller organisations that can't afford large scale data collection or labelling efforts.

Limitations. One limitation of our approach is that we fine-tune a separate model for each question. While this leads to good results, a common approach is to fine-tune a single (and thus more general) model for several questions (like pre-training **Baseline**). However, visualising the synthetic data in Figure 6.4, we observe that the underlying data distribution differs (sometimes greatly) for each question, which strongly suggests that each question benefits from fine-tuning a different model. This also aligns well with the per topic setting of online (political) discussions, considering that lightweight stance detection models can be fine-tuned in less than a minute even with a synthetic dataset size of $M = 1000$ on a reasonable GPU (NVIDIA A100). Another concern are biases that could be potentially introduced through the synthetic data. We addressed this in Section 7.6 by comparing the distributions of the synthetic data and real world data. We argue that analysing potential biases that could be introduced to the stance detection model through the synthetic data is easier in a single-question setting. In a multiple-question setting data from other topics could introduce biases into the model that are harder to detect. While effective for online discussions, due to the above observations, synthetic data quality should be assessed separately for different use cases.

Future work. Despite this work focusing on smaller interpretable models, we believe future work should investigate why the Mistral-7B model performs poorly on the X-Stance dataset. We shared our thoughts around this in Section 6.7.2, but this requires more extensive study. Another avenue for future work could be about leveraging the synthetic data distribution to generate more novel synthetic data. For instance, with synthetic data as reference we could learn a generative model which enforces a certain (distributional) distance to the synthetic reference distribution.

Conclusion. In this work, we presented how to improve stance detection models for online political discussions utilizing LLM-generated synthetic data. We used the LLM as structure inducing function T to generate synthetic data that induces a certain latent structure when passed through a BERT stance detection model. The synthetic data

essentially partitions the embedding space into the positive and negative samples allowing us to analyse and improve stance detection for online political discussions. In conclusion, we can summarise our contributions as follows:

- (i) we showed that fine-tuning with synthetic data related to the question improves the performance of the stance detection model.
- (ii) We attribute this to the LLM-generated synthetic data aligning well with the real data for the given question, while showing that the BERT model requires data that is more content aligned than structured.
- (iii) Fine-tuning with synthetic data can be further enhanced by adding the most informative samples which are identified by using the synthetic data as reference. This proves more effective than fine-tuning on all true labels, while using considerably less manually labelled samples, thus reducing labelling effort.

7

Improving Exploration by Learning a Latent Observational Space Through Representational Clustering

Corresponding publication. This chapter is based on the work:

Just Cluster It: An Approach for Exploration in High-Dimensions using Clustering and Pre-Trained Representations

Stefan Sylvius Wagner, Stefan Harmeling
International Conference On Machine Learning (ICML) 2024

Personal contributions. Stefan Sylvius Wagner, did literature review on clustering methods and intrinsic motivation. Stefan Sylvius Wagner developed and implemented the method and conducted the experiments. Stefan Sylvius Wagner wrote the initial draft of the document, which was then edited with Stefan Harmeling.

Abstract. In this paper we adopt a representation-centric perspective on exploration in reinforcement learning, viewing exploration fundamentally as a density estimation problem. We investigate the effectiveness of clustering representations for exploration in 3-D environments, based on the observation that the importance of pixel changes between transitions is less pronounced in 3-D environments compared to 2-D environments, where pixel changes between transitions are typically distinct and significant. We propose a method that performs episodic and global clustering on random representations and on pre-trained DINO representations to count states, i.e, estimate pseudo-counts. Surprisingly, even random features can be clustered effectively to count states in 3-D environments, however when these become visually more complex, pre-trained DINO representations are more effective thanks to the pre-trained inductive biases in the representations. Overall, this presents a pathway for integrating pre-trained biases into exploration. We evaluate our approach on the VizDoom and Habitat environments, demonstrating that our method surpasses other well-known exploration methods in these settings.

7.1 Preamble

In the previous two chapters we presented methods where the structure inducing function T was guided by expert knowledge. In the following, we present a method to improve exploration in reinforcement learning that systematically leverages pre-trained DINO representations by defining a latent space over the representations with a Gaussian mixture model (GMM). We use the GMM to induce a rich latent structure inherently present in 3-D high-dimensional observations. The GMM initially induces a latent space for the trajectories in the episode, these latent clusters are then assigned to a global table based on a cosine similarity threshold, to maintain state visitation counts over this latent structure. These state visitation counts can then be leveraged into a

learning component, i.e., an intrinsic reward for a PPO-based reinforcement learning agent. Compared to Cyclophobic Reinforcement learning (Chapter 5), where the data-based inductive bias is induced by data transformations, here the data-based inductive bias systematically learns a latent space with a latent variable model.

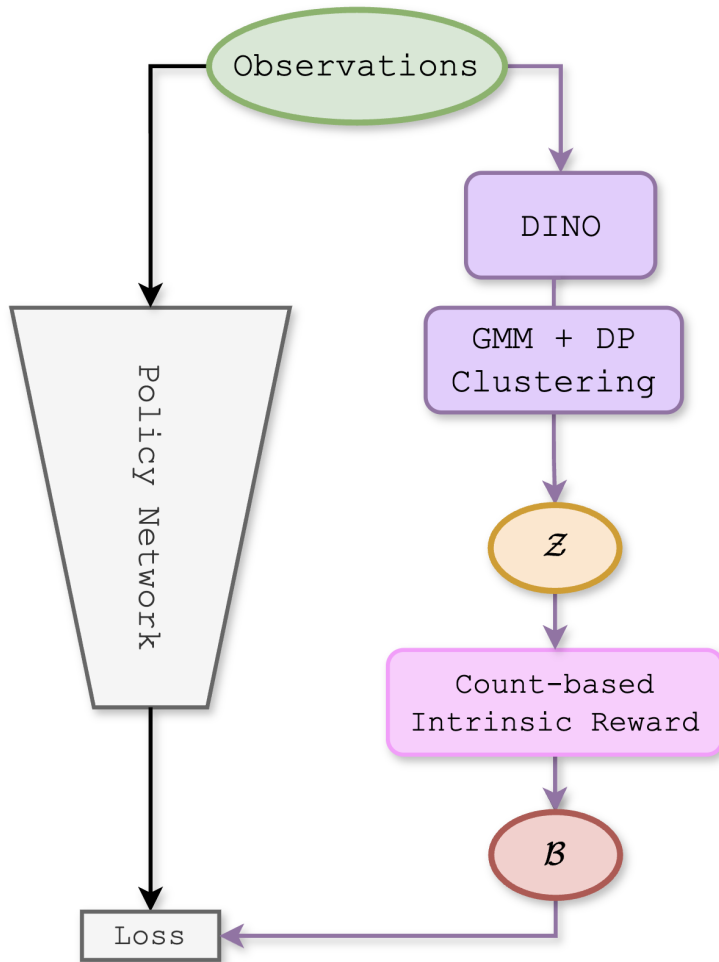


Figure 7.1: We induce a discrete latent space by clustering pre-trained DINO representations on high-dimensional 3D observations. This is then used to produce a count-based intrinsic reward that describes the latent structure and thus informs exploration.

7.2 Introduction

Exploration in reinforcement learning can inherently be seen as a density estimation problem. Approximating the density of the state-space throughout learning, allows the agent to determine which parts of the environment it has visited. This paradigm is at the core of exploration via intrinsic motivation where an intrinsic reward signal is created to quantify novelty. A simple way to do this is by counting states [Tang et al., 2016]. However, simply counting all states naively is not possible, since counts are very sparse in high-dimensional or

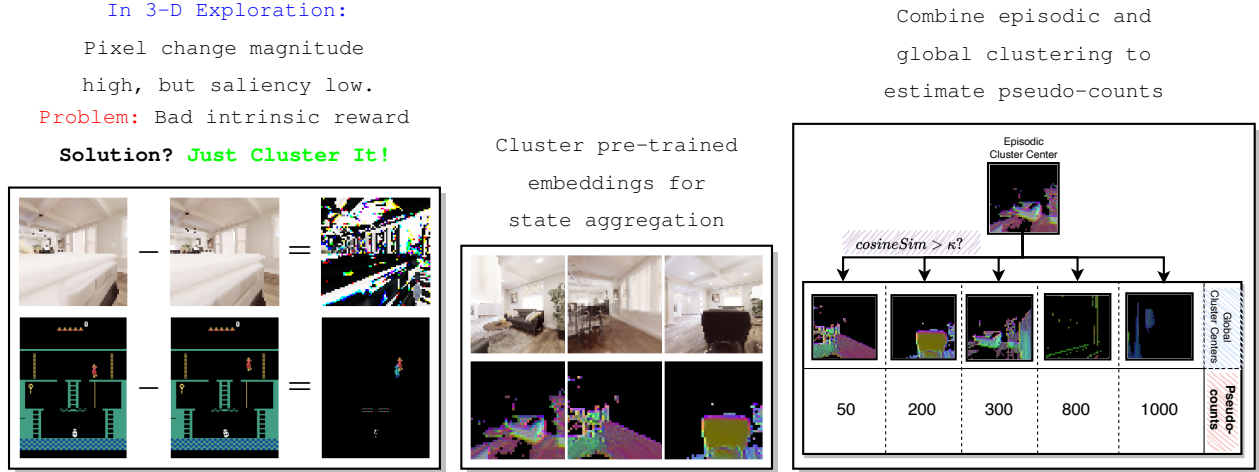


Figure 7.2: “Just Cluster It” in

a nutshell: (a) Density estimation for exploration in 3-D environments is challenging, because the magnitude of pixel change is large, but the saliency of a single transition is low. This is in contrast to 2-D environments where every transition is distinct and therefore salient. Therefore, we propose clustering representations on an episodic and global level to estimate accurate pseudo-counts that reflect the state-space distribution. (b) We show pre-trained representations from a DINO model. The embeddings from the DINO model are able to extract relevant features from the observations, which is useful for clustering. The bottom images are thresholded embeddings. (c) To estimate pseudo-counts, we store episodic cluster centers in a global cluster table over time by matching episodic cluster centers with previously added cluster centers in the cluster table. A new cluster center is added to the table, if the cosine similarity to existing cluster centers is below a threshold κ .

stochastic settings and the memory requirements to store every distinct count for a given state are very high. There are several approaches to estimating the state density: One way is to use some density estimation model that given an observation assigns a pseudo-count [Bellemare et al., 2016a], where a pseudo-count is an estimated count for a state given a density model. Other methods are based on using the prediction error, where the estimate for the state density is based on whether a model can predict fixed target features provided by a second model [Pathak et al., 2017a, Burda et al., 2018a, Raileanu and Rocktäschel, 2020a]. Established methods such as Random Network Distillation [Burda et al., 2018a] show that novelty can be determined simply by calculating the prediction error of random features. Different states will provide different feature vectors hence the model learns an indirect density estimate of the state space. Recent methods such as BYOL-Explore [Guo et al., 2022] replace random features with trained features, with great success.

The need for state aggregation in 3-D vs. 2-D. Due to its inherent simplicity, counting states for exploration in non-tabular environments has been studied extensively [Bellemare et al., 2016a, Tang et al., 2016, Ostrovski et al., 2017a]. While counting states for exploration works well in tabular settings, it becomes intractable in high dimensional settings if every state is counted explicitly. Bellemare et al. [2016a] were among the first to propose the estimation of pseudo-counts, where the counts are modeled by some density model of the state-space. Essentially, this transforms the problem of exploration into a density estimation problem of the state-space. A central property for a density model in exploration is that the model should be able to distinguish different states, such that salient transitions can be identified in the

intrinsic reward. In the case of counting states, the states should be aggregated in a sensible way such that the counts are not too granular, but also not too coarse [Ostrovski et al., 2017a, Ecoffet et al., 2021].

We identify a particular phenomenon regarding the importance of pixel changes in 3-D environments that requires states to be aggregated in order to provide a salient intrinsic reward. We provide a straightforward example in Figure 7.2(left), to demonstrate that state aggregation is necessary in 3-D environments compared to 2-D environments and thus must be accounted for by a density model in 3-D environments. We characterize this via transition saliency of pixels, i.e., the significance of pixel changes between two observations. For instance, in 2-D environments pixel change between transitions is usually small, but the significance of the pixel change is large. As a consequence, almost every observation contains new information regarding the current state of an agent. One example is Montezumas Revenge where individual groups of pixels change substantially when an action is performed, since the playable character figure changes its pixels when e.g., moving from left to right or when interacting with an object.

In 3-D environments by contrast, although the magnitude of pixel change is large, these pixel changes are usually not as significant, since after a single transition the relationship between the pixels in the observations remains largely the same. Rather, a more effective reference point for estimating the state density should be centered around salient features in the representations, such that observations with similar features are considered similar.

In conclusion, in 2-D environments the transition saliency of pixels is high, whereas in 3-D environments the transition saliency of pixels is low. Thus, the underlying density model must account for this lack of saliency in 3-D environments to yield a meaningful intrinsic reward. Similar thoughts have also been presented by Ecoffet et al. [2021], Ostrovski et al. [2017a]. We argue that tackling this transition saliency problem may bridge the gap from count-based methods to prediction-error-based methods which excel at providing a salient intrinsic reward signal when observations are discrete and pixel changes are meaningful. However, they perform worse when the observations are high-dimensional and individual transitions are not as meaningful.

Overall, if an appropriate density model, i.e., an appropriate latent structure can be found to estimate pseudo-counts in high dimensional observations, we can exploit the favorable theoretical guarantees of count-based methods [Bellemare et al., 2016a] and avoid catastrophic forgetting in neural networks. With this in mind, we deduce that a clustering-based approach, which is able to find the right level of state aggregation by finding shared features in the observations, may be much more effective in high-dimensional environments, provided the

underlying representations are expressive enough. Ideally, the level of state aggregation should be determined from the underlying features contained in the observations.

Exploration through Episodic and Global Clustering of High-Dimensional Representations. To this end, we propose a method for exploration that performs episodic and global clustering of (pre-trained) representations to estimate pseudo-counts. More precisely, in each episode we fit a Gaussian mixture model on random features or on features from a pre-trained DINO [Oquab et al., 2023a, Caron et al., 2021a] model to exploit the invariances contained in the representations. The Gaussian mixture model determines cluster means, i.e., centers for the observations in the current episode, which can then be compared with cluster centers previously stored in memory (see Figure 7.2 (right)). Particularly when utilizing pre-trained representations, as demonstrated in Figure 7.2 (middle), we see that these features align well with objects in the scene. We consider these features as various modes in the observational distribution, which makes them well suited for a Gaussian mixture model. The global clustering is then done in the cluster table, where a cosine similarity threshold κ is used to determine whether a new episodic cluster center should be added to memory or not. This is akin to the stick breaking process in Dirichlet processes and thus allows us to control the granularity of the inter-episodic clustering. We then calculate the intrinsic reward for each observation in the episode as the inverse square root of the pseudo-counts [Strehl and Littman, 2008a]. We conduct experiments on both the VizDoom [Kempka et al., 2016] and Habitat [Savva et al., 2019, Szot et al., 2021, Puig et al., 2023] environments, demonstrating that even features chosen at random outperform other exploration methods when dealing with 3-D environments.

7.3 Contributions

(i) We introduce an exploration method that determines novelty by estimating pseudo-counts of clustered pre-trained representations. We propose a method that involves both episodic and global clustering. The episodic clustering aggregates similar representations by fitting a Gaussian mixture model to pre-trained DINO representations or to random features for every episode, thus fitting the observations into a structured latent space. The global clustering is done over time by matching and storing episodic cluster centers in a global cluster table that aggregates pseudo-counts. Empirically, we demonstrate the effectiveness of this procedure in the chosen 3-D environments.

(ii) We show that both random and pre-trained DINO features can

be clustered effectively to aggregate states in 3-D environments and thus estimate pseudo-counts effectively. While random features work well on a simpler 3-D environment (VizDoom environment), with real-world observations (Habitat environment), the priors contained in the pre-trained DINO representations improve performance substantially.

(iii) By ablating two components of our method - the cosine similarity threshold κ and the episodic clusters, we show the effectiveness of clustering observations to induce a structure latent space that can be leveraged for exploration in high-dimensional 3-D environments.

7.4 Related Work

Count based methods and estimating pseudo-counts. Pseudo-counts as defined by Bellemare et al. [2016a] stand at the center of generalizing counts to high dimensional state spaces. They define state counts relative to density model that approximates the underlying state-space distribution. In principle, this definition of pseudo-counts is agnostic to the given state representation [Ostrovski et al., 2017a]. There are various exploration methods that rely on state counting, either to detect repeating observations or in order to normalize the intrinsic reward [Raileanu and Rocktäschel, 2020a, Zhang et al., 2021a, Wagner et al., 2023, Parisi et al., 2021, Tang et al., 2016]. These methods ultimately propose pseudo-counts when extending to high-dimensional state-spaces. Bellemare et al. [2016a] also demonstrate that estimating pseudo-counts using a density model provides a measure of either information gain, i.e., prediction gain. This shows that pseudo-counts are related to intrinsic reward methods that utilize prediction error as an intrinsic reward signal. When calculating an intrinsic reward bonus based on pseudo-counts, a common choice to calculate the intrinsic bonus is the inverse square root of the pseudo-counts as proposed by Strehl and Littman [2008a].

Novelty via prediction error. Most recent exploration methods fall under the category of exploration via prediction error. The idea is that the prediction error of a model with regard to some target features is a measure of novelty. Various versions of these methods exist, like Random Network Distillation [Burda et al., 2018a] (RND), in which the target features come from a neural network that is both fixed and initialized randomly. Pathak et al. [2017a] learns inverse dynamics with a separate model that learns to predict the correct actions between state transitions which makes the learned representations invariant to uncontrollable parts of the environment. Raileanu and Rocktäschel [2020a] predict the target features of the next state given the current state instead of predicting the target features of the same state. Recent

methods such as BYOL-Explore [Guo et al., 2022] learn representations online with BYOL [Grill et al., 2020] leading to better target features. Overall, these models estimate the density of the state space through the target features they approximate. Thus, the target features influence the state-space density that is learned. An advantage of prediction-error-methods is that since they are parametrized by a neural network they can generate an intrinsic reward signal ad-hoc from raw observations.

Density estimation via clustering representations. A basic method for estimating the density of the state-space involves the use of clustering. Ma et al. [2019] cluster the observations directly by performing k-means on the observations. Then they calculate the pseudo-counts on an episodic level and do not store the counts throughout the entirety of learning. “Never Give Up” [NGU, Puigdomènech Badia et al., 2020] also make use of episodic clustering to estimate episodic novelty. For each episode, NGU determines the sum of squared errors between the k-nearest neighbors of the current state and the current state itself. To capture long-term dependencies a global intrinsic reward is calculated via random network distillation. E3B [Henaff et al., 2022] performs episodic clustering by fitting a covariance matrix at each step and computing eigenvalues of the updated covariance matrix. The scale of the eigenvalues then determines the novelty. Apart from episodic clustering methods there are also methods that perform global clustering. Go-Explore [Ecoffet et al., 2021] maps observations to cells, which can also be seen as a form of clustering. However, instead of performing explicit clustering, a heuristic is used to find suitable values for downscaling parameters such as width, height and depth. The idea is that similar observations are mapped to the same cell. RECODE [Saade et al., 2023] attempts to estimate pseudo-counts by approximating clusters via Kernel Density Estimation (KDE). The current embedding is taken as central point and a density is fitted to find the nearest approximated cluster means stored in a table. If a new embedding is significantly different from its neighbors in the table it replaces an old embedding. RECODE is closely related to our method, with its Dirichlet process inspired approach. However, RECODE is also somewhat susceptible to the problem of transition saliency in 3-D environments, since it updates the cluster means at every step with only a single embedding as reference point. We also simplify the stick breaking process for the cluster table, as we only have a single threshold κ that determines whether a new cluster center should be added to the cluster table or not.

7.5 Just Cluster It: Exploration via Episodic and Global Clustering

In this section, we describe our exploration method “Just Cluster It!”, which consists of a two-stage clustering: first, we cluster after each episode, which we call *episodic clustering*, and second, we collect the newly found clusters in a global table of cluster centers, which we call *global clustering*. The cluster centers of the global clustering maintain a list of pseudo-counts to approximate the state space distribution. Therefore, it is important that the episodic cluster centers are representative of the observational distribution in the current episode, in order to aggregate the pseudo-counts correctly. Over time, the cluster table will contain pseudo-counts for the whole state-space distribution. Note that we call the counts *pseudo*, since they are defined relative to a mixture model.

For each episode, we employ a two-step process: first, we perform episodic clustering by fitting a Gaussian mixture model on the batch of representations of the current episode. This yields several episodic cluster centers. The second step updates the global cluster table that contains cluster centers from previous episodes and their corresponding counts. For each episode, counts from episodic cluster centers are then either aggregated to existing cluster centers, or a new cluster center is added depending on a cosine similarity threshold. This process of adding cluster centers continuously is similar to the stick breaking procedure in Dirichlet processes. However, instead of sampling weights from a beta distribution to determine the new cluster size, we simply add a new cluster center if the maximum cosine similarity is below a threshold $0 \leq \kappa \leq 1$. In this sense, κ controls the granularity of the clusters in the cluster table and is related to the concentration parameter for the Beta distribution in Dirichlet processes. The closer κ is to one, the more cluster centers are added to the cluster table as shown in Figure 7.5 (left). The key insight resulting from our method is that (pre-trained) representations in high dimensions are expressive enough to be clustered effectively on an episodic level, while also contributing to the approximation of the global state space distribution.

Notation and Preliminaries. We define an MDP as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$ with interactions at discrete time steps $t = 0, 1, 2, 3, \dots$. The agent receives a state from the environment as a random variable $S_t \in \mathcal{S}$, where $S_t = s$ is some representation of the state from the set of states $\mathcal{S}$ at timestep t . The agent then selects an action $A_t \in \mathcal{A}$ where $A_t = a$ is some action in the possible set of actions $\mathcal{A}$ for the agent at timestep t . This action is selected according to a policy $\pi(a|s)$ or $\pi(s)$ if the policy is deterministic. One time step later after taking the action, the agent

receives a numerical reward which is a random variable $R_{t+1} \in \mathbb{R}$, where $R_{t+1} = r$ is some numerical reward at timestep $t + 1$. Finally, the agent is in a new state $S_{t+1} = s'$. As usual γ is the discount factor. Furthermore, we define a POMDP $(\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{P}, \mathcal{R}, \mathcal{Q}, \gamma)$ for egocentric 3-D environments, where the true state s is not known. Thus, the agent sees the state $s \in \mathcal{S}$ through an observation $o \in \mathcal{O}$, where an observation function $\mathcal{Q} : \mathcal{S} \rightarrow \mathcal{O}$ maps the true state to the agent's observation. The components of our method are as follows: we define a policy network $\pi_\theta : \mathcal{O} \rightarrow \mathcal{A}$ with network parameters θ , which transforms an observation o into an action a .

We utilise a pre-trained model $f_\nu : \mathcal{O} \rightarrow \mathcal{E}$ with network parameters ν to project observations into a latent space $\mathcal{E}$, learned via self-supervised learning [e.g. DINO, Caron et al., 2021a]. A Gaussian mixture model $g : \mathcal{E} \rightarrow \mathcal{Z}$ is also defined, with $\mathcal{Z} = \{1, \dots, m\}$, which assigns a cluster label m to an embedding e , effectively defining a discrete latent space.

Our method has two hyperparameters: the cosine similarity threshold κ and the number of clusters M for the episodic Gaussian mixture model. This is less than other exploration methods based on clustering. In the following we describe our algorithm.

1. **Building episodic clusters.** We define an episode as a sequence of transitions $\{(o^{(t)}, a^{(t)}, r^{(t)}, o'^{(t)})\}_{t=1}^T$, where T is the length of the episode and $o'^{(t)}$ is the observation following $o^{(t)}$. Moreover, we define the set of observations as $B = \{o^{(t)}\}_{t=1}^T$ and let $\tilde{B} = \{e^{(t)}\}_{t=1}^T = f_\nu(B)$ be the embeddings after passing them through the pre-trained model f_ν . We then fit a Gaussian mixture model (GMM) on the embeddings $\tilde{B}$ to obtain M episodic cluster centers

$$\{\mu_{\text{ep}}^{(m)}\}_{m=1}^M. \quad (7.1)$$

The GMM induces the mapping $g : \mathcal{E} \rightarrow \{1, \dots, M\}$ which assigns each embedding $e^{(t)}$ to one of the clusters, which defines a sequence of cluster labels $m_*(t) = g(e^{(t)})$.

2. **Updating the global cluster table.** The following steps 2.1 to 2.4 are repeated for every episodic cluster $m = 1, \dots, M$. This builds the global cluster table incrementally.

2.1 Assigning an episodic cluster center to global cluster centers.

Before the first episode, the global cluster table is empty. In step 2.3 new pairs will be added. Assume for now that we have already K pairs $(\mu_{\text{global}}^{(k)}, \lambda_{\text{global}}^{(k)})$ of global cluster centers and global cluster pseudo counts

$$\{(\mu_{\text{global}}^{(k)}, \lambda_{\text{global}}^{(k)})\}_{k=1}^K. \quad (7.2)$$

Next, we calculate for fixed m the pairwise cosine similarities between $\mu_{\text{ep}}^{(m)}$ and $\mu_{\text{global}}^{(k)}$, which are collected in a K -dimensional vector D . The maximum of D gives us the maximal cosine similarity $d_*(m)$ and a corresponding global cluster label $k_*(m) \in \{1, \dots, K\}$ for each episodic cluster center.

- 2.2 Calculating the intrinsic rewards.** To calculate the intrinsic rewards along the episode, we assign the pseudo-counts for the embeddings $e^{(t)}$ that are assigned to cluster m ,

$$\rho^{(t)} = \lambda_{\text{global}}^{(k_*(m))} + \sum_{t'=1}^t [m_*(t') = m] \quad (7.3)$$

such that the pseudo-counts increase for each time an embedding belonging to the cluster m is seen in the episode.

For time steps t where $\rho^{(t)}$ has just been assigned, we calculate the intrinsic rewards as the inverse square root of the pseudo-counts [Strehl and Littman, 2008a],

$$r_{\text{int}}^{(t)} = \frac{1}{\sqrt{\rho^{(t)}}}. \quad (7.4)$$

- 2.3 Growing the global cluster table.** If the maximal cosine similarity $d_*(m)$ is smaller than the hyperparameter κ (a fixed value between 0 and 1), we add $\mu_{\text{ep}}^{(m)}$ to the list of global cluster centers and K is increased by one. The corresponding global pseudo count is initialized to zero, but will get updated in the next step.

- 2.4 Updating the global pseudo counts.** The global pseudo count of the global cluster center that corresponds to m , i.e., with index $k_*(m)$, is simply updated using the episodic cluster counts for the assigned episodic cluster center m :

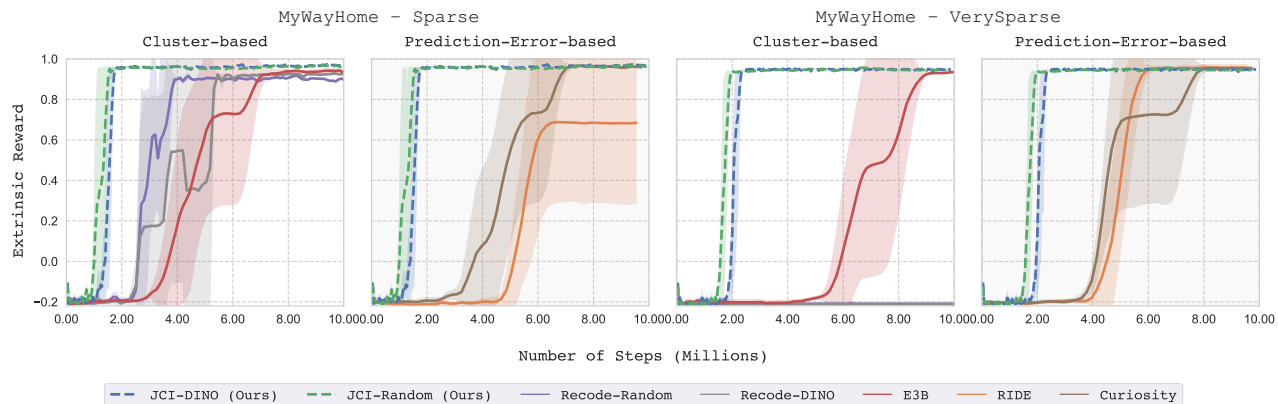
$$\lambda_{\text{global}}^{(k_*(m))} := \lambda_{\text{global}}^{(k_*(m))} + \lambda_{\text{ep}}^{(m)}. \quad (7.5)$$

- 3. Combining intrinsic and extrinsic rewards.** Finally, the intrinsic and extrinsic rewards are simply added, and we update the policy network π_θ with the transitions

$$\{(o^{(t)}, a^{(t)}, r_{\text{int}}^{(t)} + r^{(t)}, o'^{(t)})\}_{t=1}^T \quad (7.6)$$

via Proximal Policy Optimization (PPO).

Our approach effectively defines a data-based inductive biases that leverages the structure of the representations into a discrete latent spaces which allows us to perform count-based exploration in a more general way that mitigates the transition saliency problem.



7.6 Experiments

Given our proposed ideas and method we aim to answer the following questions: (i) How well does clustering features perform in high-dimensional 3-D environments? (ii) Can our method leverage pre-trained representations to improve exploration? (iii) What effect do episodic clustering and global clustering have on the performance and robustness of our method? We present the environments, experiment setup and baselines, followed by results of our experiments and corresponding ablations.

Environments. We test our method on observationally complex environments like VizDoom [Kempka et al., 2016] and Habitat-Replica [Savva et al., 2019, Szot et al., 2021, Puig et al., 2023]. In VizDoom every room has a different texture, where some moving textures are also present. In Habitat, the observations are real world scenes of apartments, hotels and office rooms which are very complex. The chosen environments are centered around navigation exploration tasks. This setup enables us to separate the issue of density estimation from the challenge of understanding complex environment dynamics. In general, all methods we evaluate including ours focus solely on the problem of density estimation, and we thus share the same assumptions. More details regarding the environments and agent setup can be found in the Appendix B.2.

Experimental setup and baselines. We compare our method to both clustering and prediction-error-based exploration approaches. Thus, we split the experiments into two categories for both environments. Both Figures 7.3 and 7.4 show comparison to only (i) clustering-based methods on the left side and to (ii) prediction-error-based methods on

Figure 7.3: **Clustering random features is effective for VizDoom:** Our method with clustering outperforms other traditional exploration methods. Interestingly, random features perform slightly better than the DINO features for VizDoom. While the complexity of the observations is not trivial, in this setting the priors in the pre-trained representations seem to be not more informative than random features. This also shows that in 3-D high-dimensional environments random features are salient enough when clustered to estimate pseudo-counts.

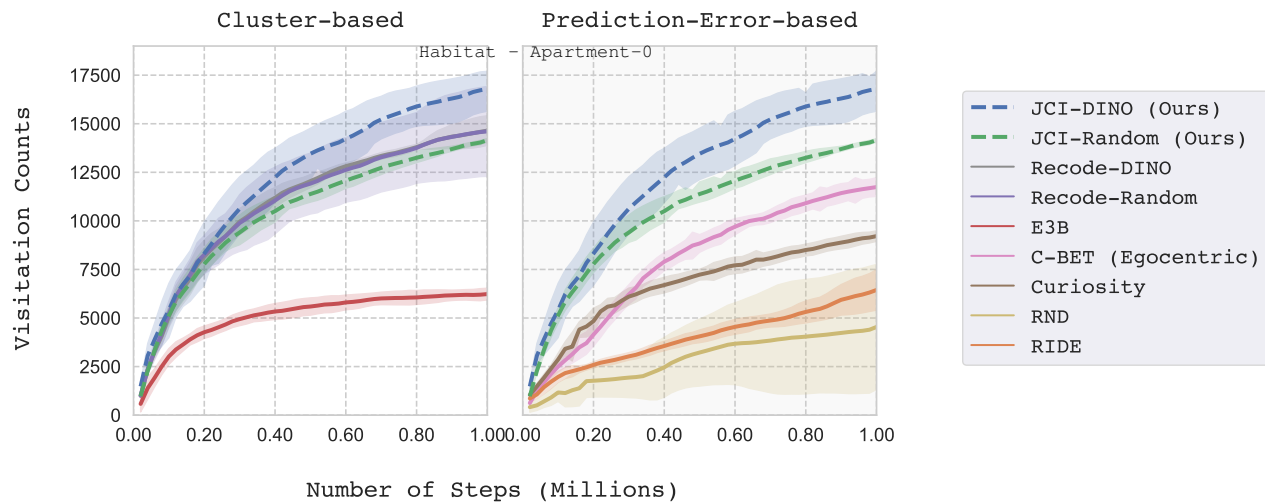


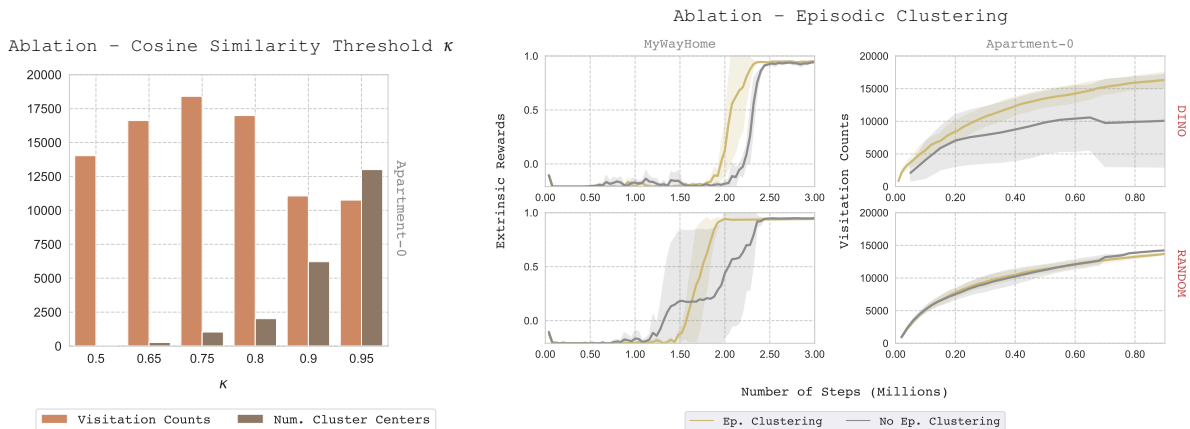
Figure 7.4: **Pre-trained DINO features excel with more complex observations:** When training on the Habitat environment we see that clustering with DINO embeddings is more effective than clustering with random features. Furthermore, only our method is able to leverage the pre-trained DINO features effectively. We argue that the priors present in the DINO embeddings help in aggregating the representations when building the episodic clusters, which ultimately determine the pseudo-counts.

the right side. For (i) we compare against RECODE [Saade et al., 2023] and E3B [Henaff et al., 2022]. For (ii) we compare against Random Network Distillation [Burda et al., 2018a], Curiosity-driven exploration [Pathak et al., 2017a] and RIDE [Raileanu and Rocktäschel, 2020a]. To see how other methods leverage pre-trained representations, we also compare against RECODE with both random features and DINO features. For all configurations we perform 3 runs with different seeds. We use PPO [Schulman et al., 2017] to train the policy network for both our method and our implementation of RECODE. E3B [Henaff et al., 2023] also uses DD-PPO for the Habitat environment. For all other methods the learning method for the policy network is IMPALA [Espeholt et al., 2018a]. Specific implementation details and hyperparameters can be found in Appendix B.3.

Evaluation metrics. For VizDoom we use the extrinsic reward as a metric for performance. The extrinsic reward is defined from 0 to 1. For Habitat, we explore only using the intrinsic reward, we thus use the number of visited states as a metric for performance as in Parisi et al. [2021]. We show the exact formula in Appendix B.2.

7.6.1 VizDoom Results—The Surprising Effectiveness of Random Features

We plot results for "MyWayHome" in Figure 7.3, where we compare our method to both cluster-based (left) and prediction-error-based methods (right) for both the "Sparse" and "VerySparse" settings. We see that clustering random features with our method (dashed, green curve) is superior to all other methods converging at around 1.5M and 1.8M steps



respectively. Interestingly, random features perform slightly better in VizDoom than the DINO features (dashed, blue curve) which converges slightly later at around 1.7M and 1.9M steps respectively. In this case, the priors in the DINO features do not seem to pose an advantage over random priors for the observations in the VizDoom environment. However, this also shows that random features in 3-D environments can be salient enough to be clustered effectively.

When comparing our method to RECODE with DINO features (grey curve), we see that our method is better able to leverage the pre-trained features. For RECODE, DINO features converge at around 5M steps compared to the random features (purple curve) which converge at 3.8M steps, while for our method random features and DINO features converge at almost the same time. For “VerySparse” we could not get RECODE to converge within the specified number of steps. One explanation for this could be that in the “VerySparse” setting, the agent never reaches the goal by chance as opposed to in the “Sparse” setting, such that the agent never finds a sensible policy. The RECODE agent does however manage to reach the goal on a few occasions. Furthermore, comparing our method to the prediction-error-based methods such as RIDE and Curiosity, we see that these methods perform worse needing at least 5M steps to converge. Overall, both our clustering method and RECODE, outperform the prediction-error-based methods. While E3B is a clustering method as well, it only performs episodic clustering as explained in Section 7.4. Usually, episodic clustering alone performs worse in singleton environments, such as VizDoom as explained in Henaff et al. [2023].

Figure 7.5: **Ablation studies: We show the effectiveness of clustering, especially episodic clustering:** (Left) Increasing the cosine similarity threshold κ increases the number of global clusters in the cluster table. If the number of clusters in the cluster table are too granular, performance decreases as shown by lower visitation counts for higher values of κ . (Right) Episodic clustering helps whenever there is structure in the representations. Especially for DINO in Habitat performance drops significantly, if episodic clustering is removed. This can also be seen in VizDoom (left panels) where episodic clustering helps both DINO and random features. For Habitat (right panels) with random features, episodic clustering has no large effect since the random features are not expressive enough for the complex observations.

7.6.2 Habitat Results—Pre-trained features help in complex observations

To test our method on more complex observations we use the Habitat environment and adopt the experimental setting from Parisi et al. [2021]. That is, we train the agent on the Habitat environment with only intrinsic rewards and plot the visitation counts (see Figure 7.4). We can see the benefit of using pre-trained DINO features in the visually more complex Habitat environment. When clustering pre-trained DINO features with our method (dashed, blue curve), we achieve 17K visitation counts on average, more than any other method. RECODE by comparison achieves only slightly under 15K visitation counts. Interestingly, our method with random features (dashed, green curve) performs slightly worse than RECODE at 14K visitation counts. We show in Figure 7.5 (right) that for Habitat, episodic clustering of random features is not better than episodic clustering of DINO features, hence the reduced performance compared to DINO features. However, we see again that our method is able to better leverage the pre-trained features compared to RECODE. Again, E3B is not suited for singleton environments barely achieving 7K visitation counts. Looking at the prediction-error-based methods, although Random Network Distillation uses random features as well, it is vastly inferior to the clustering-based methods converging at around 5K visitation counts. This validates our hypothesis about *transition saliency* presented in Section 7.2: In 3-D environments, the clustering-based methods are able to exploit the structure of the representations more effectively. We will expand on this in the following section.

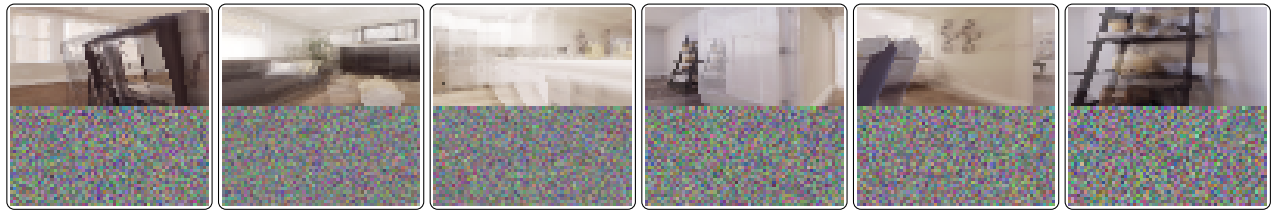
7.6.3 The Effect of Episodic and Global Clustering

Next, we analyse the effect of the episodic clustering and the global clustering to study the effectiveness of the latent structure extracted from the representations via the GMM. To ablate the episodic clustering, we perform runs for both "MyWayHome-VerySparse" and "Apartment-o" with and without episodic clustering. To ablate the influence of the global clustering, we vary the cosine similarity threshold κ and plot the number of cluster centers and visitation counts for "Apartment-o".

Increasing cosine similarity threshold κ makes pseudo-counts more granular, degrading performance. We plot the visitation counts and the number of clusters in the cluster table for different values of κ in Figure 7.5 (left). We see that increasing the cosine similarity threshold κ increases the number of global cluster centers in the cluster table. If the number of clusters in the cluster table is too high, performance decreases as shown by lower visitation counts for higher values of

κ . We argue that the pseudo-counts become too sparse, and thus the intrinsic reward fails to provide a good approximation of the state-space distribution. This aligns with the *transition saliency* problem in Section 7.2. In this case we revert to the same setting of prediction-error-based methods where the intrinsic reward becomes too granular. Note that the performance at around 10K visitation counts is still better than random network distillation [Burda et al., 2018a] in Figure 7.4, which is likely related to the fact that clustering-based methods are not affected by catastrophic forgetting.

Interestingly, for lower values of κ performance does not degrade in the same way. We observe that the smaller the value of κ is the less cluster centers get added to the cluster table. Hence the intrinsic reward is defined over fewer cluster centers. We see in Figure 7.5 (left) that up to $\kappa = 0.5$ performance remains comparable.



Episodic clustering is effective with good representations. We plot visitation counts and extrinsic reward with and without episodic clustering in Figure 7.5 (right). We see that episodic clustering helps whenever the underlying representations are structured, i.e., a structured latent space can be extracted via the GMM. This can be seen for Habitat (right panels) with DINO features where performance drops dramatically without episodic clustering (upper right). Even for VizDoom (left panels), episodic clustering helps both with DINO and random features, which shows that there is hidden structure to exploit in the comparatively simpler observations. For Habitat with random features, episodic clustering has no big effect since the random features are probably not expressive enough for the complex observations. We see this in Figure 4 (in the Appendix) where the random features are not clustered effectively compared to the DINO features. Overall, during our tests we found that 10% of the total number of steps in an episode is a good number of clusters for the Gaussian mixture model.

7.6.4 Clustering is Robust to Environmental Noise

To test the effect of random noise on exploration like in the noisy-TV problem [Pathak et al., 2017a], we adopt the experimental setup of Saade et al. [2023], where a frame of random noise is concatenated

Figure 7.6: **Visualization of DINO clusters for observations with random noise:** We plot different samples from the cluster table for the noisy observations in Habitat. Each image represents the mean image of an episodic cluster that was aggregated to the cluster table. The clustering happens on the 384-dimensional embedding. Even with the concatenated random noise the embeddings are still clustered sensibly.

to the current observation. Note that for each step the different frame with random noise is sampled. We show an example of the observation with the concatenated noise in Figure 7.7.

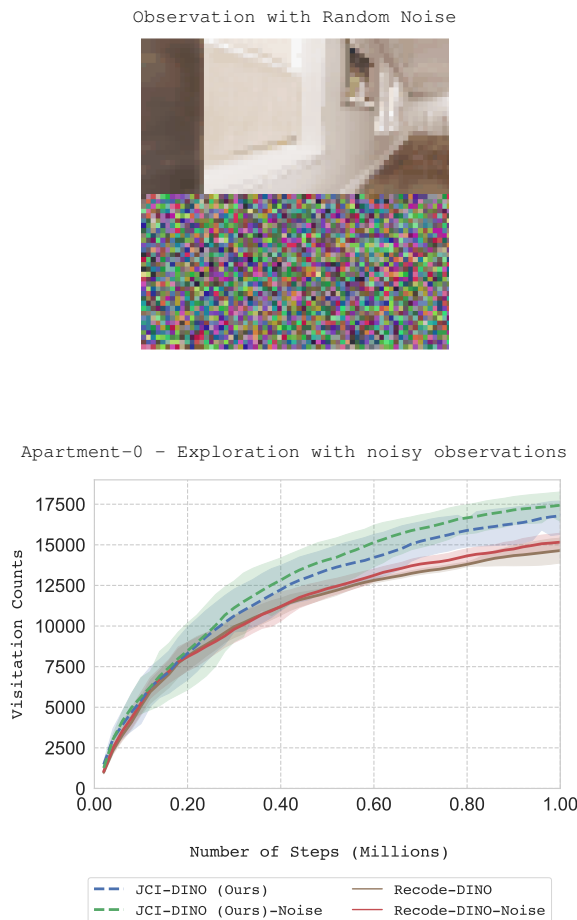


Figure 7.7: **We concatenate random noise to the observations:** To test the robustness of our method to observations with random noise, i.e, the noisy TV problem, we perform the same experiment as Saade et al. [2023] and concatenate a frame with random noise that changes after every step to the observation.

Figure 7.8: **Our method is robust to random noise in the observations:** We see that our method is robust to random noise in the observations. As a sanity check we also evaluate RECODE on noisy observations and confirm the results from Saade et al. [2023]. Suprisingly, for both methods the performance increases when adding random noise. We suspect the random noise has a regularising effect on the clustering.

Random noise does not affect exploration for clustering based methods.

We rerun the Habitat Apartment-o with noisy observations for both our method and RECODE in Figure 7.8. We show that our method is unaffected by random noise in the observations. Interestingly, performance slightly increases with noisy observations for both our method and RECODE. When looking at the number of cluster entries, we observe that when using observations with random noise the amount of clusters in the cluster table is lower than without the random noise. We therefore suspect that the random noise has a regularising effect on the clustering.

We also show the saved cluster centers in the cluster table with noisy observations in Figure 7.6. The quality of the clustering seems unaffected since similar observations are still aggregated correctly despite

the random noise.

7.7 Discussion

7.7.1 Conclusion

The starting point of our work is the insight that the magnitude of pixel-changes can be large in 3-D environments, but usually lacks in saliency. This is problematic in exploration with sparse rewards, since the agent is not able to distinguish between important and unimportant changes in the observations. To mitigate this problem, we introduce a data-based inductive bias based on episodic and global clustering of pre-trained representations, which extracts latent structure in the representations and allows for a less granular and more informative intrinsic reward.

We have shown that pre-trained representations and also random features can be leveraged effectively in environments with complex observations. We ablated different components of our method: we show that the cosine similarity threshold κ is vital to control the granularity of the clusters in the cluster table. We notice that performance degenerates if the number of clusters in the cluster table is too high, since the pseudo-counts become too sparse and the intrinsic reward too noisy. Furthermore, we found out that episodic clustering leverages the priors in the representations in order to produce better pseudo-count estimates, thus playing a crucial role for complex observations since the counts for the corresponding representations are aggregated more effectively.

7.7.2 Limitations and future work.

So far, we tested our method on singleton MDP's. We believe our method can also be extended to contextual MDPs, possibly by counting states separately on an episodic level. Overall, our method sheds light onto the problem of transition saliency in 3-D environments which is ubiquitous in exploration with sparse-rewards. In this sense, an interesting direction for future work could be investigating whether mitigating the transition saliency problem can enhance the effectiveness of prediction-error-based methods in 3-D environments. Hypothetically, clustering representations and using them as fixed targets for prediction-error-based methods should result in a more salient intrinsic reward. Future research should continue investigating which kinds of representations contain the best priors for exploration with sparse rewards. Potentially, multi-modal generative models could be used to learn representations that not only capture visual invariances, but also environmental dynamics.

Realigning Pre-Trained Vision Models with Object-Centric Representations for Multi-Object Instance Retrieval

Corresponding publication. This chapter is based on the work:

Oh-A-DINO: Understanding and Enhancing Attribute-Level Information in Self-Supervised Object-Centric Representations

Stefan Sylvius Wagner, Stefan Harmeling

Submitted to International Conference on Learning Representations 2026

Previously accepted to 7th Workshop on Data, Learning and Inference (DALI)

Personal contributions. Stefan Sylvius Wagner, did literature review. Stefan Sylvius Wagner developed and implemented the method and conducted the experiments. Stefan Sylvius Wagner wrote the initial draft of the document, which was then edited with Stefan Harmeling.

Abstract. Object-centric understanding is fundamental to human vision and required for complex reasoning. Traditional methods define slot-based bottlenecks to learn object properties explicitly, while recent self-supervised vision models like DINO have shown emergent object understanding. We investigate the effectiveness of self-supervised representations from models such as CLIP, DINOv2 and DINOv3, as well as slot-based approaches, for multi-object instance retrieval, where specific objects must be faithfully identified in a scene. This scenario is increasingly relevant as pre-trained representations are deployed in downstream tasks, e.g., retrieval, manipulation, and goal-conditioned policies that demand fine-grained object understanding. Our findings reveal that self-supervised vision models and slot-based representations excel at identifying edge-derived geometry (shape, size) but fail to preserve non-geometric surface-level cues (colour, material, texture), which are critical for disambiguating objects when reasoning about or selecting them in such tasks. We show that learning an auxiliary latent space over segmented patches, where VAE regularisation enforces compact, disentangled object-centric representations, recovers these

missing attributes. Augmenting the self-supervised methods with such latents improves retrieval across all attributes, suggesting a promising direction for making self-supervised representations more reliable in downstream tasks that require precise object-level reasoning.

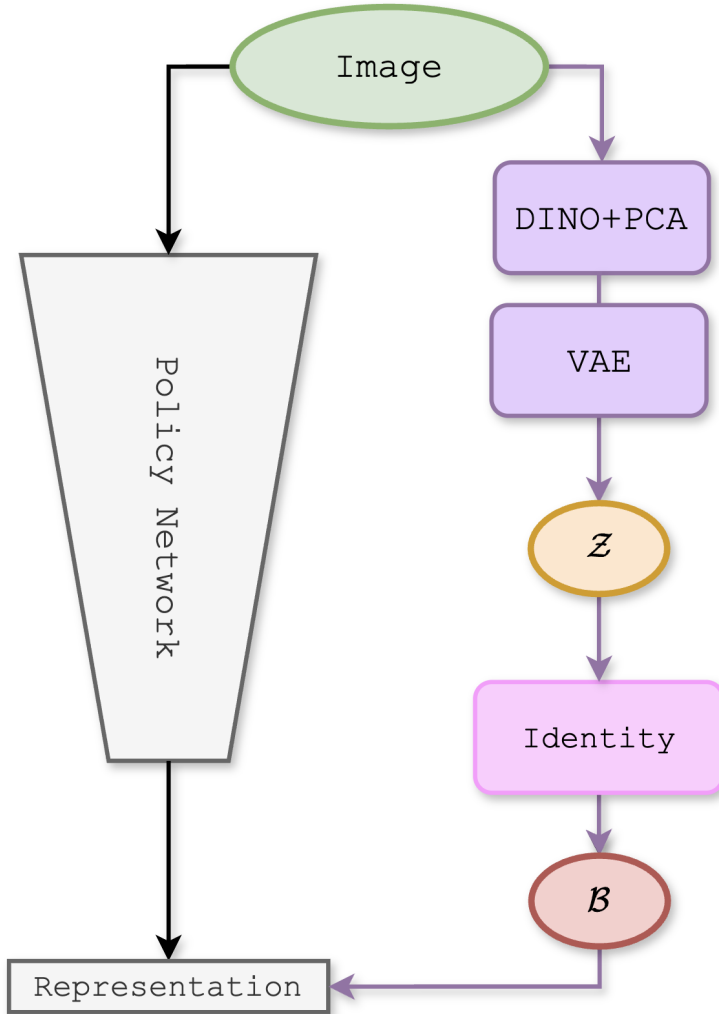


Figure 8.1: Oh-A-Dino learns a continuous latent space of fine-grained object features such as colour and material to enhance existing self-supervised learnt representations. The enhanced representations are effective at retrieving objects with multiple attributes in multi-object instance retrieval.

8.1 Preamble

Motivated by the observation that pre-trained and slot-based representations struggle at retrieving certain object attributes consistently, this method proposes a data-based inductive bias to improve fine-grained object understanding by combining pre-trained features with object-level features learnt in an unsupervised fashion.

As shown in Figure 8.1, we use a Variational Autoencoder (VAE) as structure inducing function T to learn object-level latent features

from image patches extracted with the help of the pre-trained DINO representations using PCA. We thus leverage the ability of the VAE as a latent variable model to systematically extract relevant object features from the image patches. The learned latent vector can then be used as learning component by simply concatenating the object-level representations with the original pre-trained representations. We show that this is due to the VAE features learning strong latent structure for colour and material, which complements the self-supervised learnt features. The data-based inductive bias thus mitigates the shortcomings of the pre-trained representations by augmenting the representations with finer-grained object understanding without requiring full retraining of the self-supervised model.

8.2 Introduction

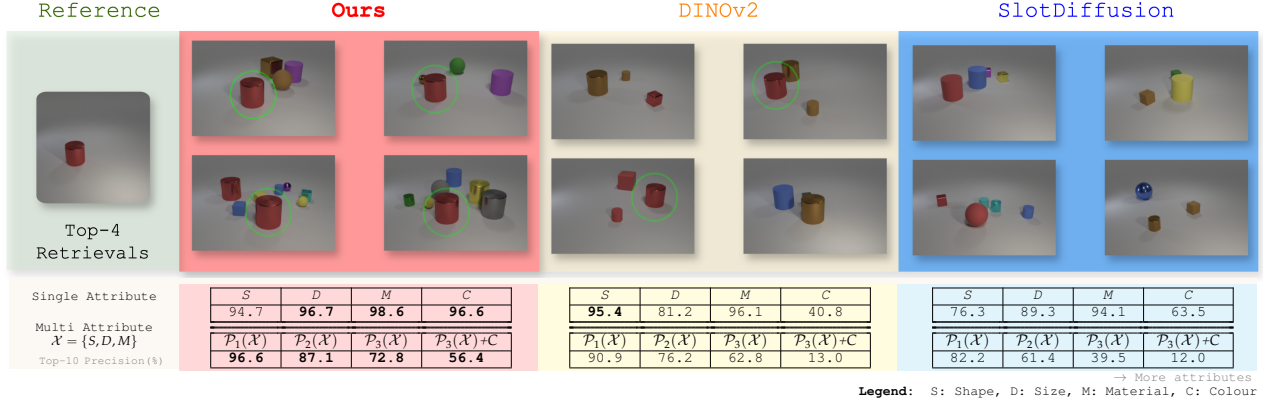
Object-centric understanding is central to human vision and a prerequisite for complex reasoning [van Steenkiste et al., 2019b, Schölkopf et al., 2021]. Learning such representations remains challenging: slot-based methods explicitly disentangle object properties [Locatello et al., 2020, Wu et al., 2023], but may compromise global scene understanding [Montero et al., 2024], while large self-supervised models such as DINO and DINOv2 [Caron et al., 2021b, Oquab et al., 2023b] show emergent object structure without task-specific supervision. This makes them attractive as general-purpose vision backbones, increasingly used in downstream tasks that require object-level reasoning [Uelwer et al., 2023, Seitzer et al., 2023, Zhou et al., 2024, Wagner and Harmeling, 2024, Szot et al., 2023].

Many of these downstream tasks such as retrieval, embodied manipulation, or goal-conditioned policies depend on representations that can reliably capture the properties needed to disambiguate between multiple objects in a scene. Whether current pre-trained and object-centric representations provide this level of fidelity, however, remains unclear.

We ask: *Do existing representations preserve the attribute-relevant information needed to reliably disambiguate objects in multi-object settings?*

Our findings reveal that while self-supervised vision models and slot-based representations excel at capturing geometric structure such as shape and size, they fail to preserve non-geometric surface-level cues such as colour and material. This leads to systematic errors when disambiguating otherwise similar objects. To address this, we propose *Object-Aware-DINO (Oh-A-DINO)* which augments self-supervised representations from DINOv2-S with object-centric latent vectors learned from segmented patches using a Variational Autoencoder [Kingma et al., 2019] (VAE). The VAE’s regularisation encourages a compact

latent space that captures these fine-grained properties, and concatenating the latents with global features improves alignment across all object attributes. We demonstrate that this approach improves multi-



object instance retrieval on CLEVR [Johnson et al., 2017] and CLEVRText [Karazija et al., 2021], with particularly notable gains in colour and material matching. We also show that these improvements transfer to real-world instance retrieval settings on the Stanford Cars [Krause et al., 2013] dataset. Overall, this highlights a broader challenge: widely used pre-trained representations, though powerful, may miss the kinds of attribute-level information that downstream tasks need for accurate object-level reasoning.

8.3 Contributions

1. We identify a key limitation of both self-supervised and slot-based representations: they capture geometric attributes well but fail to preserve non-geometric object attributes such as colour and material, which are critical for disambiguating objects in multi-object scenes.
2. We propose a simple and modular method to augment self-supervised representations with object-centric latents learned from a VAE, without retraining the backbone.
3. We demonstrate that this approach improves retrieval across all attributes, providing more reliable object-level representations and suggesting a promising direction for enhancing pre-trained representations in downstream tasks that require detailed object reasoning.

Figure 8.2: DINOv2 representations and slot-based representations struggle at multi-object instance retrieval, due to weak object-specific features (largely colour) in its embeddings. Our method is able to retrieve more relevant images by combining general scene and local representations. DINOv2 representations excel at retrieving images where multiple attributes match such as shape and size, while retrieval degrades when adding colour. Slot-based representations lack specialisation, where retrievals sometimes match fine-grained features such as colour but often failing to retrieve a similar object altogether. Our method performs the best being able to augment the DINOv2 representation to mitigate its shortcomings.

8.4 Related Work

Slot-based object centric learning. Recent advances in object-centric representation learning (e.g., SlotAttention [Locatello et al., 2020], MONet [Burgess et al., 2019], SlotDiffusion [Wu et al., 2023], AdaSlot [Fan et al., 2024], Slot-VAE [Wang et al., 2023]) have enabled models to decompose images into object-level features, making it possible to reason about individual objects within a scene. These methods mostly focus on learning disentangled representations of individual objects by enforcing an object-level slot-based bottleneck via cross attention, which then allows for faithful image reconstruction, generation of novel object configurations, or object segmentation. In order to extract the objects, the image is usually tokenised and an attention mechanism is applied to the tokens (e.g. SLATE [Singh et al., 2021]). A slot decoder then reconstructs the original image from the slot representation. STEVE [Singh et al., 2022] reconstructs the image from the slot-representation autoregressively to learn dependencies between different image patches i.e. tokens. More recent methods, have further exploited this by attempting to incorporate global scene level features into the representations [Chen et al., 2024] or by adding compositional reasoning into the slot-based representations [Jung et al., 2024].

Combining self-supervised features with task-specific features. Recent methods such as SLATE [Singh et al., 2021] and STEVE [Singh et al., 2022] have leveraged vision transformers for object-centric learning to learn improved slot representations that also capture global scene information via attention mechanisms. These methods have shown improved performance on object-centric tasks, but they require retraining the complete model, which can be computationally expensive and time-consuming. Similarly, DINOSAUR [Seitzer et al., 2023] and GOLD [Chen et al., 2024] use DINO representations to provide better natural priors to learn object-centric representations on natural images. R²Former [Zhu et al., 2023] learns global correspondences between patches from a transformer pre-trained on ImageNet and then leverages local attention maps to rerank retrieved patches of candidate images. Zhang et al. [Zhang et al., 2023] combine learnt Stable Diffusion and DINOv2 features to enhance instance retrieval. We build on this work by combining representations that occupy different semantic spaces, unlike the previously studied Stable Diffusion and DINOv2 features which share similar characteristics. This integration of semantically distinct vectors (i.e. general features and object-level features) enhances our model’s capabilities.

Assessing the quality of representations with instance retrieval. Instance retrieval is a challenging task that requires models to retrieve images with similar object attributes and configurations. Traditionally, this task has been best solved by methods that leverage representations from pre-trained models, such as ResNet [He et al., 2016] or vision transformers [Dosovitskiy et al., 2020]. Generally, instance retrieval depends on learning a good latent space or representation of the image, where retrievals are ranked on a similarity measure [Chen et al., 2021]. Naturally, instance retrieval has been used to evaluate the quality of representations in self-supervised models [Caron et al., 2021b, Oquab et al., 2023b]. Since we are interested in evaluating the quality of the representations for multi-object understanding, we choose the task of multi-object instance retrieval. Beyond traditional vision applications, multi-object instance retrieval plays a crucial role in goal-conditioned reinforcement learning (GCRL) and robotic tasks [Zheng et al., 2024]. In GCRL, agents often rely on goal images to specify desired configurations of objects in the environment, particularly in tasks requiring manipulation or navigation in multi-object settings [Le Paine et al., 2019]. Similar challenges are present in hierarchical reinforcement learning (HRL), where sub-goals often involve interacting with specific object sets. Robust multi-object retrieval can inform option selection and improve learning efficiency [Hafner et al., 2022]. Moreover, the emergence of embodied agents has seen the need for pre-trained representations that understand nuance in their environment [Szot et al., 2023, Szot et al., 2024].

8.5 Analysing Representation Quality with Object-Centric Instance Retrieval

To evaluate how well existing methods capture object-centric detail, we compare DINOv2 [Caron et al., 2021b], DINOv3 [Siméoni et al., 2025], CLIP [Radford et al., 2021], and SlotDiffusion [Wu et al., 2023] representations in multi-object instance retrieval. We use the CLEVR and CLEVRText datasets, which provide attribute labels (shape, size, material, colour), enabling us to probe whether these properties are faithfully preserved in the learned embeddings. Beyond single attributes, we also test combinations, since downstream applications often require correctly binding multiple attributes to the same object.

Figure 8.2 illustrates the setup: the top row shows qualitative examples of top-4 retrievals for a single reference object, while the bottom row reports top-10 precision across individual attributes and attribute subsets. Formally, we compute means over $\mathcal{P}_i(\mathcal{X}) = \{A \in \mathcal{P}(\mathcal{X}) : |A| = i\}$, where $\mathcal{X} = \{\text{shape, size, material, colour}\}$, and extend the evaluation to ordered permutations. We further present extended results in Table

8.3.

Attrib. Ablation Top-10 Precision (%)	CLEVR				CLEVRTex			CLEVR			CLEVRTex	
	Shape	Size	Mat.	Col.	Shape	Size	Mat.	$\mathcal{P}_2(\mathcal{X})$	$\mathcal{P}_3(\mathcal{X})$	$\mathcal{P}_3(\mathcal{X})+C$	$\mathcal{P}_2(\mathcal{X})$	$\mathcal{P}_2(\mathcal{X})+C$
Oh-A-DINOv2 (Ours)	94.7 $\pm$ 2.0	96.7 $\pm$ 2.2	98.6 $\pm$ 0.0	96.6 $\pm$ 2.6	90.7 $\pm$ 2.1	85.3 $\pm$ 1.5	45.8 $\pm$ 3.9	85.2 $\pm$ 2.1	72.8 $\pm$ 1.4	56.4 $\pm$ 1.6	56.0 $\pm$ 0.2	20.3 $\pm$ 2.6
DINOv2	95.4 $\pm$ 1.1	81.2 $\pm$ 1.2	96.1 $\pm$ 1.0	40.8 $\pm$ 2.1	87.3 $\pm$ 3.8	80.3 $\pm$ 2.6	24.2 $\pm$ 1.9	76.2 $\pm$ 3.1	62.8 $\pm$ 1.2	13.0 $\pm$ 1.1	54.0 $\pm$ 0.1	12.2 $\pm$ 2.1
DINOv3	96.9 $\pm$ 1.3	85.5 $\pm$ 1.5	97.0 $\pm$ 1.2	44.5 $\pm$ 1.9	81.3 $\pm$ 0.7	86.8 $\pm$ 1.2	11.8 $\pm$ 1.1	80.1 $\pm$ 1.2	64.8 $\pm$ 0.6	13.7 $\pm$ 1.2	44.5 $\pm$ 0.2	4.2 $\pm$ 1.5
CLIP	83.8 $\pm$ 2.3	83.2 $\pm$ 1.8	98.3 $\pm$ 0.2	66.8 $\pm$ 1.7	81.4 $\pm$ 1.1	90.1 $\pm$ 1.4	15.0 $\pm$ 1.1	63.0 $\pm$ 0.8	40.9 $\pm$ 0.9	14.6 $\pm$ 1.5	43.9 $\pm$ 0.3	3.1 $\pm$ 0.9
SlotDiff.	76.3 $\pm$ 1.2	89.3 $\pm$ 1.3	94.1 $\pm$ 1.5	63.5 $\pm$ 0.4	67.0 $\pm$ 1.7	80.2 $\pm$ 1.6	6.6 $\pm$ 0.9	56.3 $\pm$ 0.4	39.5 $\pm$ 1.3	12.0 $\pm$ 1.2	33.9 $\pm$ 2.2	1.4 $\pm$ 0.4

Single-Attribute Retrieval Multi-Attribute Retrieval

Colour is the weakest attribute across models. DINOv2 and DINOv3 both perform strongly on shape (95.4%, 96.9%) but fall sharply on colour (40.8%, 44.5%). This indicates that newer self-supervised objectives have not fully addressed colour encoding. CLIP improves colour substantially (66.8%), but at the cost of weaker shape and size performance compared to DINO models (83.8% shape, 83.2% size). SlotDiffusion also retrieves colour better than DINOv2 (63.5% vs. 40.8%), yet underperforms on shape (76.3%), suggesting a trade-off between geometric and surface-level features.

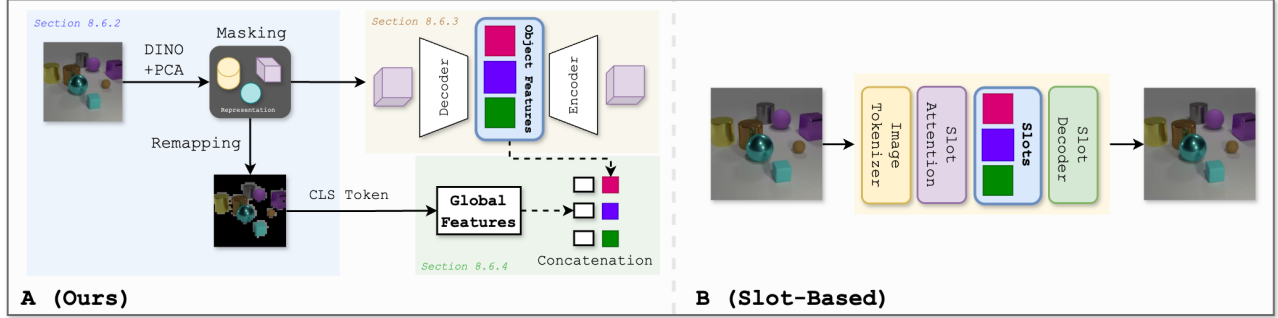
Slot-based methods struggle with attribute binding. In multi-attribute retrieval, SlotDiffusion degrades rapidly: while colour can be matched in isolation, it fails to consistently bind multiple attributes to the same object. Visual inspection confirms that while DINOv2 and DINOv3 retrievals often fail only by colour mismatch, SlotDiffusion frequently retrieves objects that share little resemblance with the query, indicating difficulties in binding attributes.

CLIP is inconsistent across domains. CLIP shows relatively strong performance on single attributes like material (98.3%), but its precision collapses when tested on CLEVR multi-attribute settings, particularly when colour is included (14.6%). This suggests that CLIP’s text-image alignment biases help capture some surface cues, but generalisation to synthetic multi-object scenes is weaker.

Overall, these results reinforce the broader picture: self-supervised and slot-based representations excel at encoding geometry but consistently neglect surface-level cues such as colour and material, which are critical for disambiguating otherwise similar objects. This motivates our method: instead of altering pre-training, which is expensive and dataset-specific, we introduce a complementary object-centric latent space. We focus on DINOv2-S as the foundation, since we will show that it delivers the most faithful attribute binding overall and can be

Figure 8.3: Object-level features improve prediction for all attributes, while also improving retrieval of multiple simultaneous attributes. The main improvement is achieved on the colour(texture) attribute, which indicates that our method mitigates for the lacking learning signal in the other methods’ representations. Slot-based methods perform worse than the pretrained DINOv2 representations when predicting multiple attributes simultaneously.

augmented most effectively. By training a Variational Autoencoder (VAE) on image patches segmented from DINOv2 features, and augmenting the global DINO embeddings with these latents, we obtain representations that recover fine-grained attributes while preserving global scene context.



8.6 Enhancing DINOv2 with Object-Centric Representations

Our approach consists of a multi-step process which is described in Figure 8.4. In Section 8.6.2, we first leverage the general object understanding of the DINOv2 representations to segment the image around important objects. For this, we apply PCA on the DINOv2 features of t consecutive images and use the first principal component to extract the object-level features. This allows us to extract a segmented image that is then split up into image patches. In Section 8.6.3, we learn a latent space over these image patches using a Variational Autoencoder (VAE). The VAE is trained to reconstruct the segmented image patches to learn a latent space at the object level. This latent space can be seen to capture the object-level attributes of the image, which we then seek to combine with the global features of the DINOv2 representations. In Section 8.6.4, we then concatenate the global features of the DINOv2 representations with the latent vectors learnt from the VAE to create a new and more informative representation.

8.6.1 Preliminaries

Given an image $x \in \mathbb{R}^{h \times w \times c}$, the first step is to retrieve the DINOv2 representations from a pre-trained DINOv2 model, which we define as a function $g : \mathbb{R}^{h \times w \times c} \rightarrow \mathbb{R}^{p_h \times p_w \times n_y}$, where $y \in \mathbb{R}^{p_h \times p_w \times n_y}$ are the $p_h \times p_w$ resulting patch embeddings each with an embedding dimension of n_y . The number of patches $p_h \cdot p_w$ depends on the chosen patch size $s_h \times s_w$. In the following, we either view the output of g as $y \in$

Figure 8.4: **Our method leverages the implicit general object understanding of DINOv2 to extract object-level features for multi-object instance retrieval.** **B:** Traditional slot-based object-centric methods learn slot-representations via cross-attention which provide little inductive bias, while having to compress global and object-level features into a small set of latents. **A:** We propose combining general self-supervised features with learnt object-level features to obtain an improved latent representation. The object-level features are learnt from image patches making training efficient and the latent space expressive.

$\mathbb{R}^{p_h \times p_w \times n_y}$ or as $y \in \mathbb{R}^{p \times n_y}$ where $p = p_h \cdot p_w$ depending on whether a list of patches is sufficient.

The DINOv2 embeddings are quite high-dimensional, hence we apply PCA to the embedding to capture the most relevant object-level features in the principal component (see supplementary material A.2). For our purposes, three components are sufficient to obtain a low-dimensional representation that corresponds to the image space. For stability, we compute the PCA components on a batch of t different images,

$$X_{1:t} = [x_1, x_2, \dots, x_t] \in \mathbb{R}^{t \times h \times w \times c}, \quad Y_{1:t} = g(X_{1:t}) \in \mathbb{R}^{t \times p \times n_y}, \quad (8.1)$$

where $x_i \in \mathbb{R}^{h \times w \times c}$ is the i -th image in the batch and $Y_{1:t}$ are the patch embeddings of the t images. We sample $X_{1:t-1}$ from the dataset and append the image of interest x as the last image x_t we want to extract the object-level features from. In our experiments, we set $t = 50$. We omit the indices and write Y where appropriate. In the following, we describe how we use PCA to extract the object-level features from y_t , i.e., the DINOv2 patch embeddings of a given image.

8.6.2 Extracting Object-Level Features with PCA

Extracting object-level features from the DINOv2 patch embeddings consists of three steps: **(i)** Separating background and foreground by applying PCA to the DINOv2 patch embeddings and thresholding the first component to obtain a segmentation mask. **(ii)** Reapplying PCA to the foreground patch embeddings to obtain a segmentation mask that is consistent across objects. **(iii)** Remapping the segmentation mask to the original image size and splitting the image into p image patches. All steps are described in detail below.

(i) Separating foreground and background. Segmentation masks have been extracted from DINO features in several ways [Siméoni et al., 2021, Wang et al., 2023]. For our purposes, we present a segmentation method that does not require training an additional model and leverages the DINO patch features through PCA. Since we want to apply PCA to the collected patches of t images, we flatten Y over the first two dimensions and apply PCA to get

$$Z' = \text{PCA}(Y) \in \mathbb{R}^{(t \cdot p)}, \quad (8.2)$$

of the DINOv2 patch embeddings. Interestingly, the first principal component carries the information what pixels belong to foreground objects. This insight can be exploited to separate foreground objects from the background: to obtain the segmentation mask we threshold the first principal component Z' against the median of its components

(i.e., $\text{median}(Z') \in \mathbb{R}$). More precisely, we define the segmentation mask to extract the foreground (abbr. as “fg”) as follows

$$\hat{M}^{\text{fg}} = \mathbb{1}(Z' > \text{median}(Z')) \in \{0, 1\}^{(t \cdot p)} \quad (8.3)$$

where the function $\mathbb{1}$ translates a boolean vector into a zero/one vector. In the following, we use the segmentation mask $\hat{M}^{\text{fg}}$ to extract the foreground objects and perform a second PCA to obtain a consistent segmentation mask across objects.

(ii) Reapplying PCA to the foreground patch embeddings. While the previous step already delivers a decent segmentation mask, we found that reapplying PCA to the foreground patch embeddings delivers a more consistent segmentation mask across objects, i.e., one that does not cut across objects. We thus define the foreground patch embeddings as

$$Y^{\text{fg}} = \hat{M}^{\text{fg}} \odot Y \in \mathbb{R}^{(t \cdot p) \times n_y}, \quad (8.4)$$

where the $\odot$ is the component-wise multiplication that is broadcasted according to the size of Y . The result Y^{fg} contains the foreground patch embeddings with the background patches zeroed out. We then reapply Equations (8.2) and (8.3) to Y^{fg} and obtain a new segmentation mask $M^{\text{fg}} \in \{0, 1\}^{(t \cdot p)}$ that improves $\hat{M}^{\text{fg}}$ by being more faithful with respect to objects.

(iii) Remapping the masked patch embeddings to the original image to get image patches that contain objects. One of the benefits of using a vision transformer architecture is that the patch embeddings can be remapped to the original image. We thus remap the segmentation mask M^{fg} to the original image to extract a segmented image to calculate global features and to extract image patches to learn object-level features with a VAE.

Initially, we reshape the masking vector M^{fg} to a $t \times p$ matrix. Since we are only interested in the last image x_t (see Section 8.6.1), we choose the last row of M^{fg} which we call $m \in \mathbb{R}^p$. This vector m is the segmentation mask of the last image $x_t \in \mathbb{R}^{h \times w \times c}$ which was our initial image x . More precisely, the mask determines what patches are relevant for the objects, i.e., an entry in m tells us whether a particular patch is masked or not. To obtain a masked image, we expand m to the size of x , by creating patches with zeros or ones, respectively, for each mask position. We thus reverse the operations in Section 8.6.1 and upscale the segmentation mask to the original image size. More precisely, we insert patches of size $s_h \times s_w$ at each patch position $p_h \times p_w$ to upscale the mask to the original image size. We define a function g' for this purpose $g' : \mathbb{R}^p \rightarrow \mathbb{R}^{(p_h \times p_w) \times (s_h \times s_w)}$, where $p = p_h \cdot p_w$. Applying g'

we get

$$m^0 = g'(m), \quad (8.5)$$

where $m^0 \in \mathbb{R}^{(p_h \times p_w) \times (s_h \times s_w)}$ is the remapped segmentation mask. In other words, each entry in m^0 is a patch of size $s_h \times s_w$ filled either with ones or with zeroes and all patches correspond to non-overlapping regions in the original image. We can now obtain the segmented image and the image patches by applying element-wise multiplication between the original image splitted into patches $x \in \mathbb{R}^{(p_h \times p_w) \times (s_h \times s_w) \times c}$ and the segmentation mask m^0

$$x^0 = x \odot m^0. \quad (8.6)$$

(again broadcasting over the extra dimension of the colour channels) where $x^0 \in \mathbb{R}^{(p_h \times p_w) \times (s_h \times s_w) \times c}$ is the segmented image. Finally, x^0 can easily be reshaped to the original image dimensions $h \times w \times c$. We also keep a list of all image patches $\rho \in \mathbb{R}^{p \times (s_h \times s_w) \times c}$ where each element of size $s_h \times s_w \times c$ is an image patch containing an object segment. We now use the **segmented image** x^0 to (i) calculate the global features of the image in Section 8.6.4 and (ii) learn the object-level features with a VAE using the **image patches** ρ .

8.6.3 Learning Object-Level Latent Representations with a Variational Autoencoder

VAE Architecture. To capture the object-level attributes of the segmented image patches, we employ a Variational Autoencoder (VAE). The VAE is trained to reconstruct the extracted image patches, hereby learning a compact latent space that encodes meaningful object attributes. The VAE consists of an encoder $q_\phi(z|\rho)$ that maps a list $\rho \in \mathbb{R}^{p \times s_h \times s_w \times c}$ of image patches (list index from 1 to p) to a list of latent vectors (arranged as a matrix) $z \in \mathbb{R}^{p \times n_z}$, and a decoder $f_\theta(\rho|z)$ that reconstructs the input patches. Here, n_z is the embedding dimension of the VAE. The training objective is given by:

$$\mathcal{L}_{\text{VAE}} = \mathbb{E}_{z \sim q_\phi(z|\rho)} [\log f_\theta(\rho|z)] - \beta D_{\text{KL}}(q_\phi(z|\rho) || f(z)), \quad (8.7)$$

where $f(z) = \mathcal{N}(z; 0, 1)$ is a Gaussian prior, and β controls the regularization strength. We set β to a small value to reduce regularisation so object-level specific clusters can emerge.

Training with object patches. To train the VAE we convert our images $x_1, \dots, x_t$ into t lists of patches

$$\rho_1, \rho_2, \dots, \rho_t \in \mathbb{R}^{p \times s_h \times s_w \times c}. \quad (8.8)$$

Each patch is resized to a fixed resolution and normalized before being fed into the encoder. The model is trained using the reparameterization trick and the Adam optimizer. After training, the encoder q_ϕ is used to map the segmented patches to a list of *latent vectors* (arranged as a matrix)

$$z \in \mathbb{R}^{p \times n_z}. \quad (8.9)$$

These vectors provide a structured representation of object attributes, which can be combined with the global DINOv2 features to enhance downstream tasks.

8.6.4 Combining Global and Local Representations

So far, we obtained the *object-level features* in form of a matrix $z \in \mathbb{R}^{p \times n_z}$. The general scene understanding is contained in the CLS token $\epsilon \in \mathbb{R}^{n_g}$ which is obtained by passing the segmented image x^o through the DINOv2 model. To combine the global and local features, we need to repeat ϵ exactly p times to match the number of object-level latent vectors:

$$z \in \mathbb{R}^{p \times n_z}, \quad e = [\epsilon_1, \dots, \epsilon_p] \in \mathbb{R}^{p \times n_g}.$$

we now have all the necessary components to create a new representation that combines global and local features. For this we concatenate z and e to obtain the final representation

$$v = [e, z] \in \mathbb{R}^{p \times (n_g + n_z)} = \mathbb{R}^{p \times n_v}.$$

Summarising, v is a structured representation that combines global and local features with p embedding vectors of dimension $n_v = n_g + n_z$. This combined representation can now be used for instance retrieval by calculating the cosine similarity between query and candidate images as in [Caron et al., 2021b], which entails calculating the pairwise cosine similarity between the entries in v, v' yielding a cosine similarity score. We describe this process in more detail for completeness in Appendix D.1.

8.7 Experiments

8.7.1 Datasets

A key challenge in evaluating object-centric representations is finding datasets that expose systematic variation in object attributes while remaining relevant for real-world tasks. Most natural-image benchmarks lack consistent multi-object scenes with fine-grained labels, while purely synthetic datasets may be dismissed as too controlled. We therefore combine synthetic datasets for diagnostic evaluation with a real-world dataset for external validation.

CLEVR [Johnson et al., 2017] provides rendered 3D scenes where objects vary along shape, size, material, and colour. It offers clearly disentangled factors that make it possible to probe which attributes are preserved in the embeddings.

CLEVRText [Karazija et al., 2021] extends CLEVR by introducing realistic textures, increasing visual complexity and making attribute retrieval more challenging. Together, CLEVR and CLEVRText serve as controlled testbeds to diagnose representation quality in multi-object settings.

Stanford Cars [Krause et al., 2013] complements these synthetic datasets with a real-world benchmark. Many car classes share near-identical geometry and differ only in colour and surface-level details, which directly tests whether representations capture the appearance-level attributes needed to disambiguate objects.

For training, we fit the VAE on 2,000 CLEVR/CLEVRText images segmented into object patches. Evaluation is performed on 500 query images and 5,000 candidates for retrieval, with Stanford Cars used only for evaluation under its standard splits.

8.7.2 Experimental Setup

Retrieval tasks. We evaluate representations on a retrieval task where the goal is to recover objects with matching attributes given a reference image. For CLEVR and CLEVRText, attribute labels allow systematic evaluation of **shape**, **size**, **material**, and **colour** (CLEVR) or **shape**, **size**, and **material** (CLEVRText, with material combining texture and colour). For Stanford Cars, which lacks attribute annotations, we focus on appearance-level cues by computing the average deviation in RGB pixel values to assess colour consistency. Dataset splits are reported in Appendix D.2.

Part I: Synthetic diagnostic benchmarks. On CLEVR and CLEVRText, we carry out both (i) *performance evaluation* and (ii) *attribute-level analysis*. First, we measure retrieval quality using Top-10 precision, weighted precision, and error rate across all attribute subsets, probing how well each model preserves object identity. Second, we perform an ablation study which was presented in Table 8.3 to examine retrieval fidelity for individual attributes and combinations, highlighting failures in binding surface-level cues (e.g. colour, texture). In this context, we analyse the effect of using standalone VAE features and show whether VAE-augmented variants of DINOv2, DINOv3, CLIP, and SlotDiffusion improve attribute alignment. Finally, we test whether increasing backbone capacity (DINOv2-L) helps encode surface-level features.

Part II: Real-world benchmark. To validate beyond synthetic settings, we repeat our retrieval experiments on the Stanford Cars dataset. Here, cars of the same class often share nearly identical geometry but differ in colour and surface details, providing a natural testbed for appearance-sensitive retrieval. This complements the controlled synthetic benchmarks and demonstrates that the limitations we identify are not confined to toy datasets. In particular, we measure the *average Euclidean distance in colour space* between the query and retrieved instances.



Figure 8.5: Visualisations of retrievals for CLEVRText, augmenting with VAE object level features allows for retrieval in complex multi-object scenes. Oh-Ah-Dino consistently retrieves the object in question with the correct properties such as the shape, size and material. DINOv2 on the other hand often retrieves the correct object but with the wrong texture.

Evaluation metrics. We report three metrics:

1. *Top-10 Precision*: fraction of the first ten retrieved images that match the reference attributes;
2. *Weighted Precision*: same as above, but weighted by retrieval rank ($1/i$ for rank i), emphasising higher-ranked matches;
3. *Error Rate*: percentage of queries for which no correct retrieval is found in the top ten (for CLEVRText, normalised for queries with fewer than ten valid candidates).

Main Results (%)	CLEVR			CLEVRText		
	Top-10 Precision↑	Weighted Precision↑	Error Rate↓	Top-10 Precision↑	Weighted Precision↑	Error Rate↓
Oh-A-DINOv2 (Ours)	56.4±2.0	49.0±2.2	1.0±0.0	20.3±2.6	11.2±1.4	41.0±1.0
DINOv2	13.0±1.1	13.8±1.2	28.0±1.0	12.2±2.1	8.0±1.3	55.6±0.7
DINOv3	13.7±1.2	14.5±1.3	28.0±2.0	4.2±1.1	1.5±0.3	82.4±1.9
CLIP	14.7±1.5	16.3±1.6	29.2±1.9	2.9±1.2	0.7±0.2	90.6±0.4
SlotDiff.	12.0±1.2	12.7±1.3	33.6±1.5	1.4±0.4	0.9±0.2	93.4±0.2

Baselines. For each method we run seven trials with 50 query images and 5,000 candidate images, sampling queries without replacement. SlotDiffusion is pre-trained separately on CLEVR and CLEVRText. Seeds are chosen at random and kept consistent across models. We report standard deviation across runs.

Figure 8.6: Oh-A-DINOv2 improves over SSL and slot-based features for multi-object instance retrieval. Our method outperforms both self-supervised features and slot-based approaches. The table shows that Oh-A-DINOv2 achieves higher precision and lower error rates on CLEVR and CLEVRText, indicating that combining DINOv2 with object-level VAE features addresses the limitations of pure SSL features and slot-attention methods.

8.7.3 CLEVR and CLEVRText: Multi-Object Instance Retrieval

Our approach improves multi-object instance retrieval. We show in Table 8.6 that the learnt object features are expressive and complement the DINOv2 representations well. On CLEVR, **Oh-A-DINOv2** achieves a Top-10 Precision of **56.4%**, compared to only **13.0%** for **DINOv2**, **13.7%** for **DINOv3**, **14.7%** for **CLIP**, and **12.0%** for **SlotDiffusion**. On CLEVRText, **Oh-A-DINOv2** reaches **20.3%**, substantially higher than the **12.2%** of **DINOv2**, the **4.2%** of **DINOv3**, the **2.9%** of **CLIP**, and the **1.4%** of **SlotDiffusion**. Interestingly, **DINOv3** performs markedly worse than **DINOv2** on CLEVRText (4.2% vs. 12.2%), suggesting that the newer model is less robust to textured multi-object scenes. These results together with Table 8.3 confirm that SSL features (DINO/CLIP) fail to capture non-geometric attributes, slot-based features collapse under increased scene complexity, and that combining SSL (specifically DINOv2) with object-level VAE representations directly addresses both shortcomings.

VAE Features (%)	CLEVR			CLEVRText		
	Top-10 Precision↑	Weighted Precision↑	Error Rate↓	Top-10 Precision↑	Weighted Precision↑	Error Rate↓
VAE (Ours)	53.1±3.6	29.2±2.7	2.0±1.1	11.7±3.7	4.1±1.3	60.0±3.4
Oh-A-DINOv2 (Ours)	56.4±2.0	49.0±2.2	1.0±0.0	20.3±2.6	11.2±1.4	41.0±1.0
DINOv2	13.0±1.1	13.8±1.2	28.0±1.0	12.2±2.1	8.0±1.3	55.6±0.7

Attrib. Ablation Top-10 Precision (%)	CLEVR				CLEVRText				CLEVR			CLEVRText	
	Shape	Size	Mat.	Col.	Shape	Size	Mat.		$\mathcal{P}_2(\mathcal{X})$	$\mathcal{P}_3(\mathcal{X})$	$\mathcal{P}_3(\mathcal{X})+C$	$\mathcal{P}_2(\mathcal{X})$	$\mathcal{P}_2(\mathcal{X})+C$
VAE (Ours)	93.4±1.2	99.2±0.4	99.6±0.3	98.7±0.4	89.0±0.9	94.4±0.8	43.5±5.3		82.8±0.5	63.3±0.5	53.1±3.6	52.5±0.1	11.7±3.7
Oh-A-DINOv2 (Ours)	94.7±2.0	96.7±2.2	98.6±0.0	96.6±2.6	90.7±2.1	85.3±1.5	45.8±3.9		85.2±2.1	72.8±1.4	56.4±1.6	56.0±0.2	20.3±2.6
DINOv2	95.4±1.1	81.2±1.2	96.1±1.0	40.8±2.1	87.3±3.8	80.3±2.6	24.2±1.9		76.2±3.1	62.8±1.2	13.0±1.1	54.0±0.1	12.2±2.1

Single-Attribute
Multi-Attribute

The Role of the VAE features **Object-level features improve multi-object instance retrieval when combined with DINOv2.** Table 8.7 (bottom) shows that individual attribute prediction improves significantly with the learned object-level representations **VAE** for both CLEVR and CLEVRText. This improves color prediction in **Oh-A-DINO**. However, Table 8.7 (top) reveals that only the Oh-A-DINO combination manages to utilise this color signal to improve multi-attribute retrieval. When comparing **Oh-A-DINO** and **VAE** in CLEVR, even with a smaller performance gap in CLEVR Top-10 precision, our method delivers substantially better weighted precision, indicating that DINO’s scene features help retrieve more consistent objects.

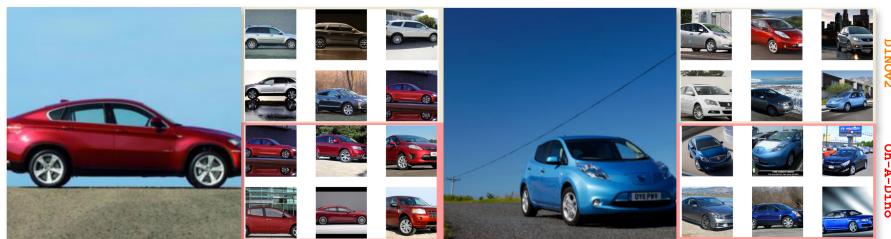
Figure 8.7: Object-level VAE features enhance color and multi-attribute retrieval. While the standalone **VAE** improves fine-grained color retrieval on both CLEVR and CLEVRText, only **Oh-A-DINOv2** effectively integrates this signal with DINOv2 scene-level features. This leads to substantial gains in multi-attribute retrieval, where pure VAE features or SSL features (**DINOv2**) alone fall short.

Augmenting with VAE (%)	CLEVR			CLEVRText		
	Top-10 Precision↑	Weighted Precision↑	Error Rate↓	Top-10 Precision↑	Weighted Precision↑	Error Rate↓
Oh-A-DINOv2 (Ours)	56.4±2.0	49.0±2.2	1.0±0.0	20.3±2.6	11.2±1.4	41.0±1.0
DINOv3-VAE	34.8±2.6	25.2±1.9	10.8±1.8	5.7±1.6	1.0±0.2	82.0±1.8
CLIP-VAE	55.0±2.4	31.8±1.5	1.0±0.0	3.1±1.0	0.9±0.2	86.4±1.2
SlotDiff.-VAE	14.3±1.5	6.8±0.6	30.6±2.0	2.8±0.1	0.8±0.0	87.2±1.7

Model Capacity (%)	CLEVR			CLEVRText		
	Top-10 Precision↑	Weighted Precision↑	Error Rate↓	Top-10 Precision↑	Weighted Precision↑	Error Rate↓
Oh-A-DINOv2 (Ours)	56.4±2.0	49.0±2.2	1.0±0.0	20.3±2.6	11.2±1.4	41.0±1.0
DINOv2-S	13.0±1.1	13.8±1.2	28.0±1.0	12.2±2.1	8.0±1.3	55.6±0.7
DINOv2-L	13.3±1.2	14.8±1.1	24.0±2.0	13.2±1.8	9.0±1.1	50.6±0.9

Augmenting other SSL models with VAE latent features **Augmenting SSL features with VAE latents is effective, but depends on the backbone.** Table 8.8 (left) further compares augmentations of different SSL and slot-based features with our learnt VAE latents. For **DINOv3-VAE**, improvements on CLEVR are moderate (34.8% Top-10 Precision) and largely vanish on CLEVRText. In contrast, **CLIP-VAE** shows strong gains on CLEVR (55.0% Top-10 Precision, comparable to **Oh-A-DINOv2**) but only minor improvements on CLEVRText. Finally, combining **SlotDiffusion** with either **DINOv2** or **VAE** provides negligible benefit, suggesting that slot-based representations are not sufficiently disentangled to compose well with object-level latents. Overall, these results highlight that while VAE augmentation consistently helps SSL features, the degree of improvement depends strongly on the underlying representation, with **DINOv2** benefiting the most.

Main Results (%)	Stanford Cars
	Distance to Ref Colour↓
Oh-A-DINOv2 (Ours)	0.432±0.011
DINOv2	0.512±0.014
CLIP	0.524±0.021



Effect of model capacity **Scaling up SSL model capacity does not resolve the retrieval bottleneck.** Table 8.8(right) compares **DINOv2-S** with its larger variant **DINOv2-L**. While **DINOv2-L** achieves a slight improvement over **DINOv2-S** on both CLEVR (Top-10 Precision 13.3% vs. 13.0%) and CLEVRText (13.2% vs. 12.2%), the gains are marginal and far from closing the gap to **Oh-A-DINOv2** (56.4% on CLEVR, 20.3% on CLEVRText). This suggests that simply increasing model capacity within SSL frameworks does not lead to more disentangled or attribute-sensitive object representations. By contrast, augmenting with object-level VAE features directly addresses the compositional limitations of SSL features, providing consistent improvements irrespective

Figure 8.8: **Ablation on VAE augmentation and model capacity.** (top) Augmenting SSL and slot-based features with object-level VAE latents improves retrieval across methods, with the largest gains observed for **Oh-A-DINOv2**. (bottom) Scaling backbone capacity from **DINOv2-S** to **DINOv2-L** provides only minor improvements.

Figure 8.9: **Our method improves over baselines and leverages DINOv2’s general object understanding.** (left) Table: comparison against baselines on Stanford Cars shows that our method achieves lower distance to the colours of the reference image. (right) Oh-A-Dino can consistently retrieve cars which match orientation and shape while matching colour and material opposed to DINOv2.

of backbone size.

8.8 *Stanford Cars: Real-World Instance Retrieval*

VAE features help retrieve surface level attributes more faithfully for Stanford Cars. Figure 8.9 (left) shows that **Oh-A-DINOv2** achieves a lower distance to the reference colour (**0.432**) compared to both **DINOv2** (0.512) and **CLIP** (0.524). This indicates that **Oh-A-DINOv2** not only captures the global shape and orientation of cars but is also more sensitive to fine-grained attributes such as colour and material, which are crucial for reliable retrieval in real-world settings. The qualitative examples on the right confirm this behaviour: our method consistently retrieves cars that align in orientation and shape while matching the query’s visual appearance, addressing the key limitations of purely SSL-based features.

8.9 *Discussion*

Limitations and Future Work. While we presented an effective method, there are a few limitations that should be considered. Our work only uses a single pre-trained model, DINOv2, to demonstrate the effectiveness of our method. Potentially, other larger pre-trained models like CLIP [Radford et al., 2021], which have shown strong generalisation capabilities could prove effective as well. However we focused on DINOv2 as it has shown emergent object understanding. Our method relies on learning expressive object-level features, which in turn relies on the ability to extract relevant features with pre-trained representations. In this sense, the quality of the pre-trained representations determine how powerful the model that learns the object-level features has to be in order to mitigate the shortcomings in the pre-trained representations. Future work should explore the effectiveness of our method on real-world tasks that require multi-object understanding. For instance, our method could improve the performance of goal-conditioned reinforcement learning agents that rely on finer-grained object understanding to identify similar objects with different functional purposes. Furthermore, there is a need for benchmark suites that test this ability from the perspective of reinforcement learning control.

Conclusion. We presented a method that enhances pre-trained representations through a data-based inductive bias that learns and introduces specialised object-level representations into the pre-trained representations. The data-based inductive bias is able to incorporate object-level features into the pre-trained representations to mitigate its shortcomings with retrieving finer-grained object-level features like

colour and texture. Our method leverages the inherent object understanding of pre-trained models to extract object-level image patches thus enabling us to learn object-level representations. We showed that combining both these types of representations improves the shortcomings of traditional slot-based and DINOv2 representations in finer-grained instace retrieval. As self-supervised models continue to advance, we believe this is a scalable approach for multi-object understanding in deep learning tasks.

9

Conclusion and Outlook

This thesis has explored the role of inductive biases in deep learning models, with a specific focus on leveraging data-based inductive biases to enhance learning in complex environments. By developing a unified framework that distinguishes between these two types of inductive biases, we have shown how they can be effectively combined to improve performance across various domains including reinforcement learning, stance detection, and instance retrieval.

9.1 Key Contributions and Insights

9.1.1 Theoretical Framework for Inductive Biases

We have established a distinction between model-based inductive biases, which are inherent to the architecture, loss function, and regularization techniques and data-based inductive biases, which leverage structure from the data itself. Data-based inductive biases prove particularly valuable when the structure of the data reveals information about the task that would be difficult to capture through model-based inductive biases alone.

9.1.2 Effective Applications of Data-Based Inductive Biases

Across multiple domains, we have demonstrated how data-based inductive biases can be extracted and leveraged:

1. **Cyclophobic Reinforcement Learning:** By defining a latent space through hierarchical views of the environment and by counting state-revisitations on these different hierarchical views, we achieved more efficient exploration in reinforcement learning tasks.
2. **LLM-Generated Synthetic Data for Stance Detection:** We leveraged LLMs as a structure-inducing function to generate synthetic

data that helps improve stance detection in political discussions. The synthetic data created a useful latent structure that allowed for identifying ambiguous samples that were most informative for human labeling, substantially reducing annotation effort and masking out ambiguous samples.

3. **Structured Representations for Reinforcement Learning Exploration:** Using a Gaussian mixture model to cluster pre-trained representations, we defined a latent space over the observations of episodic trajectories which aggregated over time enabled better exploration through more accurate pseudo-counts of visited states. This enabled more efficient exploration in high-dimensional 3-D environments, compared to other state of the art methods.
4. **Enhanced Multi-Object Instance Retrieval:** We showed how pre-trained representations can be augmented with object-centric features learned from image patches using a VAE. This approach mitigated the shortcomings in finer-grained object understanding of existing models by combining general scene understanding with specific object-level features learnt through the latent structure defined by training a VAE on the image patches.

9.1.3 *Practical Insights on Combining Different Types of Biases*

Our work has uncovered several practical insights for effectively combining model-based and data-based inductive biases:

1. The transition saliency problem in 3D environments demonstrated that clustering representations can be more effective than prediction-error-based approaches for density estimation.
2. Pre-trained representations from foundation models provide useful priors but often have blind spots (such as colour invariance in DINOv2) that can be mitigated by learning complementary object-level features.
3. In stance detection, synthetic data from LLMs can align well with real-world data distributions, providing an effective way to enhance models without requiring extensive manual annotation.
4. For model-based inductive biases, we found that architectures like CNNs can learn invariances (e.g., to color) that tabular methods cannot, highlighting the complementary nature of different approaches.

9.2 Outlook for Future Research

From the works proposed in this thesis, several promising directions for future research emerge:

9.2.1 *Combining Data-Based Inductive Biases with Foundation Models*

As foundation models continue to grow in capability and scope, finding effective ways to adapt them to specific tasks without full fine-tuning becomes increasingly important. Our work suggests that data-based inductive biases could provide a lightweight approach to enhancing pre-trained models for downstream tasks. Future research could explore more sophisticated methods for extracting and incorporating such biases.

9.2.2 *Dynamic and Adaptive Inductive Biases*

Most current approaches to inductive biases are static, defined before or during training. Future work could explore adaptive inductive biases that evolve during training based on the data encountered or the performance on specific subtasks. This might lead to more robust and generalizable models.

9.2.3 *Applications to Multimodal Learning*

The increasing focus on multimodal learning presents unique challenges for representation learning. Our findings suggest that different modalities might benefit from different types of inductive biases. Exploring how to leverage data-based and model-based inductive biases in multimodal settings could lead to more effective multimodal models.

9.2.4 *Scaling to Real-World Applications*

Many of our experiments were conducted in controlled environments. An important direction for future work is scaling these approaches to more complex, real-world applications. This includes exploring how data-based inductive biases can be effectively leveraged in noisy, high-dimensional, and dynamic environments.

9.2.5 *Interpretability and Explainability*

As models become more complex, understanding their behavior becomes more challenging. The explicit structure introduced by data-based inductive biases might provide a path toward more interpretable

models. Future research could explore how leveraging these biases can be used to steer models in a more transparent and explainable way.

9.3 *Concluding Remarks*

This thesis has demonstrated that the combination of model-based and data-based inductive biases can lead to more effective learning in a variety of domains. By developing a framework that distinguishes between these types of biases and showing how they can be effectively combined, we have provided both theoretical insights and practical approaches that can guide future research in deep learning.

As the field continues to evolve, different types of inductive biases will likely play an increasingly important role in developing models that can learn effectively from limited data and generalise robustly to new situations

Bibliography

- Abeer ALDayel and Walid Magdy. Stance detection on social media: State of the art and trends. *Information Processing & Management*, 58(4):102597, 2021. ISSN 0306-4573. DOI: <https://doi.org/10.1016/j.ipm.2021.102597>. URL <https://www.sciencedirect.com/science/article/pii/S0306457321000960>.
- André Barreto, Will Dabney, Rémi Munos, Jonathan J. Hunt, Tom Schaul, Hado van Hasselt, and David Silver. Successor Features for Transfer in Reinforcement Learning. *arXiv e-prints*, art. arXiv:1606.05312, June 2016.
- Maike Behrendt, Stefan Sylvius Wagner, and Stefan Harmeling. Supporting online discussions: Integrating ai into the adhocracy+ participation platform to enhance deliberation, 2024. URL <https://arxiv.org/abs/2409.07780>.
- Marc G. Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying Count-Based Exploration and Intrinsic Motivation. *arXiv e-prints*, art. arXiv:1606.01868, June 2016a. DOI: 10.48550/arXiv.1606.01868.
- Marc G. Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying Count-Based Exploration and Intrinsic Motivation. *arXiv e-prints*, art. arXiv:1606.01868, June 2016b.
- Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by Random Network Distillation. *arXiv e-prints*, art. arXiv:1810.12894, October 2018a. DOI: 10.48550/arXiv.1810.12894.
- Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by Random Network Distillation. *arXiv e-prints*, art. arXiv:1810.12894, October 2018b.
- Christopher P. Burgess, Loic Matthey, Nicholas Watters, Rishabh Kabra, Irina Higgins, Matt Botvinick, and Alexander Lerchner. MONet: Unsupervised Scene Decomposition and Representation. *arXiv e-prints*, art. arXiv:1901.11390, January 2019. DOI: 10.48550/arXiv.1901.11390.

Michael Burnham. Stance Detection: A Practical Guide to Classifying Political Beliefs in Text. *arXiv e-prints*, art. arXiv:2305.01723, May 2023. DOI: 10.48550/arXiv.2305.01723.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging Properties in Self-Supervised Vision Transformers. *arXiv e-prints*, art. arXiv:2104.14294, April 2021a. DOI: 10.48550/arXiv.2104.14294.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging Properties in Self-Supervised Vision Transformers. *arXiv e-prints*, art. arXiv:2104.14294, April 2021b. DOI: 10.48550/arXiv.2104.14294.

Tonglin Chen, Yinxuan Huang, Zhimeng Shen, Jinghao Huang, Bin Li, and Xiangyang Xue. Learning Global Object-Centric Representations via Disentangled Slot Attention. *arXiv e-prints*, art. arXiv:2410.18809, October 2024. DOI: 10.48550/arXiv.2410.18809.

Wei Chen, Yu Liu, Weiping Wang, Erwin Bakker, Theodoros Georgiou, Paul Fieguth, Li Liu, and Michael S. Lew. Deep Learning for Instance Retrieval: A Survey. *arXiv e-prints*, art. arXiv:2101.11282, January 2021. DOI: 10.48550/arXiv.2101.11282.

Maxime Chevalier-Boisvert, Lucas Willems, and Suman Pal. Minimalistic gridworld environment for openai gym. <https://github.com/maximecb/gym-minigrid>, 2018.

Alan Darmasaputra Chowanda, Albert Richard Sanyoto, Derwin Suhartono, and Criscentia Jessica Setiadi. Automatic debate text summarization in online debate forum. *Procedia Computer Science*, 116:11–19, 2017. ISSN 1877-0509. DOI: <https://doi.org/10.1016/j.procs.2017.10.003>. URL <https://www.sciencedirect.com/science/article/pii/S1877050917320409>. Discovery and innovation of computer science technology in artificial intelligence era: The 2nd International Conference on Computer Science and Computational Intelligence (ICCSCI 2017).

Iain J. Cruickshank and Lynnette Hui Xian Ng. Prompting and Fine-Tuning Open-Sourced Large Language Models for Stance Classification. *arXiv e-prints*, art. arXiv:2309.13734, September 2023. DOI: 10.48550/arXiv.2309.13734.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Tamar Solorio, editors, *Proceedings of the 2019 Conference of the North American*

- Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. DOI: 10.18653/v1/N19-1423. URL <https://aclanthology.org/N19-1423>.
- Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.
- Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. First return, then explore. *nat*, 590(7847):580–586, February 2021. DOI: 10.1038/s41586-020-03157-9.
- Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymyr Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures. *arXiv e-prints*, art. arXiv:1802.01561, February 2018a. DOI: 10.48550/arXiv.1802.01561.
- Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymyr Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures. *arXiv e-prints*, art. arXiv:1802.01561, February 2018b.
- Ke Fan, Zechen Bai, Tianjun Xiao, Tong He, Max Horn, Yanwei Fu, Francesco Locatello, and Zheng Zhang. Adaptive slot attention: Object discovery with dynamic slot number. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23062–23071, 2024.
- Klaus Greff, Sjoerd van Steenkiste, and Jürgen Schmidhuber. On the Binding Problem in Artificial Neural Networks. *arXiv e-prints*, art. arXiv:2012.05208, December 2020.
- Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. Not what you’ve signed up for: Compromising real-world llm-integrated applications with indirect prompt injection. In *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security, AISec ’23*, page 79–90, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400702600. DOI: 10.1145/3605764.3623985. URL <https://doi.org/10.1145/3605764.3623985>.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised Learning. *arXiv e-prints*, art. arXiv:2006.07733, June 2020. DOI: 10.48550/arXiv.2006.07733.

İlker Gül, Rémi Lebret, and Karl Aberer. Stance Detection on Social Media with Fine-Tuned Large Language Models. *arXiv e-prints*, art. arXiv:2404.12171, April 2024. DOI: 10.48550/arXiv.2404.12171.

Zhaohan Daniel Guo, Shantanu Thakoor, Miruna Pîslar, Bernardo Avila Pires, Florent Altché, Corentin Tallec, Alaa Saade, Daniele Calandriello, Jean-Bastien Grill, Yunhao Tang, Michal Valko, Rémi Munos, Mohammad Gheshlaghi Azar, and Bilal Piot. BYOL-Explore: Exploration by Bootstrapped Prediction. *arXiv e-prints*, art. arXiv:2206.08332, June 2022. DOI: 10.48550/arXiv.2206.08332.

Danijar Hafner, Kuang-Huei Lee, Ian Fischer, and Pieter Abbeel. Deep Hierarchical Planning from Pixels. *arXiv e-prints*, art. arXiv:2206.04114, June 2022. DOI: 10.48550/arXiv.2206.04114.

Momchil Hardalov, Arnav Arora, Preslav Nakov, and Isabelle Augenstein. A survey on stance detection for mis- and disinformation identification. In Marine Carpuat, Marie-Catherine de Marnette, and Ivan Vladimir Meza Ruiz, editors, *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1259–1277, Seattle, United States, July 2022. Association for Computational Linguistics. DOI: 10.18653/v1/2022.findings-naacl.94. URL <https://aclanthology.org/2022.findings-naacl.94>.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Mikael Henaff, Roberta Raileanu, Minqi Jiang, and Tim Rocktäschel. Exploration via Elliptical Episodic Bonuses. *arXiv e-prints*, art. arXiv:2210.05805, October 2022. DOI: 10.48550/arXiv.2210.05805.

Mikael Henaff, Minqi Jiang, and Roberta Raileanu. A Study of Global and Episodic Bonuses for Exploration in Contextual MDPs. *arXiv e-prints*, art. arXiv:2306.03236, June 2023. DOI: 10.48550/arXiv.2306.03236.

- Alex Holub, Pietro Perona, and Michael C Burl. Entropy-based active learning for object recognition. In *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–8. IEEE, 2008.
- Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models. *arXiv e-prints*, art. arXiv:2106.09685, June 2021. DOI: 10.48550/arXiv.2106.09685.
- Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, L  lio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timoth  e Lacroix, and William El Sayed. Mistral 7B. *arXiv e-prints*, art. arXiv:2310.06825, October 2023. DOI: 10.48550/arXiv.2310.06825.
- 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 *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2901–2910, 2017.
- Whie Jung, Jaehoon Yoo, Sungjin Ahn, and Seunghoon Hong. Learning to Compose: Improving Object Centric Learning by Injecting Compositionality. *arXiv e-prints*, art. arXiv:2405.00646, May 2024. DOI: 10.48550/arXiv.2405.00646.
- Laurynas Karazija, Iro Laina, and Christian Rupprecht. Clevrtex: A texture-rich benchmark for unsupervised multi-object segmentation. *arXiv preprint arXiv:2111.10265*, 2021.
- Micha  Kempka, Marek Wydmuch, Grzegorz Runc, Jakub Toczek, and Wojciech Ja kowski. ViZDoom: A Doom-based AI research platform for visual reinforcement learning. In *IEEE Conference on Computational Intelligence and Games*, pages 341–348, Santorini, Greece, Sep 2016. IEEE. DOI: 10.1109/CIG.2016.7860433. The Best Paper Award.
- Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- Diederik P Kingma, Max Welling, et al. An introduction to variational autoencoders. *Foundations and Trends  in Machine Learning*, 12(4): 307–392, 2019.
- Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *2013 IEEE Interna-*

- tional Conference on Computer Vision Workshops*, pages 554–561, 2013. DOI: 10.1109/ICCVW.2013.77.
- Dilek Küçük and Fazli Can. Stance detection: A survey. *ACM Comput. Surv.*, 53(1), feb 2020. ISSN 0360-0300. DOI: 10.1145/3369026. URL <https://doi.org/10.1145/3369026>.
- Kostiantyn Kucher, Carita Paradis, Magnus Sahlgren, and Andreas Kerren. Active learning and visual analytics for stance classification with alva. *ACM Trans. Interact. Intell. Syst.*, 7(3), oct 2017. ISSN 2160-6455. DOI: 10.1145/3132169. URL <https://doi.org/10.1145/3132169>.
- Heinrich Küttler, Nantas Nardelli, Alexander H. Miller, Roberta Raileanu, Marco Selvatici, Edward Grefenstette, and Tim Rocktäschel. The NetHack Learning Environment. *arXiv e-prints*, art. arXiv:2006.13760, June 2020.
- Tom Le Paine, Caglar Gulcehre, Bobak Shahriari, Misha Denil, Matt Hoffman, Hubert Soyer, Richard Tanburn, Steven Kapturowski, Neil Rabinowitz, Duncan Williams, Gabriel Barth-Maron, Ziyu Wang, Nando de Freitas, and Worlds Team. Making Efficient Use of Demonstrations to Solve Hard Exploration Problems. *arXiv e-prints*, art. arXiv:1909.01387, September 2019. DOI: 10.48550/arXiv.1909.01387.
- Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015. DOI: 10.1038/nature14539. URL <https://doi.org/10.1038/nature14539>.
- Chang Li and Dan Goldwasser. Encoding social information with graph convolutional networks for Political perspective detection in news media. In Anna Korhonen, David Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2594–2604, Florence, Italy, July 2019. Association for Computational Linguistics. DOI: 10.18653/v1/P19-1247. URL <https://aclanthology.org/P19-1247>.
- Yingjie Li, Tiberiu Sosea, Aditya Sawant, Ajith Jayaraman Nair, Diana Inkpen, and Cornelia Caragea. P-stance: A large dataset for stance detection in political domain. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2355–2365, Online, August 2021. Association for Computational Linguistics. DOI: 10.18653/v1/2021.findings-acl.208. URL <https://aclanthology.org/2021.findings-acl.208>.
- Zhuoyan Li, Hangxiao Zhu, Zhuoran Lu, and Ming Yin. Synthetic data generation with large language models for text classification:

- Potential and limitations. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. URL <https://openreview.net/forum?id=MmBjKmHIND>.
- Yujian Liu, Xinliang Frederick Zhang, David Wegsman, Nick Beauchamp, and Lu Wang. POLITICS: Pretraining with Same-story Article Comparison for Ideology Prediction and Stance Detection. *arXiv e-prints*, art. arXiv:2205.00619, May 2022. DOI: 10.48550/arXiv.2205.00619.
- Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-Centric Learning with Slot Attention. *arXiv e-prints*, art. arXiv:2006.15055, June 2020. DOI: 10.48550/arXiv.2006.15055.
- Xiao Ma, Shen-Yi Zhao, and Wu-Jun Li. Clustered Reinforcement Learning. *arXiv e-prints*, art. arXiv:1906.02457, June 2019. DOI: 10.48550/arXiv.1906.02457.
- Marlos C. Machado, Sriram Srinivasan, and Michael Bowling. Domain-Independent Optimistic Initialization for Reinforcement Learning. *arXiv e-prints*, art. arXiv:1410.4604, October 2014.
- Ghazaleh Mahmoudi, Babak Behkamkia, and Sauleh Eetemadi. Zero-shot stance detection using contextual data generation with LLMs. In *AAAI-2024 Workshop on Public Sector LLMs: Algorithmic and Sociotechnical Design*, 2024. URL <https://openreview.net/forum?id=n9yozEWDG0>.
- Katerina Margatina, Giorgos Vernikos, Loïc Barrault, and Nikolaos Aletras. Active Learning by Acquiring Contrastive Examples. *arXiv e-prints*, art. arXiv:2109.03764, September 2021. DOI: 10.48550/arXiv.2109.03764.
- Houman Mehrafarin, Sara Rajaei, and Mohammad Taher Pilehvar. On the importance of data size in probing fine-tuned models. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Findings of the Association for Computational Linguistics: ACL 2022*, pages 228–238, Dublin, Ireland, May 2022. Association for Computational Linguistics. DOI: 10.18653/v1/2022.findings-acl.20. URL <https://aclanthology.org/2022.findings-acl.20>.
- Blake Miller, Fridolin Linder, and Walter R. Mebane. Active learning approaches for labeling text: Review and assessment of the performance of active learning approaches. *Political Analysis*, 28(4):532–551, 2020. DOI: 10.1017/pan.2020.4.

Piotr Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andrew J. Ballard, Andrea Banino, Misha Denil, Ross Goroshin, Laurent Sifre, Koray Kavukcuoglu, Dharshan Kumaran, and Raia Hadsell. Learning to Navigate in Complex Environments. *arXiv e-prints*, art. arXiv:1611.03673, November 2016.

Saif M. Mohammad, Parinaz Sobhani, and Svetlana Kiritchenko. Stance and sentiment in tweets. *Special Section of the ACM Transactions on Internet Technology on Argumentation in Social Media*, 17(3), 2017.

Anders Giovanni Møller, Jacob Aarup Dalsgaard, Arianna Pera, and Luca Maria Aiello. Is a prompt and a few samples all you need? Using GPT-4 for data augmentation in low-resource classification tasks. *arXiv e-prints*, art. arXiv:2304.13861, April 2023. DOI: 10.48550/arXiv.2304.13861.

Milton L. Montero, Jeffrey S. Bowers, and Gaurav Malhotra. Successes and Limitations of Object-centric Models at Compositional Generalisation. *arXiv e-prints*, art. arXiv:2412.18743, December 2024. DOI: 10.48550/arXiv.2412.18743.

NLLB Team, Marta R. Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, Anna Sun, Skyler Wang, Guillaume Wenzek, Al Youngblood, Bapi Akula, Loic Barrault, Gabriel Mejia Gonzalez, Prangthip Hansanti, John Hoffman, Semarley Jarrett, Kaushik Ram Sadagopan, Dirk Rowe, Shannon Spruit, Chau Tran, Pierre Andrews, Necip Fazil Ayan, Shruti Bhosale, Sergey Edunov, Angela Fan, Cynthia Gao, Vedanuj Goswami, Francisco Guzmán, Philipp Koehn, Alexandre Mourachko, Christophe Ropers, Safiyyah Saleem, Holger Schwenk, and Jeff Wang. No Language Left Behind: Scaling Human-Centered Machine Translation. *arXiv e-prints*, art. arXiv:2207.04672, July 2022. DOI: 10.48550/arXiv.2207.04672.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning Robust Visual Features without Supervision. *arXiv e-prints*, art. arXiv:2304.07193, April 2023a. DOI: 10.48550/arXiv.2304.07193.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas,

- Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning Robust Visual Features without Supervision. *arXiv e-prints*, art. arXiv:2304.07193, April 2023b. DOI: 10.48550/arXiv.2304.07193.
- Georg Ostrovski, Marc G. Bellemare, Aaron van den Oord, and Remi Munos. Count-Based Exploration with Neural Density Models. *arXiv e-prints*, art. arXiv:1703.01310, March 2017a. DOI: 10.48550/arXiv.1703.01310.
- Georg Ostrovski, Marc G. Bellemare, Aaron van den Oord, and Remi Munos. Count-Based Exploration with Neural Density Models. *arXiv e-prints*, art. arXiv:1703.01310, March 2017b.
- Simone Parisi, Victoria Dean, Deepak Pathak, and Abhinav Gupta. Interesting Object, Curious Agent: Learning Task-Agnostic Exploration. *arXiv e-prints*, art. arXiv:2111.13119, November 2021.
- Simone Parisi, Victoria Dean, Deepak Pathak, and Abhinav Gupta. Interesting object, curious agent: Learning task-agnostic exploration. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL <https://openreview.net/forum?id=knKJgksd7kA>.
- Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven Exploration by Self-supervised Prediction. *arXiv e-prints*, art. arXiv:1705.05363, May 2017a. DOI: 10.48550/arXiv.1705.05363.
- Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven Exploration by Self-supervised Prediction. *arXiv e-prints*, art. arXiv:1705.05363, May 2017b.
- Xavi Puig, Eric Undersander, Andrew Szot, Mikael Dallaire Cote, Ruslan Partsey, Jimmy Yang, Ruta Desai, Alexander William Clegg, Michal Hlavac, Tiffany Min, Theo Gervet, Vladimir Vondrus, Vincent-Pierre Berges, John Turner, Oleksandr Maksymets, Zsolt Kira, Mrinal Kalakrishnan, Jitendra Malik, Devendra Singh Chaplot, Unnat Jain, Dhruv Batra, Akshara Rai, and Roozbeh Mottaghi. Habitat 3.0: A co-habitat for humans, avatars and robots, 2023.
- Adrià Puigdomènech Badia, Pablo Sprechmann, Alex Vitvitskyi, Daniel Guo, Bilal Piot, Steven Kapturowski, Olivier Tieleman, Martín Arjovsky, Alexander Pritzel, Andrew Bolt, and Charles Blundell. Never Give Up: Learning Directed Exploration Strategies. *arXiv e-prints*, art. arXiv:2002.06038, February 2020. DOI: 10.48550/arXiv.2002.06038.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Aspell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.

Roberta Raileanu and Tim Rocktäschel. RIDE: Rewarding Impact-Driven Exploration for Procedurally-Generated Environments. *arXiv e-prints*, art. arXiv:2002.12292, February 2020a. DOI: 10.48550/arXiv.2002.12292.

Roberta Raileanu and Tim Rocktäschel. RIDE: Rewarding Impact-Driven Exploration for Procedurally-Generated Environments. *arXiv e-prints*, art. arXiv:2002.12292, February 2020b.

Julia Romberg and Tobias Escher. Automated topic categorisation of citizens’ contributions: Reducing manual labelling efforts through active learning. In *Electronic Government: 21st IFIP WG 8.5 International Conference, EGOV 2022, Linköping, Sweden, September 6–8, 2022, Proceedings*, page 369–385, Berlin, Heidelberg, 2022. Springer-Verlag. ISBN 978-3-031-15085-2. DOI: 10.1007/978-3-031-15086-9_24. URL https://doi.org/10.1007/978-3-031-15086-9_24.

Julia Romberg and Tobias Escher. Making sense of citizens’ input through artificial intelligence: A review of methods for computational text analysis to support the evaluation of contributions in public participation. *Digit. Gov.: Res. Pract.*, jun 2023. DOI: 10.1145/3603254. URL <https://doi.org/10.1145/3603254>. Just Accepted.

Alaa Saade, Steven Kapturowski, Daniele Calandriello, Charles Blundell, Pablo Sprechmann, Leopoldo Sarra, Oliver Groth, Michal Valko, and Bilal Piot. Unlocking the Power of Representations in Long-term Novelty-based Exploration. *arXiv e-prints*, art. arXiv:2305.01521, May 2023. DOI: 10.48550/arXiv.2305.01521.

Mikayel Samvelyan, Robert Kirk, Vitaly Kurin, Jack Parker-Holder, Mingqi Jiang, Eric Hambro, Fabio Petroni, Heinrich Küttler, Edward Grefenstette, and Tim Rocktäschel. MiniHack the Planet: A Sandbox for Open-Ended Reinforcement Learning Research. *arXiv e-prints*, art. arXiv:2109.13202, September 2021.

Mikayel Samvelyan, Robert Kirk, Vitaly Kurin, Jack Parker-Holder, Mingqi Jiang, Eric Hambro, Fabio Petroni, Heinrich Küttler, Edward Grefenstette, and Tim Rocktäschel. Minihack the planet: A sandbox for open-ended reinforcement learning research. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021. URL <https://openreview.net/forum?id=skFwlyefkWJ>.

Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A Platform for Embodied AI Research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1312–1320, Lille, France, 07–09 Jul 2015. PMLR. URL <https://proceedings.mlr.press/v37/schaul15.html>.

Klaus Schmidt, Andreas Niekler, Cathleen Kantner, and Manuel Burghardt. Classifying speech acts in political communication: A transformer-based approach with weak supervision and active learning. In *2023 18th Conference on Computer Science and Intelligence Systems (FedCSIS)*, pages 739–748, 2023. DOI: 10.15439/2023F3485.

Bernhard Schölkopf, Francesco Locatello, Stefan Bauer, Nan Rosemary Ke, Nal Kalchbrenner, Anirudh Goyal, and Yoshua Bengio. Towards Causal Representation Learning. *arXiv e-prints*, art. arXiv:2102.11107, February 2021. DOI: 10.48550/arXiv.2102.11107.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *arXiv e-prints*, art. arXiv:1707.06347, July 2017. DOI: 10.48550/arXiv.1707.06347.

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*, 2023. URL https://openreview.net/forum?id=b9tUk-f_aG.

H. S. Seung, M. Oppen, and H. Sompolinsky. Query by committee. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, page 287–294, New York, NY, USA, 1992. Association for Computing Machinery. ISBN 089791497X. DOI: 10.1145/130385.130417. URL <https://doi.org/10.1145/130385.130417>.

Alex Sherstinsky. Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *Physica D: Nonlinear Phenomena*, 404:132306, 2020.

Oriane Siméoni, Gilles Puy, Huy V Vo, Simon Roburin, Spyros Gidaris, Andrei Bursuc, Patrick Pérez, Renaud Marlet, and Jean Ponce. Localizing objects with self-supervised transformers and no labels. *arXiv preprint arXiv:2109.14279*, 2021.

Oriane Siméoni, Huy V Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, et al. Dinov3. *arXiv preprint arXiv:2508.10104*, 2025.

Gautam Singh, Fei Deng, and Sungjin Ahn. Illiterate DALL-E Learns to Compose. *arXiv e-prints*, art. arXiv:2110.11405, October 2021. DOI: 10.48550/arXiv.2110.11405.

Gautam Singh, Yi-Fu Wu, and Sungjin Ahn. Simple Unsupervised Object-Centric Learning for Complex and Naturalistic Videos. *arXiv e-prints*, art. arXiv:2205.14065, May 2022. DOI: 10.48550/arXiv.2205.14065.

Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J. Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, Anton Clarkson, Mingfei Yan, Brian Budge, Yajie Yan, Xiaqing Pan, June Yon, Yuyang Zou, Kimberly Leon, Nigel Carter, Jesus Briales, Tyler Gillingham, Elias Mueggler, Luis Pesqueira, Manolis Savva, Dhruv Batra, Hauke M. Strasdat, Renzo De Nardi, Michael Goesele, Steven Lovegrove, and Richard Newcombe. The Replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019.

Alexander L. Strehl and Michael L. Littman. An analysis of model-based interval estimation for markov decision processes. *Journal of Computer and System Sciences*, 74(8):1309–1331, 2008a. ISSN 0022-0000. DOI: <https://doi.org/10.1016/j.jcss.2007.08.009>. URL <https://www.sciencedirect.com/science/article/pii/S0022000008000767>. Learning Theory 2005.

Alexander L. Strehl and Michael L. Littman. An analysis of model-based interval estimation for markov decision processes. *Journal of Computer and System Sciences*, 74(8):1309–1331, 2008b. ISSN 0022-0000. DOI: <https://doi.org/10.1016/j.jcss.2007.08.009>. URL <https://www.sciencedirect.com/science/article/pii/S0022000008000767>. Learning Theory 2005.

Andrew Szot, Alex Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Chaplot, Oleksandr Maksymets, Aaron Gokaslan, Vladimir Vondrus, Sameer Dharur, Franziska Meier, Wojciech Galuba, Angel Chang, Zsolt Kira, Vladlen Koltun, Jitendra Malik, Manolis Savva, and Dhruv Batra.

- Habitat 2.0: Training home assistants to rearrange their habitat. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.
- Andrew Szot, Max Schwarzer, Harsh Agrawal, Bogdan Mazouze, Walter Talbott, Katherine Metcalf, Natalie Mackraz, Devon Hjelm, and Alexander Toshev. Large Language Models as Generalizable Policies for Embodied Tasks. *arXiv e-prints*, art. arXiv:2310.17722, October 2023. DOI: 10.48550/arXiv.2310.17722.
- Andrew Szot, Bogdan Mazouze, Omar Attia, Aleksei Timofeev, Harsh Agrawal, Devon Hjelm, Zhe Gan, Zsolt Kira, and Alexander Toshev. From multimodal llms to generalist embodied agents: Methods and lessons. *arXiv e-prints*, pages arXiv–2412, 2024.
- Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. #Exploration: A Study of Count-Based Exploration for Deep Reinforcement Learning. *arXiv e-prints*, art. arXiv:1611.04717, November 2016. DOI: 10.48550/arXiv.1611.04717.
- Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023. DOI: <https://doi.org/10.48550/arXiv.2302.13971>.
- Tobias Uelwer, Jan Robine, Stefan Sylvius Wagner, Marc Höftmann, Eric Upschulte, Sebastian Konietzny, Maike Behrendt, and Stefan Harmeling. A Survey on Self-Supervised Representation Learning. *arXiv e-prints*, art. arXiv:2308.11455, August 2023. DOI: 10.48550/arXiv.2308.11455.
- Jannis Vamvas and Rico Sennrich. X-Stance: A multilingual multi-target dataset for stance detection. In *Proceedings of the 5th Swiss Text Analytics Conference (SwissText) & 16th Conference on Natural Language Processing (KONVENS)*, Zurich, Switzerland, jun 2020. URL <http://ceur-ws.org/Vol-2624/paper9.pdf>.
- Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008. URL <http://jmlr.org/papers/v9/vandermaaten08a.html>.
- Sjoerd van Steenkiste, Klaus Greff, and Jürgen Schmidhuber. A Perspective on Objects and Systematic Generalization in Model-Based RL. *arXiv e-prints*, art. arXiv:1906.01035, June 2019a.
- Sjoerd van Steenkiste, Klaus Greff, and Jürgen Schmidhuber. A Perspective on Objects and Systematic Generalization in Model-

Based RL. *arXiv e-prints*, art. arXiv:1906.01035, June 2019b. DOI: 10.48550/arXiv.1906.01035.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017a. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017b.

Veniamin Veselovsky, Manoel Horta Ribeiro, Akhil Arora, Martin Josifoski, Ashton Anderson, and Robert West. Generating faithful synthetic data with large language models: A case study in computational social science. *ArXiv*, abs/2305.15041, 2023. URL <https://api.semanticscholar.org/CorpusID:258866005>.

Stefan Sylvius Wagner and Stefan Harmeling. Just cluster it: an approach for exploration in high-dimensions using clustering and pre-trained representations. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Stefan Sylvius Wagner, Peter Arndt, Jan Robine, and Stefan Harmeling. Cyclophobic Reinforcement Learning. *arXiv e-prints*, art. arXiv:2308.15911, August 2023. DOI: 10.48550/arXiv.2308.15911.

Xudong Wang, Rohit Girdhar, Stella X Yu, and Ishan Misra. Cut and learn for unsupervised object detection and instance segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3124–3134, 2023.

Yanbo Wang, Letao Liu, and Justin Dauwels. Slot-VAE: Object-Centric Scene Generation with Slot Attention. *arXiv e-prints*, art. arXiv:2306.06997, June 2023. DOI: 10.48550/arXiv.2306.06997.

Ziyi Wu, Jingyu Hu, Wuyue Lu, Igor Gilitschenski, and Animesh Garg. SlotDiffusion: Object-Centric Generative Modeling with Diffusion Models. *arXiv e-prints*, art. arXiv:2305.11281, May 2023. DOI: 10.48550/arXiv.2305.11281.

Junyi Zhang, Charles Herrmann, Junhwa Hur, Luisa Polania Cabrera, Varun Jampani, Deqing Sun, and Ming-Hsuan Yang. A tale of two

- features: Stable diffusion complements dino for zero-shot semantic correspondence. *Advances in Neural Information Processing Systems*, 36: 45533–45547, 2023.
- Tianjun Zhang, Huazhe Xu, Xiaolong Wang, Yi Wu, Kurt Keutzer, Joseph E Gonzalez, and Yuandong Tian. Noveld: A simple yet effective exploration criterion. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 25217–25230. Curran Associates, Inc., 2021a. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/d428d070622e0f4363fcea11f4a3576-Paper.pdf.
- Tianjun Zhang, Huazhe Xu, Xiaolong Wang, Yi Wu, Kurt Keutzer, Joseph E. Gonzalez, and Yuandong Tian. Noveld: A simple yet effective exploration criterion. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021b. URL <https://openreview.net/forum?id=CYUzpn0kFJp>.
- Wenqian Zhang, Shangbin Feng, Zilong Chen, Zhenyu Lei, Jundong Li, and Minnan Luo. KCD: Knowledge Walks and Textual Cues Enhanced Political Perspective Detection in News Media. *arXiv e-prints*, art. arXiv:2204.04046, April 2022. DOI: 10.48550/arXiv.2204.04046.
- Ying Zheng, Lei Yao, Yuejiao Su, Yi Zhang, Yi Wang, Sicheng Zhao, Yiyi Zhang, and Lap-Pui Chau. A Survey of Embodied Learning for Object-Centric Robotic Manipulation. *arXiv e-prints*, art. arXiv:2408.11537, August 2024. DOI: 10.48550/arXiv.2408.11537.
- Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. DINO-WM: World Models on Pre-trained Visual Features enable Zero-shot Planning. *arXiv e-prints*, art. arXiv:2411.04983, November 2024. DOI: 10.48550/arXiv.2411.04983.
- Sijie Zhu, Linjie Yang, Chen Chen, Mubarak Shah, Xiaohui Shen, and Heng Wang. R2former: Unified retrieval and reranking transformer for place recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19370–19380, 2023.
- Marc Ziegele, Timo Breiner, and Oliver Quiring. What Creates Interactivity in Online News Discussions? An Exploratory Analysis of Discussion Factors in User Comments on News Items. *Journal of Communication*, 64(6):1111–1138, 10 2014. ISSN 0021-9916. DOI: 10.1111/jcom.12123. URL <https://doi.org/10.1111/jcom.12123>.

Caleb Ziems, William Held, Omar Shaikh, Jiaao Chen, Zhehao Zhang, and Diyi Yang. Can Large Language Models Transform Computational Social Science? *arXiv e-prints*, art. arXiv:2305.03514, April 2023. DOI: 10.48550/arXiv.2305.03514.

Appendices

A Cyclophobic Reinforcement Learning

A.1 Preliminary results with PPO

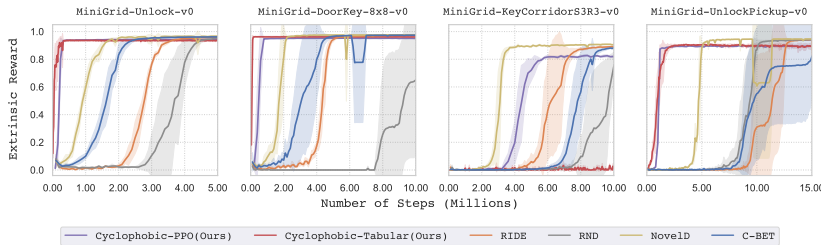


Figure 1: Using PPO together with only the cyclophobic intrinsic reward also converges far quicker than C-BET. Moreover, we are able to converge in KeyCorridorS3R3 confirming our theory in Section 5.7.1.

We train a PPO agent using only the cyclophobic intrinsic reward. Figure 1 shows that for the given environments the PPO agent also explores more efficiently than C-BET. Furthermore, our theory about observation complexity from 5.7.1 is supported as the PPO agent is able to solve the “KeyCorridor” environment. Note that these are preliminary results. In future work we seek to implement the hierarchical state representations for the neural agent.

A.2 Environment Details

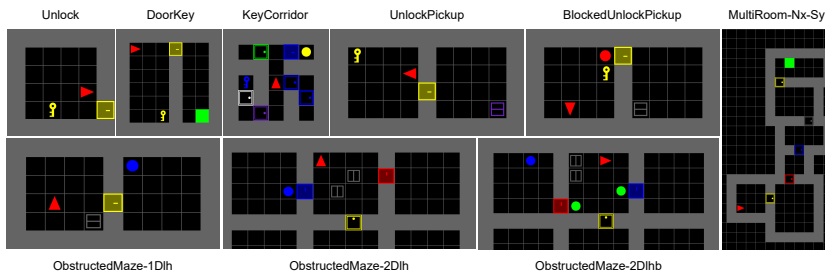


Figure 2: An overview of the MiniGrid environments used in this paper.

MiniGrid Experiments. We test on some of the same environments as [Parisi et al., 2021]. In the following we describe the environments as [Parisi et al., 2021] and add our own comments where relevant.

- **Unlock:** pick up the key and unlock the door. (288 steps per episode)
- **DoorKey-8x8:** pick up the key, unlock the door, and go to green goal (640 steps per episode)
- **KeyCorridorS3R3:** pick up the key, unlock the door, and pick up the ball (only the door before the ball is locked) (270 steps per episode).

- **UnlockPickup**: pick up the key, unlock the door, and open the box (288 steps per episode).
- **BlockedUnlockPickup**: pick up the ball in front of the door, drop it somewhere else, pick up the key, unlock the door, and open the box (576 steps per episode).
- **ObstructedMaze-1Dlh**: open the box to reveal the key, pick it up, unlock the door, and pick up the ball (288 steps per episode).
- **ObstructedMaze-2Dlh**: same as above, but with two doors to unlock (576 steps per episode).
- **ObstructedMaze-2Dlhb**: same as above, but with two balls in front of the doors (like in BlockedUnblockPickup) (576 steps per episode).
- **MultiRoom-N6**: navigate through six rooms of maximum size ten and go to the green goal (all doors are already unlocked) (120 steps per episode).
- **MultiRoom-N12-S10**: navigate through twelve rooms of maximum size ten and go to the green goal (all doors are already unlocked) (120 steps per episode).

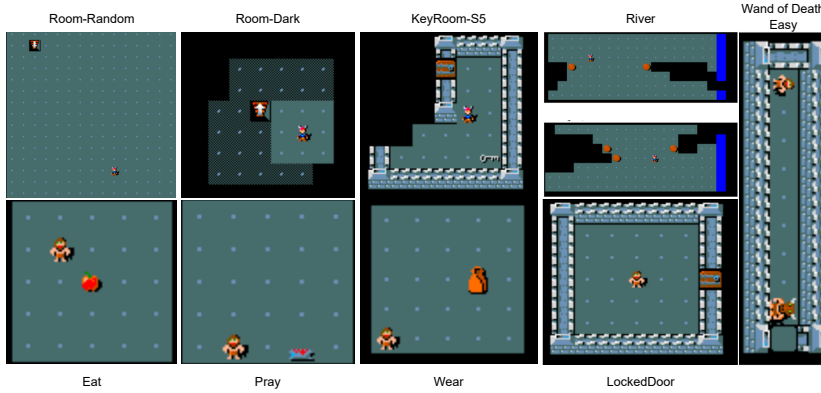


Figure 3: An overview of the Minihack environments tested in this paper. The top row shows the *Navigation* tasks, while the bottom row shows the *Skill* tasks.

MiniHack Experiments. We perform experiments inspired by the baselines from [Samvelyan et al., 2021]. Here, [Samvelyan et al., 2021] define a set of navigation and skill acquisition tasks. The navigation tasks range from requiring the agent to navigate an empty room or maze, up to navigating environments with monsters and hazards. The skill acquisition tasks require the agent to perform a certain set of skills to be able to solve the environments.

Navigation tasks. In the following we describe the environments used from the navigation task.

- **Room-Random-15x15:** This task is set in a single square room, where the goal is to reach the staircase down. In the random version of this task, the start and goal positions are randomized.
- **Room-Dark-15x15:** Same as Room-Random, but the agent only sees parts of the environment it has already discovered.
- **KeyRoom-S5:** These tasks require an agent to pick up a key, navigate to a door, and use the key to unlock the door, reaching the staircase down within the locked room. The locations of the door, key and starting position are randomized.
- **River:** The river environment requires the agent to cross a river using boulders. When pushed into the water, they create a dry land to walk on, allowing the agent to cross it.

Skill tasks. The nature of commands in NetHack requires the agent to perform a sequence of actions so that the initial action, which is meant for interaction with an object, has an effect. The exact sequence of subsequent can be inferred by the in-game message bar prompts. Overall, in each of the following tasks the agent has to perform a sequence of actions that allow the agent to perform the required skill.

- **Eat:** The agent has to navigate and eat an apple. Doing so requires a sequence of actions and confirming a prompt to eat the apple.
- **Pray:** The agent has to pray on an altar. This is more complex than eating an apple as praying requires the agent to confirm more prompts.
- **Wear:** The agent has to wear armor. This is a more difficult task than the previous tasks.
- **LockedDoor-Fixed:** The agent has to kick a locked door.
- **Wand of Death-Easy:** The agent starts with a wand of death in its inventory and has to zap it towards a sleeping monster. That is, the agent needs to equip the wand of death walk to the monster and zap it.

In all of these environments the positions of the agent and of the object that has to be interacted with are randomised. Furthermore, the complexity is also increased by the confirmation prompts the agent has to answer in order to perform a certain action.

A.3 Hyperparameters

Common hyperparameters. In the following we present a table with all hyperparameters used for training the environments. In general, our method has very few hyperparameters. In this case the only really tunable hyperparameters are the step size η , the random action selection coefficient ϵ for epsilon-greedy exploration and the discount γ :

η	γ	Num. views
0.2	0.99	5

In our case we determined experimentally that 5 views is a good number of views for training an agent.

Epsilon-greedy parameter ϵ . While learning with the cyclophobic intrinsic reward we still perform random action from time to time by having an epsilon-greedy action selection. That is, the agent samples a value n from a uniform distribution and selects a random action if this $n < \epsilon$. These are the settings for ϵ for the respective environments:

Environments	ϵ
Unlock	0.1
DoorKey	0.1
KeyCorridor	0.1
UnlockPickup	0.3
MultiRoom-N6	0.1
MultiRoom-N12-S10	0.1
BlockedUnlockPickup	0.3
ObstructedMaze-1Dlh	0.3
ObstructedMaze-2Dlh	0.1
ObstructedMaze-2Dlhb	0.3
All MiniHack	0.3

Intrinsic reward tradeoff ρ . The intrinsic reward tradeoff ρ has the effect that the extrinsic reward gets propagated more when the agent encounters it, which is necessary for more complex environments. ρ is a hyperparameter that can be defined for each environment for best performance.

Environments	ρ
Unlock	1.0
DoorKey	1.0
KeyCorridor	1.0
UnlockPickup	2.0
MultiRoom-N6	2.0
MultiRoom-N12-S10	2.0
BlockedUnlockPickup	5.0
ObstructedMaze-1Dlh	2.0
ObstructedMaze-2Dlh	5.0
ObstructedMaze-2Dlhb	5.0
River	5.0
WOD-Easy	5.0
Eat	2.0
Pray	2.0
Rest of MiniHack	2.0

PPO Hyperparameters.

GAE- λ	0.95
γ	0.99
Batch size	256
Learning rate	0.001
Entropy coefficient	0.01
Value loss coefficient	0.5
Max Grad Normt	0.5
Clipping- ϵ	0.2
RMSProp- ϵ	1e-8
RMSProp- α	0.99

B Just Cluster It

B.1 Clustering DINO vs Random Features

We compare the clusters from DINO (magenta) and random features (cyan) in Figure 4. We can see that the episodic clusters from DINO and random features are fundamentally different. Overall, the DINO cluster centers are much less granular compared to the random features. This can be seen by the fact that the cluster mean images for the random features are free from distortions. Moreover, some cluster centers for the random features are very similar to each other, which is harder to find for the DINO features. From the counts we can see that random features still are able to aggregate counts, however very similar observations may not be aggregated as well as with the DINO clusters. This can be seen by the fact that some counts are unusually high for the random features

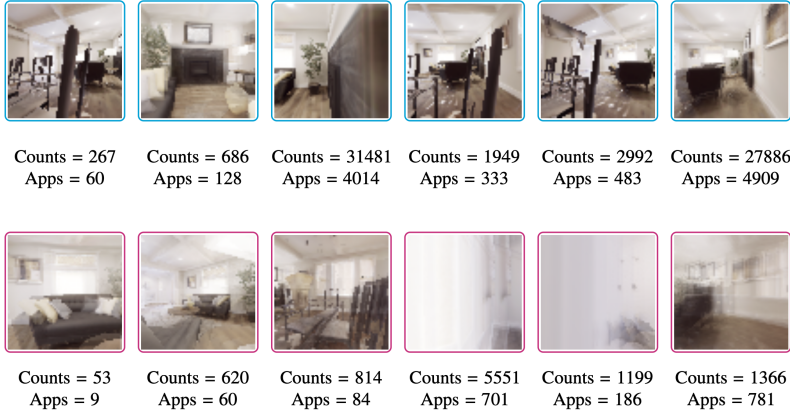


Figure 4: **Comparison of DINO clusters and Random clusters:** Observations from the Habitat environment visualized as samples from the cluster table for DINO (magenta) and random features (cyan). DINO clusters aggregate more dissimilar but semantically related observations, whereas random clusters are more granular and distortion-free. Random features still form aggregated clusters, but some counts are unusually high, suggesting less reliable matching of similar observations.

indicating that some episodic clusters may be assigned erroneously to the same global cluster center. We were not able to find such high counts for the DINO features. Overall, we see that episodic clustering together with global clustering is effective for both DINO and random features, since episodic cluster centers are matched multiple times throughout learning to the global cluster centers, as can be seen by the number of appearances “Apps” in Figure 4.

B.2 Environments

VizDoom Environment Details. We use the “MyWayHome” environment from VizDoom [Kempka et al., 2016] in both its sparse reward configurations “Sparse” and “VerySparse”. In this environment the agent has to reach a piece of armor located in the farthest accessible room of the environment. The configurations differ in the starting position of the agent, one being further away from the goal than the other. In “Sparse”, 270 steps away from goal and in “VerySparse”, 340 steps away from goal. The agent has an action space of *Wait*, *Left*, *Right*, *Forward* and an episode has a length of 2100 steps. We use a standard frame skip of 4. The agent receives a reward of +1 when it reaches the armor and a -0.0001 penalty for every timestep.

Hyperparameters-VizDoom. For VizDoom, we show the hyperparameters for PPO in Table 1 and the hyperparameters for the episodic clustering in Table 2. Additionally, we also show the hyperparameters for RECODE in Table 3. We use the same hyperparameters for both “Sparse” and “VerySparse” configurations. Note that in Table 3 we tried different sizes for the RECODE memory as we found this had the most predictable effect on performance, however for VerySparse even with different memory sizes we were not able to get RECODE to work.

Hyperparameter	Value
Learning rate	0.0001
Batch size	256
Epochs	4
Gamma	0.99
Clip epsilon	0.1
Entropy coefficient	0.005
Value function coefficient	0.5
Workers	32
Recurrence	64
Intrinsic Reward Scaling	0.1

Table 1: PPO Hyperparameters for VizDoom

Parameter	Value
Cosine Similarity Threshold κ (DINO)	0.8
Cosine Similarity Threshold κ (Random)	0.8
Number of Clusters M	250

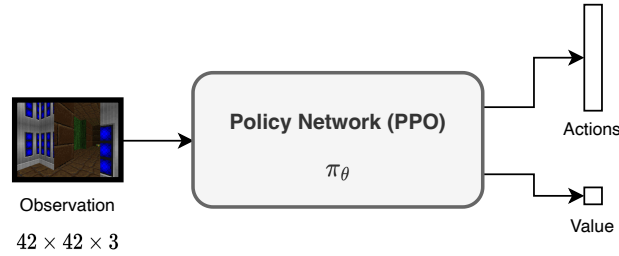
Table 2: Cosine Similarity and Episodic Clustering Parameters

Parameter	Value
RECODE memory size	$\{100, \mathbf{700}, 1 \times 10^4\}$
RECODE discount γ	0.999
RECODE insertion probability η	0.05
RECODE relative tolerance κ	0.2
RECODE reward constant c	0.01
RECODE decay rate τ	0.9999
RECODE neighbors k	20

Table 3: RECODE Parameters VizDoom - Bold value is used for experiments

Habitat Environment Details. **Habitat-Replica** [Savva et al., 2019, Szot et al., 2021, Puig et al., 2023] is a framework for exploration in real world 3-D scenes. The Habitat-Lab API is made to be used with different datasets. We use the Replica dataset [Straub et al., 2019] as in Parisi et al. [2021], where we train on the “Apartment-o” environment, which is the biggest room in the Replica dataset. The agent has an action space of *Forward* by 0.25 coordinate points, *Left* 10° and *Right* 10° and the episode has a length of 500 steps. The agent is rewarded with intrinsic reward only, where the count of visited locations is the performance metric.

B.3 Implementation Details



Observation encoding and network architecture. As policy network we take the Proximal Policy Optimization (PPO) [Schulman et al., 2017] implementation from `lcs-willems` and modify it to our needs. As shown in Figure 5, we downscale the observations to $42 \times 42 \times 3$ and feed the downscaled observations to the following network architecture,

Figure 5: **High level overview of policy network:** The policy network receives the observations and outputs the logits for the action probabilities and the value function.

Listing 1: PPO Network Architecture

```
ACModel(
    (image_conv): Sequential(
      (0): Conv2d(3, 32, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
      (1): ELU(alpha=1.0)
      (2): Conv2d(32, 32, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
      (3): ELU(alpha=1.0)
      (4): Conv2d(32, 32, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
      (5): ELU(alpha=1.0)
      (6): Conv2d(32, 32, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
      (7): ELU(alpha=1.0)
    )
    (memory_rnn1): LSTMCell(288, 256)
    (memory_rnn2): LSTMCell(288, 256)
    (actor): Sequential(
      (0): Linear(in_features=256, out_features=64, bias=True)
```

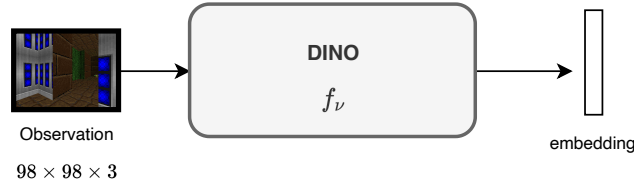
```

        (1): Tanh()
        (2): Linear(in_features=64, out_features=4, bias=True)
    )
    (critic): Sequential(
      (0): Linear(in_features=256, out_features=64, bias=True)
      (1): Tanh()
      (2): Linear(in_features=64, out_features=1, bias=True)
    )
  )
)

```

where the “out_features” for the actor and critic are the number of actions and 1, respectively. Since we have partial observability in our 3-D environments, we additionally use two LSTM cells after the convolutional encoder.

B.4 DINO Setup



For the pre-trained DINO representations we use the “small” version of the “dino-v2” [Oquab et al., 2023a] model. As shown in Figure 6, the model receives an input of size $98 \times 98 \times 3$ since it splits the observation into 14×14 patches. After passing the observations through the model, we take the CLS token of the DINO model as feature vector which is 384-dimensional. For the model size ablation, we use the “large” model instead which has a 1024-dimensional feature vector with $196 \times 196 \times 3$ sized observations as input.

Figure 6: **High level overview of how we use DINO:** We use the pre-trained DINO model to extract embeddings from the observations. We then use the cosine similarity to cluster the embeddings.

Random Features Setup. For the random features we use the target network in Random Network Distillation which is analogous to Listing 1 without the LSTM cells. After the convolutional encoder we simply output a 384-dimensional feature vector. As a consequence, the embeddings, i.e., the cluster means are also 384-dimensional.

Other Libraries. For the Gaussian mixture model we simply use the implementation from [scikit-learn](#).

C The Power Of LLM-generated Synthetic Data For Online Political Discussions

C.1 Synthetic Data

Analysing the synthetic data distribution. As described in Section 6.7.2, we analyse the distribution of the synthetic data in comparison to the real world data. In the following tables we provide samples of the interquartile ranges of the entropy distribution. Tables C.1 and C.1 contain the same samples in English and German respectively. Table C.1 contain comments from the real data distribution.

We see that that the synthetic data samples are longer than the real data samples and contain more reserved language and practically no emotional language. In comparison, the real data samples are shorter and concise with some samples in the lower entropy range containing more emotional language or little information. This however is common for online political discussions. In any case, we see that the underlying notions behind the comments are similar in both the synthetic and real data samples.

Distribution Range	Comment
Minimum	muss die integrationsvereinbarungen noch studieren
0% – 25%	Wer sich anpasst und korrekt verhält darf auch hier bleiben.
25% – 50%	Integrationsvereinbarungen nicht generell verhängen, sondern individuell anordnen, wenn die Person auch nach längerer Anwesenheit in der Schweiz Integrationsdefizite hat oder bei neu zugezogenen Personen, wenn mit Anpassungsschwierigkeiten zu rechnen ist. Keine Standardformulare, keine Automatismen.
50% – 75%	Ich zweifle an der Umsetzbarkeit solcher Methoden, eines "Vertrags". Es ist anstrengenswert, dass ein guter Integrationsablauf statt finden kann. Allerdings ist dafür immer eine Gesellschaft notwendig, die Personen aufnehmen und integrieren will, wie auch Personen die diese Möglichkeit wahrnehmen können und wollen.
75% – 100%	Ich finde die Idee Integrationsmassnahmen verbindlich zu machen sinnvoll. Sprachkurse sind ein ideales Instrument dafür. Es darf aber nicht sein, dass gewisse Forderungen zur Voraussetzung werden für die Aufenthaltserlaubnis. Nicht jeder Mensch lernt gleich schnell eine Sprache oder kann sich gleich schnell integrieren. Missbräuche und Diskriminierungen können auftreten. Dies muss verhindert werden. Ich wäre eher für einen obligatorischen kostenlosen Sprachkurs.
Maximum	Heikler Punkt, im Prinzip JA für Alle, nur - auch gleiche Rechte/Pflichten für alle - warum sollen die EU-EFTA-Bürger keine solchen Integrationsvereinbarungen abschliessen. Der bildungsferne Rumäne fährt besser als der hochausgebildete Chinese/US-Amerikaner/IT-Spezialist aus Indien. Die ganze Einwanderung muss im Sinne der MEI umgesetzt werden.

Table 6: Corresponding German comments sampled from the interquartile ranges of the entropy distribution of the **real** data.

Translated Data Samples We show the translated questions used for synthetic data generation in Table 7 and some samples of generated comments in 9. We see that the questions are translated correctly and synthetic data can be generated for both favor and against stances.

Question in German	Question in English
Sollen sich die Versicherten stärker an den Gesundheitskosten beteiligen (z.B. Erhöhung der Mindestfranchise)	Should insured persons contribute more to health costs (e.g. increase in the minimum deductible)?
Befürworten Sie ein generelles Werbeverbot für Alkohol und Tabak?	Do you support a general ban on advertising alcohol and tobacco?
Soll eine Impfpflicht für Kinder gemäss dem schweizerischen Impfplan eingeführt werden?	Should compulsory vaccination of children be introduced in accordance with the Swiss vaccination schedule?
Soll die Aufenthaltserlaubnis für Migrant/innen aus Nicht-EU/EFTA-Staaten schweizweit an die Erfüllung verbindlicher Integrationsvereinbarungen geknüpft werden?	Should the residence permit for migrants from non-EU/EFTA countries be linked to the fulfilment of binding integration agreements throughout Switzerland?
Soll der Bund erneuerbare Energien stärker fördern?	Should the federal government promote renewable energy more?
Befürworten Sie eine strengere Kontrolle der Lohnleichheit von Frauen und Männern?	Do you support stricter control of gender pay?
Soll der Bund die finanzielle Unterstützung für die berufliche Weiterbildung und Umschulung ausbauen?	Should the Federal Government expand its financial support for continuing vocational training and retraining?
Sollen in der Schweiz vermehrt Spitäler geschlossen werden, um die Kosten im Gesundheitsbereich zu senken?	Should more hospitals be closed in Switzerland to reduce healthcare costs?
Eine eidenössische Volksinitiative verlangt, dass der Bundesrat direkt vom Volk gewählt werden soll. Unterstützen Sie dieses Anliegen?	A sworn federal people's initiative demands that the Federal Council be elected directly by the people. Do you support this concern?
Soll die Einführung der elektronischen Stimmabgabe bei Wahlen und Abstimmungen (E-Voting) weiter vorangetrieben werden?	Should the introduction of electronic voting in elections and voting (e-voting) be further promoted?

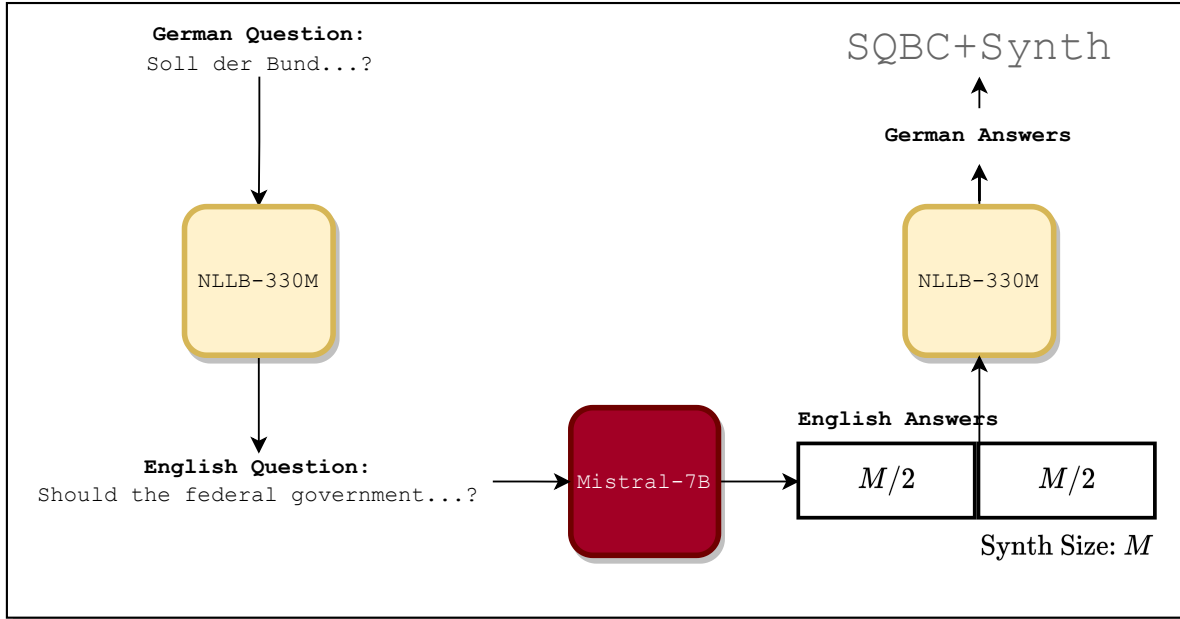
Table 7: Chosen questions for stance detection in German and their English translation

Comment in English	Comment in German
I don't believe that insured persons should contribute more to their own healthcare costs by increasing the minimum deductible. While it may seem like a good idea to increase deductibles as a way to reduce overall healthcare costs, this only shifts the burden onto patients and puts them at risk of facing high out-of-pocket expenses. It's important for insurance companies to find ways to control costs without placing the onus on insured individuals.	Ich glaube nicht, dass Versicherte mehr zu ihren eigenen Gesundheitskosten beitragen sollten, indem sie den Mindestabzug erhöhen. Die Erhöhung der Selbstbehalte als Mittel zur Senkung der gesamten Gesundheitskosten mag zwar als gute Idee erscheinen, verschiebt jedoch die Belastung nur auf die Patienten und gefährdet sie mit hohen Kosten, die sie selbst tragen müssen. Es ist wichtig für Versicherungsgesellschaften, Wege zu finden, Kosten zu kontrollieren, ohne die Versicherten mit der Last zu belasten.
Hi, friends! I think it's a great idea for insured person to contribute more to their health costs by increasing the minimum deductible. As someone who believes in taking ownership of my own health and finances, I believe that paying more out-of-pocket can help me better manage my expenses and make more informed decisions about my healthcare. Plus, I think that by contributing more, I can save money in the long run and avoid potential financial strain in other areas of my life.	Ich finde es eine gute Idee, dass Versicherte mehr zu ihren Gesundheitskosten beitragen, indem sie den Mindestabzug erhöhen. Als jemand, der daran glaubt, die Verantwortung für seine Gesundheit und seine Finanzen zu übernehmen, glaube ich, dass mehr Geld aus eigener Tasche helfen kann, meine Ausgaben besser zu managen und fundierte Entscheidungen über meine Gesundheitsversorgung zu treffen. Außerdem denke ich, dass ich durch einen größeren Beitrag auf lange Sicht Geld sparen und finanzielle Belastungen in anderen Bereichen meines Lebens vermeiden kann.
"In my opinion, it's wrong to link the residence permit for migrants from non-EU/EFTA countries to the fulfillment of binding integration agreements in Switzerland. This policy puts too much emphasis on the ability of migrants to adapt to Swiss culture and language at the expense of their personal freedoms and rights as human beings. It also creates incentives for migrants to cheat or lie in order to gain a residency permit and access to valuable services and resources."	"Meiner Meinung nach ist es falsch, die Aufenthaltserlaubnis für Migranten aus Nicht-EU/EFTA-Ländern mit der Erfüllung verbindlicher Integrationsvereinbarungen in der Schweiz zu verbinden. Diese Politik legt zu viel Wert auf die Fähigkeit der Migranten, sich an die Schweizer Kultur und Sprache anzupassen, auf Kosten ihrer persönlichen Freiheiten und Rechte als Menschen. Es schafft auch Anreize für Migranten, zu betrügen oder zu lügen, um eine Aufenthaltserlaubnis zu erhalten und Zugang zu wertvollen Dienstleistungen und Ressourcen zu erhalten "
I believe that the residence permit for migrants from non-EU/EFTA countries should be linked to the fulfillment of binding integration agreements throughout Switzerland. This is because integration should not be seen as a one-sided process, but rather as a mutually beneficial relationship between the immigrant and the host country. By linking the residence permit to the successful completion of integration agreements, we incentivize immigrants to actively participate in the integration process and become active members of Swiss society. Additionally, this approach ensures that immigrants do not receive a free ride by simply obtaining a residence permit without contributing to the well-being of Swiss society. Ultimately, a comprehensive integration approach benefits both the immigrant and the host country by creating a more inclusive and harmonious society.	Ich glaube, dass die Aufenthaltserlaubnis für Migranten aus Nicht-EU/EFTA-Ländern mit der Erfüllung verbindlicher Integrationsvereinbarungen in der gesamten Schweiz verbunden sein sollte. Die Integration sollte nicht als einseitiger Prozess, sondern als eine gegenseitig vorteilhafte Beziehung zwischen dem Einwanderer und dem Aufnahmeland betrachtet werden. Durch die Verknüpfung der Aufenthaltserlaubnis mit dem erfolgreichen Abschluss von Integrationsvereinbarungen fördern wir die aktive Teilnahme der Einwanderer am Integrationsprozess und die Förderung ihrer Teilnahme an der Schweizer Gesellschaft. Darüber hinaus wird durch diese Vorgehensweise sichergestellt, dass Einwanderer nicht einfach eine Aufenthaltserlaubnis erhalten, ohne zum Wohlergehen der Schweizer Gesellschaft beizutragen. Letztlich kommt einem umfassenden Integrationsansatz sowohl der Einwanderer als auch dem Aufnahmeland zugute, da er eine integrativere und harmonischere Gesellschaft schafft

Table 8: Sample of translated comments from comments generated by the LLM used for fine-tuning the stance detection model.

Comment in English	Intended Stance
I don't believe that insured persons should contribute more to their own healthcare costs by increasing the minimum deductible. While it may seem like a good idea to increase deductibles as a way to reduce overall healthcare costs, this only shifts the burden onto patients and puts them at risk of facing high out-of-pocket expenses. It's important for insurance companies to find ways to control costs without placing the onus on insured individuals.	AGAINST
Hi, friends! I think it's a great idea for insured person to contribute more to their health costs by increasing the minimum deductible. As someone who believes in taking ownership of my own health and finances, I believe that paying more out-of-pocket can help me better manage my expenses and make more informed decisions about my healthcare. Plus, I think that by contributing more, I can save money in the long run and avoid potential financial strain in other areas of my life.	FAVOR
"In my opinion, it's wrong to link the residence permit for migrants from non-EU/EFTA countries to the fulfillment of binding integration agreements in Switzerland. This policy puts too much emphasis on the ability of migrants to adapt to Swiss culture and language at the expense of their personal freedoms and rights as human beings. It also creates incentives for migrants to cheat or lie in order to gain a residency permit and access to valuable services and resources."	AGAINST
I believe that the residence permit for migrants from non-EU/EFTA countries should be linked to the fulfillment of binding integration agreements throughout Switzerland. This is because integration should not be seen as a one-sided process, but rather as a mutually beneficial relationship between the immigrant and the host country. By linking the residence permit to the successful completion of integration agreements, we incentivize immigrants to actively participate in the integration process and become active members of Swiss society. Additionally, this approach ensures that immigrants do not receive a free ride by simply obtaining a residence permit without contributing to the well-being of Swiss society. Ultimately, a comprehensive integration approach benefits both the immigrant and the host country by creating a more inclusive and harmonious society.	FAVOR
As someone who is against stricter controls on equal pay for women and men, I believe that all individuals should be paid based on their individual skills, performance, and contributions to the company, regardless of their gender. I think it's important to recognize and reward hard work and dedication, regardless of whether someone is male or female. Additionally, I argue that it's already illegal and unethical to discriminate on the basis of gender and that such regulations would only create unnecessary tensions and potential conflicts within the workplace. Instead, we should focus on creating a truly equitable society where everyone has access to equal opportunities for education, job training, and development.	AGAINST
I personally believe that stricter controls on equal pay for women and men are necessary because it's not always fair that men receive higher salaries just because of their gender. In many cases, women have to work harder than men to achieve the same level of performance, but they still earn less money. Stricter controls would help ensure that everyone's hard work is recognized and rewarded fairly, regardless of their gender. Additionally, it's important to promote gender equality and give all individuals an equal opportunity to succeed. Overall, I think stricter controls on equal pay are necessary to help reduce the gender pay gap and create a more equitable society.	FAVOR

Table 9: Sample of comments generated by the LLM used for fine-tuning the stance detection model.



C.2 Additional Experimental Details

Evaluation. For fine-tuning and testing we evaluate the given model separately on 10 chosen questions from the test dataset of X-Stance for all experiments. For each question q we split $\mathcal{D}_{\text{test}}^{(q)}$ into a 60/40 train/test split (repeated with 5 different seeds to get error bars) and use the train split for fine-tuning to the given question and the test split for evaluation. Our main results report the average F1 score over 10 selected questions from the test dataset evaluated on the comments from the test split. The error bars represent the average standard deviation over the 10 questions for 5 runs with different seeds.

Compute and Runtime. We conduct our experiments on a single NVIDIA A100 80GB GPU and a 32 core CPU. With this setup, for Mistral-7B, the generation of synthetic data takes approximately 3 hours per question for a synthetic dataset size of $M = 1000$. Fine-tuning the BERT model with the synthetic data takes less than a minute. For the active learning methods, the selection of the most informative samples takes less than a minute. Hence the largest computational effort is the generation of the synthetic data.

Translation of the X-Stance dataset for synthetic data generation. In Figure 7 we show the pipeline for translating the X-Stance dataset for synthetic data generation. We start with a question q from the X-Stance test dataset and translate the question to English with a NLLB-330M model [NLLB Team et al., 2022]. Then we let the Mistral-7B model

Figure 7: **Overview of the pipeline for active learning with synthetic data:** We start with a question q from the X-Stance test dataset and translate the question to English with a NLLB-330M model [NLLB Team et al., 2022]. Then we let the Mistral-7B model generate synthetic data, i.e., comments for the translated question. The generated comments are then translated back to German to be used for fine-tuning the model in our experiments.

generate synthetic data, i.e., comments for the translated question. The generated comments are then translated back to German to be used for fine-tuning the model in our experiments.

D Oh-A-DINO

D.1 Calculating similarities.

We calculate the similarity on patch level between a query image x and a candidate image x' using the representations v and v' . Since v and v' share the same embedding dimension, we can calculate the similarity between the two patches $v_i \in \mathbb{R}^{n_v}$ and $v'_j \in \mathbb{R}^{n_v}$ by computing the pairwise cosine similarity between the two representations:

$$S_{ij} = \frac{v_i \cdot v'_j}{||v_i|| \cdot ||v'_j||} \in \mathbb{R}, \quad (1)$$

for all $i, j \in \{1, \dots, p\}$.

Then, for each query patch v_i we retrieve the patch with the highest cosine similarity, i.e., the most similar patch from v'

$$s_i^{\max} = \max_{j=\{1, \dots, p\}} S_{ij}. \quad (2)$$

The list $S^{\max} = [s_1^{\max}, \dots, s_p^{\max}]$ then contains the cosine similarities for the most similar patches from v' for each patch in v . Finally, our *similarity score* of the query image x with another image x' is the average of the similarities in $S^{\max}$, that is $\bar{s} = \text{average}(S^{\max})$. Note, if we compare x against n candidate images x' , we will have n similarity scores for the given query image x .

D.2 Additional Dataset Details

Dataset	#Images	#Objects	#Shapes	#Colors	#Materials	#Backgrounds
CLEVR	100k	3-10	3	8	2	1
CLEVRText	50k+10k	3-10	4	–	60	60

Dataset Statistics. We provide details on the number of attributes and objects in the CLEVR and CLEVRText dataset in Table 10. While both datasets have the same number of possible objects in an image, the CLEVRText Dataset has different materials compared to the CLEVR dataset which has 8 colors and 2 materials. This presents different challenges to the learnt representations. In the CLEVR case, the representations must account for redundancy in the object features, while for the CLEVRText dataset the large number of materials makes retrieval more challenging since some materials are quite similar and only differ slightly.

Table 10: Dataset characteristics for CLEVR and CLEVRText. The CLEVRText dataset has additional textures for the objects and backgrounds compared to CLEVR. It also has no color information for the objects, which is replaced by the material information. In general, instance retrieval with the CLEVRText dataset is much harder due to the increased number of possible object configurations.

Stanford Cars Dataset Splits

<i>Validation-Query</i>	<i>Candidates</i>	<i>Query Size</i>
175	5000	25

CLEVR/CLEVRText Dataset Splits

<i>Train</i>	<i>Validation-Query</i>	<i>Candidates</i>	<i>Query Size</i>
2000	500	5000	50

D.3 Experiment Details

VAE Hyperparameters

<i>Parameter</i>	<i>Value</i>
Input size	$64 \times 64 \times 3$
Latent size	32
Learning rate	$1e-4$
β	$1e-4$

DINO Hyperparameters

<i>Parameter</i>	<i>Value</i>
Input size	518×518
Embedding Dimension	384
Patch Size	14

DINO Hyperparameters. Note that for DINOv2 and DINOv3 we used the non register variants since we noticed no difference in our analysis and the non register variants were faster.

Compute. The experiments where performed on an RTX 3090 24GB VRAM with Ryzen 3900X CPU. The main bottleneck is running the experiments which requires caching all the required query and candidate images. For this, we used 128GB of RAM.

Table 11: **Data splits for Stanford Cars dataset and CLEVR and CLEVRText datasets.** The train set is used to train the VAE, while the validation set is used to select query images for the retrieval task. The test set contains the candidate images for the retrieval task. We only evaluate on the Stanford Cars Dataset where we take a subset of 5000 candidate images and 175 query images split into groups of 25.