Université de Montréal

Toward Causal Representation and Structure Learning

par

Sayed Mohammadamin Mansouri Tehrani

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Mémoire présenté en vue de l'obtention du grade de
Maître ès sciences (M.Sc.)
en Informatique

August 31, 2023

Université de Montréal

Faculté des arts et des sciences

Ce mémoire intitulé

Toward Causal Representation and Structure Learning

présenté par

Sayed Mohammadamin Mansouri Tehrani

a été évalué par un jury composé des personnes suivantes :

Guillaume Rabusseau

(président-rapporteur)

Irina Rish

(directeur de recherche)

Eugene Belilovsky

(membre du jury)

Résumé

Dans les annales de l'Intelligence Artificielle (IA), la quête incessante pour émuler la cognition humaine dans les machines a sous-tendu l'évolution technologique, repoussant les limites du potentiel humain et des capacités de résolution de problèmes. L'intégration de l'IA a catalysé des progrès remarquables, pénétrant divers domaines et redéfinissant des industries.

Cependant, un défi demeure imperturbable : l'obstacle de la généralisation hors de la distribution (OOD). Alors que l'IA triomphe avec des données familières, elle échoue avec des données en dehors de son domaine d'entraînement. En santé, en finance et au-delà, les limitations de l'IA entravent l'adaptation à des scénarios nouveaux. Cette lacune découle de l'écart entre les schémas appris et les caractéristiques causales et invariantes sous-jacentes, entravant l'adaptabilité à des scénarios inexplorés.

Cette thèse franchit des étapes significatives pour aborder cette question en innovant et en exploitant des méthodes issues de l'apprentissage de structure causale et de représentation. Le parcours commence par un algorithme novateur d'apprentissage de structure, les "Reusable Factor Graphs", qui tire parti des biais inductifs issus de la causalité et de la cognition humaine pour une meilleure généralisation. Ensuite, en explorant l'apprentissage de représentation causale, nous découvrons des représentations désenchevêtrées centrées sur les objets en utilisant une supervision faible basée sur une connaissance partielle de la structure causale des données. Ces connaissances se conjuguent pour préconiser l'apprentissage conjoint de la structure causale et de la représentation. L'architecture proposée, les "Reusable Slotwise Mechanisms" (RSM), relie théorie et pratique, démontrant une promesse réelle à travers ses représentations centrées sur les objets et ses mécanismes causaux réutilisables. Cette fusion offre une solution potentielle pour surmonter les limitations de la généralisation OOD en IA.

Mots-clés: apprentissage automatique, apprentissage profond, apprentissage de représentation causale, apprentissage de structure, mécanismes causaux, généralisation hors de la distribution, apprentissage centré sur les objets, représentations désenchevêtrées

Abstract

In the annals of Artificial Intelligence (AI), an enduring quest to emulate human cognition in machines has underpinned technological evolution, driving the boundaries of human potential and problem-solving capabilities. The integration of AI has catalyzed remarkable progress, infiltrating various domains and redefining industries.

Yet, a challenge remains unshaken: the hurdle of out-of-distribution (OOD) generalization. While AI triumphs with familiar data, it falters with data outside its training realm. In healthcare, finance, and beyond, AI's limitations hinder adaptation to novel scenarios. This deficiency arises from the gap between learned patterns and underlying causal and invariant features, hindering adaptability to uncharted scenarios.

This thesis takes significant steps toward tackling this issue by innovating and leveraging methods from causal structure and representation learning. The journey begins with an innovative structure learning algorithm, Reusable Factor Graphs, leveraging inductive biases from causality and human cognition for improved generalization. Next, delving into causal representation learning, we uncover object-centric disentangled representations using weak supervision from partial knowledge of the causal structure of data. These insights synergize in advocating joint learning of causal structure and representation. The proposed Reusable Slotwise Mechanisms (RSM) architecture bridges theory and practice, demonstrating real-world promise through its object-centric representations and reusable causal mechanisms. This fusion offers a potential solution for tackling OOD generalization limitations in AI.

Keywords: machine learning, deep learning, causal representation learning, structure learning, causal mechanisms, out-of-distribution generalization, object-centric learning, disentangled representations

Contents

List of tables

List of figures

List of acronyms and abbreviations

General AI	
AI	Artificial Intelligence
ML	Machine Learning
DL	Deep Learning
NN	Neural Network
BN	Bayesian Network
FC	Fully Connected (layer)
MLP	Multilayer Perceptron
CNN/ConvNet	Convolutional Neural Network
RNN	Recurrent Neural Network
ReLU	Rectified Linear Unit
LSTM	Long Short-Term Memory
GRU	Gated Recurrent Unit
i.i.d.	independent and identically distributed
OOD	Out-of-distribution
MSE	Mean Squared Error
SGD	Stochatic gradient descent
DGP	Data Generation Process
Adam	Adaptive Moment Estimation
Methodology	
RFG	Reusable Factor Graphs
RSM	Reusable Slotwise Mechanisms
TD-RFG	Time-Directed Reusable Factor Graphs
DAG	Directed Acyclic Graph
CGM	Causal Graphical Model
ICP	Invariant Causal Prediction
SCM	Structural Causal Models

GST	Global Workspace Theory
ICA	Independent Component Analysis
EM	Expectation-Maximization technique
VAE	Variational autoencoder
LP	Linear Program
CLP	Constrained Linear Programs
LR	Linear Regression
RP	Random Projections
PC	Principal Components
GNNs	Graph Neural Networks
LD	Linear Disentanglement
MCC	Mean Correlation Coefficient
CCI	Central Contextual Information
HSV	A colour encoding using Hue, Saturation, Value instead of RGB
VQA	Visual Question Answering
HD	Hamming Distance
Models	
SA	Slot Attention
MESH	Minimize Entropy of Sinkhorn
SA-MESH	Slot Attention augmented with MESH
SAVi	Slot Attention for Video
SAVi-Dyn	SAVi augmented with Transformer-LSTM dynamics module
STEVE	Slot-TransformEr for VidEos
NPS	Neural Production Systems
C-SWM	Contrastively-trained Structured World Models
ResNet	Residual Networks
Datasets	
CLEVR	Compositional Language and Elementary Visual Reasoning
CLEVRER	CoLlision Events for Video REpresentation and Reasoning
PHYRE	PHYsical REasoning

Acknowledgements

I would like to express my sincere gratitude to my research advisor, Irina Rish. Undoubtedly Irina has been the most nurturing and benevolent mentor I could have envisioned, and she has certainly been the most influential academic figure I have met in the past three years. The privilege of having her as my advisor is something I hold in great esteem, and I will always be profoundly grateful for the opportunity to work under her guidance and joining her lab. Her unwavering support has enabled me to delve into exciting realms of research, collaborate on various projects, participate in internships and workshops that have greatly enriched my academic experience. She has played an instrumental role in helping me find my footing in this field, encouraging me to explore, persevere through failures, and ultimately triumph. Her impact on my life and personal growth is immeasurable and difficult to put into words.

I am deeply grateful to Kartik Ahuja and Jason Hartford, whom I would like to hold in very high regard as my unofficial co-supervisors. Collaborating with such brilliant theoreticians has been an immense blessing and a wellspring of inspiration, propelling me to redouble my efforts, delve further into my research, and approach challenges with unparalleled rigor. Countless vibrant discussions, spanning research concepts, mathematical intricacies, and career pathways, have enriched my journey, crafting a treasury of cherished memories. The privilege of working alongside them and absorbing the wealth of their expertise over a span of more than two years is a stroke of great fortune that I hold dear.

I extend my sincere gratitude to Yoshua Bengio, whose contribution in offering invaluable high-level guidance has been pivotal. Through skillful steering of the projects and posing brilliantly insightful questions, he has played a crucial role in leading them towards success. Yoshua's inquiries, characterized by profound contemplation, have consistently ignited our curiosity to venture into unexplored territories. His presence exudes warmth and motivation, serving as a source of encouragement throughout. I have been consistently astounded by his remarkable ability to swiftly grasp and assimilate complex ideas, his extensive reservoir of knowledge, unwavering dedication to rigor, and his propensity to generate a continuous stream of innovative concepts. His unwavering commitment to scientific excellence, coupled with his relentless pursuit of refining ideas, has been a guiding light. It is a genuine honor to have had the privilege of collaborating with him, learning from him, and being truly inspired by his remarkable qualities.

I would also like to thank my collaborators Trang Nguyen, and Yan Zhang. Trang started with us as an exceptionally hardworking and smart undergraduate student, and continued leading her paper during her master's with exemplar diligence and persistence to its perfection. Her perseverance and determination in surmounting challenges to see her initial endeavor through to its completion has been a profound source of inspiration for me. I am also grateful to Yan, a truly talented collaborator, whose integration into our team significantly contributed to enhancing the robustness and elegance of our method, making it operate seamlessly and proficiently.

In the end, I wish to extend my heartfelt gratitude to my immediate family – my mother Maryam, older brother Ehsan, twin brother Erfan, and father Ali – for their unwavering support and boundless love throughout my academic journey. I am forever indebted to Maryam for her enduring love that has been a guiding light; from the nights she supported my dream of learning about stars by taking care of me while I was left in awe, exploring the mysteries of the deep sky with a telescope, to when she flew across the ocean to Canada to warm my heart, she has been a source of immeasurable strength. I am also profoundly grateful to Ehsan, who has not only been a guiding older brother, but also a mentor whose presence has been a steady beacon. When I was younger, he taught me calculus and physics to prepare for the national Olympiad competition, and when I grew older, he imparted invaluable life lessons. His guidance throughout my undergraduate and graduate journey has been instrumental in making crucial career decisions. I extend my heartfelt gratitude to Erfan and Ali for their unwavering emotional support, believing in me and propelling me to achieve beyond my expectations over these transformative years.

Introduction

Throughout history, the pursuit of Artificial Intelligence (AI) has been fueled by an unwavering human fascination with recreating intelligence in machines. The ambition to replicate human thought processes and decision-making within algorithms has driven AI's evolution, transcending generations and shaping technological paradigms. From the visionary aspirations of early computer scientists to the transformative applications of today, the quest for AI has been underpinned by the desire to amplify human potential, augment problem-solving capabilities, and forge connections between human cognition and machine operation. In recent years, the progress of AI has been nothing short of remarkable, permeating various facets of our lives and revolutionizing industries ranging from healthcare to transportation. State-of-the-art AI systems have astounded us with their abilities to diagnose diseases from medical images, translate languages with human-like fluency, and even defeat world champions in complex games. As AI has advanced, the fields of Machine Learning (ML) and Deep Learning (DL) have emerged as pivotal paths in this journey, each contributing distinct dimensions to the realization of intelligent systems. Machine Learning, nestled within the AI framework, represents a paradigm shift that pivots away from traditional rule-based programming. Instead, it empowers algorithms to learn patterns and relationships from data, adaptively refining their performance over time. This adaptive learning capability has bestowed ML systems with the capacity to tackle complex problems, ranging from language translation to fraud detection, by discerning patterns that might elude conventional programming approaches. ML's versatility and ability to extrapolate insights from data have elevated it to a cornerstone of modern AI, facilitating predictive modeling, classification, clustering, and more. Within the realm of Machine Learning, Deep Learning emerges as a subset that stands out for its remarkable aptitude in handling complex, high-dimensional data. Inspired by the architecture of neural networks in the human brain, Deep Learning models consist of multiple layers that hierarchically extract intricate features from input data. This hierarchical feature extraction enables DL models to excel in tasks like image recognition, natural language processing, and even playing strategic games. The success of Deep Learning stems from its capacity to automatically learn hierarchical abstractions, capturing nuanced representations from massive datasets.

Yet, even within this era of triumphs, a persistent hurdle remains: the challenge of out-of-distribution (OOD) generalization. While AI systems excel when presented with data that resembles their training samples, they often falter when confronted with inputs that deviate even slightly from their training distribution. This phenomenon, known as out-of-distribution generalization, highlights a fundamental limitation of contemporary AI models. Out-of-distribution generalization encapsulates the limitation of AI systems to adapt to new or uncommon situations that differ from their training data. Imagine a model trained to recognize faces under perfect lighting conditions. When exposed to low-light or obscured images, its performance plummets, revealing the fragility of its training. This deficiency in adapting to novel instances stems from the fact that AI systems are often bound by the patterns they have learned during training that do not necessarily correspond to the underlying *causal* and *invariant* features of interest, failing to generalize effectively to new and unforeseen data points. For more examples, consider a self-driving car trained on clear, sunny day scenes. When confronted with foggy or rainy conditions, the car's ability to perceive obstacles and make informed decisions might significantly degrade due to its lack of exposure to such conditions during training. OOD generalization challenge poses significant implications across various sectors. In healthcare, models trained on specific patient populations might fail to provide accurate diagnoses for individuals with unique medical conditions. In finance, AI systems optimized for normal market conditions could prove ineffective during unexpected economic fluctuations. Addressing the challenge of out-of-distribution generalization is imperative for unlocking the full potential of AI in diverse real-world scenarios.

In this thesis, we embark on a journey to address the central challenge of out-of-distribution generalization through the lens of causal structure and representation learning. Our research is fueled by the understanding that effectively learning causal relationships and building robust representations could offer the key to bridging the gap between AI's performance on known data and its adaptability to unforeseen scenarios. After a brief introduction to the foundations required to grasp the rest of the thesis in Chapter 1, in Chapter 2, we start by studying causal structure learning in the form of reusable factor graphs to learn the reusable relationships among causal variables, assuming the availability of the latter. The latter assumption is then relaxed in Chapter 3 where we shift our focus to causal representation learning by leveraging weak assumptions on the causal structure. The results of chapters 2 and 3 then converge into a practical proposal in Chapter 4 for joint causal structure and representation learning in various challenging real-world tasks, enabling the generalization of performance to unseen distributions.

The first article (Chapter 2) proposes an innovative structure learning algorithm named Reusable Factor Graphs and navigates theoretical waters and tests them on synthetic environments. The goal is to find an efficient way for leveraging crucial inductive biases from both causality and higher-level human cognition into structure learning for better generalization. These inductive biases include the principle of independent causal mechanisms, modularity and reusability of such mechanisms, and the sparsity of their input space. This work enhances the sample efficiency of structure learning by successfully showing how we can exploit the reusability of causal mechanisms and thus opens avenues for more effective causal discovery.

In the first article, we assume access to causal representations and leverage their temporal changes to uncover a (reusable) causal structure. However, the availability of such causal representations is far from guaranteed. Thus, in the second article (Chapter 3), we pivot our focus to the realm of causal representation learning, effectively revisiting the challenge posed in the first article. Crucially, this work also departs from the traditional assumptions of causal representation learning and embraces the idea that natural phenomena often revolve around objects and their interactions and proposes an algorithm, for the first time, to learn *object-centric* disentangled representations, as opposed to monolithic fixed-size vector representations. This work harnesses weak supervision derived from partial and incomplete knowledge of the underlying causal structure inherent within observations to learn causal representations. We also demonstrate how embracing the object-centricity of the natural world can lead to significant sample efficiency gains for learning such representations.

These articles interact in a complementary manner, setting the stage for the third and our concluding paper. The first article starts by assuming access to causal representations, which are used to infer the causal structure. In contrast, the second article takes a different approach and deals with the challenge of learning causal representations in the presence of scarce knowledge of the causal structure in the form of weak supervision. The synergy of these insights suggests a promising avenue—a direction that naturally emerges as both facets of the problem find their respective resolutions, and that is the joint learning of the causal structure and representation given the gathered insights on how they enable one another.

Finally, the third article (Chapter 4) bridges theory and practice by introducing the Reusable Slotwise Mechanisms (RSM) architecture. Through joint learning of object-centric representations and leveraging the reusability of causal mechanisms, RSM demonstrates promise in real-world scenarios. The culmination of these articles emphasizes the potential for merging causal structure and representation learning to tackle the challenge of out-of-distribution generalization.

Contributions

Contributions of this thesis encompass a threefold exploration into the realm of out-of-distribution generalization. The first contribution, presented in the initial article, introduces the Reusable Factor Graphs algorithm, highlighting an effective means to integrate key inductive biases from both causal understanding and higher-level cognitive processes into structure learning. The second contribution, represented by the subsequent article, proposes a pioneering approach to causal representation learning, uncovering object-centric disentangled representations sample efficiently by leveraging weak supervision. These two contributions collectively contribute to the overarching goal of augmenting the existing generalization capabilities of current deep learning methods, therefore, they merge seamlessly in the third and concluding article, where the Reusable Slotwise Mechanisms (RSM) architecture bridges theory and practice. Through the joint learning of object-centric representations and reusable causal mechanisms, this work holds potential for real-world application. Collectively, these contributions carve a path towards addressing the challenge of out-of-distribution generalization, harnessing insights from causal structure and representation learning to push the field forward.

Personal Contributions.

(1) **Structure Learning of Reusable Factor Graphs**
- The inception of the idea of RFG should be credited to Kartik, however, the final version of RFG, and TD-RFG is the result of discussions between me and Kartik
- I implemented and ran all the experiments (including hyperparameter search) with the exception of those resulting in figures 2.6, 2.7
- I produced and analyzed all the plots except for those in figures 2.6, 2.7
- I established the connection to EM and wrote the corresponding section.
- I came up with how to infer the number of factors when it is unknown, using soft Hamming Distance.
- I co-authored the paper together with Kartik.

(2) **Object-Centric Causal Representation Learning**
- I did the literature review surrounding object-centric learning and methods to use alongside Ahuja et al. (2022a) for disentanglement.

- I implemented an extensive and scalable codebase for the fast generation of numerous variations of both the 2D and 3D datasets, each designed to probe a specific aspect of our method and explore the possible failure modes. The datasets used different engines, therefore there was little transfer of code from one to another, and I carried out both from scratch.
- I implemented all the baselines, and our disentanglement method based on Slot Attention (SA) and SA-MESH. Then ran all the experiments with known and unknown perturbations, with various combinations of parameters, as well as the hyperparameter search. This was achieved due to the expansive codebase I had written for swift integration of different models and datasets.
- I extensively and meticulously troubleshooted the failure modes of our method resulting in crucial theoretical insights about the necessary conditions on the perturbations.
- I came up with three solutions to address the problem of matching, implemented them, extensively compared their run-time complexities and performance in various settings, and carried out the experiments with the fastest and best-performing algorithm.
- I produces all the plots, figures, and result tables.
- I wrote all of the paper with the exception of these sections: 3.3, 3.4, B.1
- I co-authored the rest of the paper together with Jason, Yan, and Kartik.

(3) **Reusable Slotwise Mechanisms**

- I did the literature review surrounding object-centric learning and dynamics modeling and the baselines NPS, C-SWM.
- I implemented an extensive and scalable codebase for swift prototyping of various models in conjunctions with different datasets. However, the implementation of the final version of RSM and fitting that alongside baselines in the codebase I provided, was done by Trang.
- I oversaw the smooth conduction of the experiment, providing frequent implementation and troubleshooting feedback to Trang. The troubleshooting includes delving into low-level details of coding, and suggesting experimentation to probe the various aspects of the model.
- The final version of the RSM algorithm was the result of discussions among myself, Dianbo, Kartik, and Yoshua.

- I provided guidance regarding the implementation and usage of object-centric methods.
- Trang and I produced figure 4.1.
- Throughout the various stages of the project, I mentored Trang in implementation, writing, presenting results, co-authoring author rebuttals for our submission, as well as presenting Trang with the ideas and math behind RSM and guiding her in realizing those in practice.
- I wrote the Introduction, Related Work, and Conclusion, and extensively edited the rest of the paper toward its final version.

Outline

The thesis is organized into five chapters, beginning with an introductory chapter that acquaints the reader with the foundational aspects of the subjects and the challenges addressed in the subsequent chapters. Each article is prefaced with context, and the author's individual contribution. Chapter 5 summarizes the thesis with an overview of the article conclusions, and is then followed by supplementary material for each paper in appendices A, B, and C.

Funding Acknowledgment

During the course of my master's degree, I have been fortunate to receive invaluable support through a range of scholarships. These include the Microsoft Scholarship (2020), the Excellence in Research scholarship from the University of Montreal (2021, 2022), the Excellence in AI Research award (2023), and the Excellence in research and joining different strands of the Montreal Declaration for a Responsible Development of Artificial Intelligence (2020). Additionally, I was granted the Bourse C scholarship, which covered my tuition fees throughout my studies from 2020 to 2023. I also wish to express my gratitude to my advisor Irina Rish and Mila (Quebec Artificial Intelligence Institute) for providing me with the necessary computer equipment. Furthermore, both Mila and the Digital Research Alliance of Canada (formerly known as Compute Canada) generously offered computational resources essential for conducting the various experiments undertaken in this thesis.

Chapter 1

Background

This section provides an overview of the key concepts that are essential for comprehending the contributions of this work. It begins by reviewing the fundamentals of machine learning, including its foundations in statistics and various types of learning paradigms. It is then followed by an introduction of neural networks and various layers that often are employed within them. Subsequently, we provide an overview of optimization methods for training deep models and techniques for stabilizing them. The chapter ends with touching on the topics of representation learning and delving into the problem of OOD generalization and its relation to causal learning.

1.1. Machine Learning

The field of machine learning addresses problems where deterministic solutions through human-interpretable rules are infeasible. It is a branch of applied statistics that leverages computers to model complex data distributions and solve challenging problems without the need for explicit rule-based programming. The primary goal of machine learning is to develop models that can learn patterns and relationships from data and make accurate predictions or decisions on new, unseen data. This is achieved by training the models on a dataset and optimizing their parameters to minimize the discrepancy between predicted and actual outcomes. More concretely, given a dataset of input-output pairs $(x^{(i)}, y^{(i)})$ where $x^{(i)}$ represents the input data and $y^{(i)}$ represents the corresponding output or target value, the goal of machine learning is to find a model h_θ that is parameterized by θ that approximates the underlying relationship between x and y, such that $h(x^{(i)})$ provides accurate predictions of $y^{(i)}$ for new, unseen inputs. Machine learning tasks are typically categorized into several major paradigms: supervised learning, unsupervised learning, reinforcement learning, and weakly supervised learning.

- **Supervised Learning:** In supervised learning, the algorithm learns from a labeled dataset where each data point is associated with a corresponding target label. The goal is to learn a mapping from input data to output labels in order to make accurate predictions on new, unseen data. For example, in an image classification task, the algorithm learns to differentiate between various objects by analyzing a dataset of images paired with corresponding labels denoting the object in the image.
- **Unsupervised Learning:** Unsupervised learning involves working with unlabeled data, where the algorithm seeks to uncover hidden patterns or structures within the data. Clustering and dimensionality reduction are common tasks in unsupervised learning. Clustering involves grouping similar data points together, while dimensionality reduction aims to reduce the complexity of the data by representing it in a lower-dimensional space. An example of unsupervised learning is customer segmentation in marketing, where similar customer behavior is grouped together for targeted campaigns.
- **Reinforcement Learning:** Reinforcement learning focuses on training algorithms to make sequences of decisions in an environment to maximize a cumulative reward. This paradigm is often applied in tasks involving decision-making and control. In reinforcement learning, an agent interacts with an environment, learns from the consequences of its actions, and adjusts its strategy to achieve optimal performance. A classic example is training an AI agent to play games like chess or Go, where it learns to make moves that lead to winning outcomes.
- **Weakly Supervised Learning:** Weakly supervised learning bridges the gap between supervised and unsupervised learning. It involves learning from partially labeled or noisy data, where the labels are not fully precise or complete. This paradigm is particularly useful when obtaining accurate labels for training data is expensive or time-consuming. Weak supervision methods aim to extract useful information from imperfect labels, allowing the algorithm to learn patterns and make predictions in scenarios where fully labeled data is scarce.

The main emphasis of this thesis revolves around unsupervised and weakly supervised learning, and we provide further elaboration on these topics as well as supervised learning in the following sections for completeness. However, the scope of this work does not encompass reinforcement learning, and therefore, it will not be discussed any further.

1.1.1. Supervised Learning

Supervised learning constitutes a foundational component of machine learning, wherein models extract patterns from labeled data to formulate predictions. To illustrate this, consider the archetype of linear regression, a technique aimed at discerning the optimal linear relationship between input x and output y by identifying suitable values for the slope w and the intercept b, formulated as $y = wx + b$.

At its core, the objective is to minimize the difference between the predicted $\hat{y}$ and the observed y values. This minimization is typically accomplished by computing the mean squared error (MSE) across all data points, expressed as:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

Here, n signifies the count of data samples, while $\hat{y}_i$ denotes the predicted output for the i-th sample. Through iterative optimization techniques like gradient descent, the model refines the values of w and b to diminish the MSE.

Supervised learning serves as a powerful approach for constructing predictive models from labeled data. However, the effectiveness of this approach largely depends on the availability of labeled data, which can be limited or costly to acquire. In domains like natural language processing or video analysis, where the volume of data far exceeds the capacity for manual annotation, solely relying on labeled data becomes infeasible. This challenge is particularly evident in tasks such as language translation or video classification, where the nuances of human language or the complexity of visual content demand extensive training data to capture their richness. As a result, the traditional supervised learning paradigm encounters limitations in cases where the resource-intensive process of labeling data becomes a bottleneck. It is precisely in these situations that the significance of unsupervised and weakly supervised learning comes to the fore, offering alternative approaches that capitalize on vast amounts of available, yet unlabeled or partially labeled, data. By harnessing the inherent structures and patterns within such data, these techniques extend the reach of machine learning to tackle complex, real-world challenges that surpass the boundaries of traditional supervised methods.

1.1.2. Unsupervised Learning

Unsupervised learning constitutes a pivotal realm of machine learning where the primary objective is to discern patterns, structures, or relationships within a dataset devoid of explicit labels. Here, we expand on unsupervised learning by studying one of its popular methods, the k-means algorithm, and leverage it to elucidate concepts of capacity, overfitting, and underfitting, that are central to understanding the concept of **generalization**. It is important to note that unsupervised and supervised learning are not mutually exclusive and their boundaries are nebulous Goodfellow et al. (2016). That is why we demonstrate the concepts of capacity, overfitting, and underfitting through an example of unsupervised learning, whereas traditionally these are introduced by an example of supervised learning algorithms such as linear regression.

Unsupervised learning finds its niche in scenarios where the provided data lacks predefined labels, rendering traditional supervised methods infeasible. It is akin to tackling a puzzle without a picture on the box – the challenge lies in deciphering the inherent structures that underlie the data. Clustering, dimensionality reduction, and generative modeling are common unsupervised learning tasks.

A quintessential algorithm that represents unsupervised learning is k-means clustering. Given a dataset X comprising n data points $x^{(1)}, x^{(2)}, ..., x^{(n)}$, the k-means algorithm seeks to partition the data into k clusters, with each point belonging to the cluster whose centroid is closest. The k-means algorithm operates as follows:

- Initialization: Randomly select k initial cluster centroids $c_1, c_2, ..., c_k$.
- Assignment: Assign each data point $x^{(i)}$ to the nearest centroid c_j based on Euclidean distance: $j = \text{argmin}_j |x^{(i)} - c_j|^2$.
- Update: Recalculate centroids as the mean of the points assigned to each cluster: $c_j = \frac{1}{|C_j|} \sum_{x^{(i)} \in C_j} x^{(i)}$, where C_j is the set of data points assigned to cluster j.
- Repeat: Iterate the assignment and update steps until convergence.

The k-means algorithm operates as follows: Given a dataset and the desired number of clusters k, it initializes k cluster centroids randomly. The algorithm then iterates through two main steps. First, it assigns each data point to the nearest cluster centroid based on the Euclidean distance. Second, it updates the cluster centroids to the mean of the data points assigned to each cluster. These two steps iteratively refine the cluster assignments and centroids until convergence. Through these iterations, the algorithm aims to minimize the within-cluster sum of squared distances, effectively clustering data points around centroids that represent the center of each cluster. The k-means algorithm converges when the centroids no longer change significantly between iterations or after a specified number of iterations. The resulting centroids define the cluster centers, and the assignment of data points to clusters provides the clustering solution. Now with the help of this example, we can introduce the fundamental concepts of model capacity, overfitting, and underfitting that influence the performance of machine learning models, irrespective of their unsupervised or supervised context. Let us examine these within the k-means framework.

Capacity: The capacity of a model refers to its ability to capture intricate patterns in data. A k-means model's capacity is determined by the choice of k. Small k may result in the underrepresentation of data patterns, whereas excessively large k might lead to capturing noise rather than genuine clusters.

Overfitting: Overfitting occurs when a model learns noise or anomalies present in the training data, resulting in poor generalization to new data. In k-means, overfitting can manifest if k is set too high, causing the algorithm to partition noise into spurious clusters.

Underfitting: Underfitting transpires when a model's capacity is too low to capture the underlying patterns. In k-means, underfitting can materialize if k is set too low, leading to the amalgamation of distinct clusters.

These notions of capacity, overfitting, and underfitting, showcased in the realm of unsupervised learning, come together to shed light on the heart of generalization in machine learning. Capacity dictates a model's knack for capturing underlying patterns while overfitting warns against tailoring to noise, and underfitting reminds us of the dangers of oversimplification. This trio plays a role in how well a model extends its insights to new, unseen data. Generalization is like finding a sweet spot—capturing patterns without being bogged down by noise—enabling the model to truly grasp the essence of data. This harmony resonates through various machine learning techniques, underscoring the importance of comprehending generalization across different algorithms and applications.

1.1.3. Weakly Supervised Learning

In the realm of machine learning, gathering labeled data for supervised training can be a laborious endeavor. Weakly supervised learning arises as a strategy to mitigate the labeling burden while still harnessing the power of supervised methods. Weakly supervised learning addresses scenarios where complete, accurate labels are scarce or expensive to obtain. It operates under the premise that partial or noisy labels, along with domain knowledge or constraints, can guide model learning. This paradigm bridges the gap between supervised and unsupervised learning, allowing models to learn from imperfect information.

Consider an example of classifying images of objects into categories but obtaining detailed annotations for every object in a large dataset might be impractical. Instead of having precise object-level labels, you might have access to image-level labels indicating the presence of certain categories, such as "beach," "mountain," or "city" to guide the model's learning process. This is a weak form of supervision, as the labels are less informative than specifying the exact objects within the image.

Weakly supervised learning encompasses a variety of strategies to exploit limited labels effectively:

- **Multi-instance Learning:** In cases where multiple instances share a single label, multi-instance learning leverages the collective information from instances to make predictions (Dietterich et al., 1997; Maron and Lozano-Pérez, 1998; Andrews et al., 2003).
- **Noisy Labels:** Models can be trained to tolerate and adapt to noisy labels by incorporating uncertainty measures during training (Reed et al., 2014; Goldberger et al., 2016; Sukhbaatar et al., 2014).
- **Constraint-based Learning:** Introducing domain-specific constraints or rules can guide the learning process in the absence of accurate labels (Patel and Dolz, 2022; Pathak et al., 2015).

The landscape of machine learning is intricate and nuanced, with unsupervised, supervised, and weakly supervised learning often intertwined. Weakly supervised learning's versatile nature allows it to harness the strengths of both unsupervised and supervised learning. It demonstrates that harnessing even limited labeling information can yield meaningful insights and predictions. As we will see in chapter 3, a form of weak supervision is exactly what allows for a minimal realistic assumption on the data distribution that still yields useful results for representation learning.

1.2. Deep Learning

In this section, we provide an introduction to deep learning, a subset of machine learning that harnesses the power of neural networks with multiple layers. These networks excel in capturing complex patterns and representations within data, making them a fundamental tool in modern AI research and applications.

1.2.1. Neural Networks

In this section, we provide an introduction to the fundamental components of neural networks, which serve as the building blocks of deep learning models. Neural networks are powerful architectures capable of learning intricate patterns from data. They consist of layers that transform input data through a series of operations to produce meaningful outputs.

Layers in a Neural Network. A neural network typically comprises an input layer, hidden layers, and an output layer. Each layer consists of multiple neurons, also known as nodes, which process and propagate information through weighted connections. A layer's output serves as the input to the subsequent layer, creating a hierarchical representation of the data.

Fully Connected (Dense) Layers. A fully connected layer, also referred to as a dense layer, is the simplest type of layer in a neural network. Neurons in this layer are connected to all neurons in the previous layer. Let x be the input vector of size n, and W be the weight matrix of size $m \times n$, where m is the number of neurons in the current layer. The output y of the fully connected layer can be calculated as:

$$y = Wx + b$$

where b is the bias vector of size m.

Convolutional Layers. Convolutional layers are essential for processing grid-like data, such as images. They employ filters (kernels) to extract features from local regions of the input. Let I be the input feature map, K be the filter, and S be the stride. The output feature map O can be computed using the convolution operation:

$$O[i,j] = \sum_{p=0}^{P-1} \sum_{q=0}^{Q-1} I[i \cdot S + p, j \cdot S + q] \cdot K[p,q]$$

where P and Q are the filter dimensions.

Transpose Convolution (Deconvolution) Layers. Transpose convolution layers, also known as deconvolution layers, are used for upsampling or generating higher-resolution feature maps. Let I be the input feature map, K be the filter, and S be the stride. The output feature map O is computed as:

$$O[i,j] = \sum_{p=0}^{P-1} \sum_{q=0}^{Q-1} I[i \cdot S + p, j \cdot S + q] \cdot K[p,q]$$

Transpose convolution layers can help reconstruct spatial details lost during pooling or downsampling operations.

Activation Functions. Activation functions introduce non-linearity to neural networks, enabling them to model complex relationships in data. One common activation function is the Rectified Linear Unit (ReLU), defined as:

$$f(x) = \max(0, x)$$

ReLU has gained popularity due to its simplicity and effectiveness in preventing the vanishing gradient problem (see next section on recurrent layers), allowing deep networks to be trained more effectively. Sigmoid and hyperbolic tangent (tanh) are other activation functions often used. The sigmoid function is defined as:

$$\sigma(x) = \frac{1}{1 + \exp(-x)}$$

It maps inputs to values between 0 and 1, making it suitable for binary classification problems. The hyperbolic tangent (tanh) function is defined as:

$$\tanh(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)}$$

It maps inputs to values between -1 and 1, providing a centered activation that can also help mitigate the vanishing gradient problem.

Recurrent Layers. Recurrent layers are a crucial component of neural networks designed to handle sequential data, such as time series or natural language. Unlike feedforward layers, recurrent layers possess connections that loop back, allowing them to maintain a hidden state that captures temporal information.

A general recurrent layer computes the hidden state h_t at time step t using the input x_t, the previous hidden state h_{t-1}, and the weight matrices W, U, and V. The hidden state update equation can be expressed as:

$$h_t = \sigma(Wx_t + Uh_{t-1} + b)$$

where σ represents the activation function and b is the bias vector.

However, the training of deep recurrent networks often faces the problem of vanishing gradients. This issue arises when gradients propagated backward through the network become extremely small, causing the network to learn slowly or even stagnate. The vanishing gradient problem is particularly evident in deep architectures with multiple layers.

Long Short-Term Memory (LSTM). LSTM is a popular form of recurrent layer designed to mitigate the vanishing gradient problem and capture long-term dependencies in sequential data. It achieves this by introducing three gating mechanisms: the input gate, the forget gate, and the output gate. These gates control the flow of information into and out of the cell state, enabling the LSTM to selectively retain or discard information.

Mathematically, an LSTM unit computes the hidden state h_t and the cell state c_t at time step t using the input x_t, the previous hidden state h_{t-1}, the previous cell state c_{t-1}, and learnable weight matrices and bias terms (in the equations below, $\odot$ denotes element-wise multiplication).

$$\begin{aligned}
f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \\
i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\
\tilde{C}_t &= \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \\
C_t &= f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \\
o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\
h_t &= o_t \odot \tanh(C_t)
\end{aligned}$$

Gated Recurrent Unit (GRU). GRU is another popular recurrent layer that simplifies the LSTM architecture while maintaining competitive performance. GRU introduces the update gate and reset gate, which determine the balance between retaining and updating information in the hidden state.

The update gate z_t and reset gate r_t are calculated using the input x_t, the previous hidden state h_{t-1}, and appropriate weight matrices. The hidden state h_t is then updated as a combination of the previous hidden state and a new candidate hidden state.

$$\begin{aligned}
z_t &= \sigma(W_z \cdot [h_{t-1}, x_t] + b_z) \\
r_t &= \sigma(W_r \cdot [h_{t-1}, x_t] + b_r) \\
\tilde{h}_t &= \tanh(W_h \cdot [r_t \odot h_{t-1}, x_t] + b_h) \\
h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t
\end{aligned}$$

Normalization Layers. Normalization layers, such as Batch Normalization (Ioffe and Szegedy, 2015), Layer Normalization (Ba et al., 2016), and Instance Normalization (Ulyanov et al., 2016), are employed to stabilize and accelerate the training process by reducing internal covariate shift. They ensure that the input to each layer has a consistent distribution, leading to more stable gradients and faster convergence.

Mathematically, Batch Normalization adjusts the mean μ and standard deviation σ of the input batch, applying a scale γ and shift β factor to normalize the output.

$$\begin{aligned} \mu_B &= \frac{1}{m}\sum_{i=1}^{m} x_i \\ \sigma_B^2 &= \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2 \\ \hat{x}_i &= \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \\ y_i &= \gamma\hat{x}_i + \beta \end{aligned}$$

In these equations, x_i represents the input to the Batch Normalization layer for the ith example in a mini-batch of size m. μ_B and σ_B^2 are the mean and variance computed over the mini-batch, respectively. $\hat{x}_i$ is the normalized input, γ and β are learnable scaling and shifting parameters, and y_i is the output of the Batch Normalization layer.

Batch Normalization helps stabilize training and accelerates convergence by normalizing the input distributions of layers within a neural network. The parameter ϵ is a small constant added to the denominator to prevent division by zero.

Layer Normalization and Instance Normalization perform similar operations but across different dimensions. Batch Normalization is commonly used in convolutional neural networks for image data, while Layer Normalization is preferred in recurrent networks to handle variable-length sequences. Instance Normalization is often applied in style transfer and image-to-image translation tasks.

Attention Mechanisms. Attention mechanisms enable neural networks to focus on specific parts of the input while ignoring others. They are particularly useful for tasks involving sequences, such as language translation. The attention weight a assigned to each input element can be calculated as:

$$a_i = \frac{\exp(e_i)}{\sum_{j=1}^{n}\exp(e_j)}$$

where e_i is a measure of compatibility between the current target element and the ith input element.

In summary, neural networks encompass a variety of layers, including fully connected, convolutional, transpose convolution, attention, and recurrent layers like LSTM and GRU, each serving specific purposes. Activation functions introduce non-linearity, aiding in feature extraction and pattern recognition. Normalization layers play a crucial role in stabilizing training and speeding up convergence by standardizing input distributions. These components work in harmony to extract features, capture patterns, and enable the network to learn complex mappings from input to output.

1.2.2. Neural Network Optimization

Neural network training involves finding the model parameters that minimize a chosen loss function. Optimization methods play a critical role in this process, as they determine how the model parameters are updated during training to converge towards an optimal solution. Various optimization algorithms have been developed, each with its strengths and weaknesses.

Gradient Descent. Gradient Descent is a fundamental optimization algorithm widely used in neural network training. Given a loss function L and model parameters θ, the goal is to minimize L by updating θ in the opposite direction of the gradient of L with respect to θ. The update rule is given by:

$$\theta_{t+1} = \theta_t - \eta \nabla L(\theta_t)$$

where η is the learning rate, controlling the step size of the updates.

Stochastic Gradient Descent (SGD). Stochastic Gradient Descent optimizes the loss function using random subsets of the training data, known as mini-batches. The update rule becomes:

$$\theta_{t+1} = \theta_t - \eta \nabla L(\theta_t; x^{(i)}, y^{(i)})$$

where $(x^{(i)}, y^{(i)})$ is a mini-batch sample.

Momentum. Momentum addresses the slow convergence of Gradient Descent by incorporating a moving average of past gradients. The update rule becomes:

$$v_{t+1} = \mu v_t - \eta \nabla L(\theta_t)$$

$$\theta_{t+1} = \theta_t + v_{t+1}$$

where v_t is the velocity at step t, and μ is the momentum hyperparameter.

Adagrad (Adaptive Gradient Algorithm). Adagrad (Duchi et al., 2011) adapts the learning rate for each parameter based on the historical gradient information. It provides larger updates for parameters with infrequent updates and smaller updates for frequently updated parameters. The update rule is given by:

$$g_{t+1} = g_t + (\nabla L(\theta_t))^2$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{g_{t+1} + \epsilon}} \nabla L(\theta_t)$$

where g_t is the sum of squared gradients up to step t, and ϵ prevents division by zero.

Adagrad's accumulation of squared gradients ensures that parameters with small gradients receive a larger learning rate adjustment, making it suitable for sparse data or features. However, this accumulation can cause the learning rates to shrink over time, leading to slow convergence.

Adam (Adaptive Moment Estimation). Adam (Kingma and Ba, 2015) combines the benefits of both Momentum and AdaGrad. It adapts the learning rate for each parameter based on the first and second moments of the gradients. The update rule is given by:

$$m_{t+1} = \beta_1 m_t + (1 - \beta_1) \nabla L(\theta_t)$$

$$v_{t+1} = \beta_2 v_t + (1 - \beta_2)(\nabla L(\theta_t))^2$$

$$\hat{m}_{t+1} = \frac{m_{t+1}}{1 - \beta_1^{t+1}}$$

$$\hat{v}_{t+1} = \frac{v_{t+1}}{1 - \beta_2^{t+1}}$$

$$\theta_{t+1} = \theta_t - \eta \frac{\hat{m}_{t+1}}{\sqrt{\hat{v}_{t+1}} + \epsilon}$$

where β_1 and β_2 are exponential decay rates, and ϵ prevents division by zero.

Adam's combination of momentum and adaptive learning rates makes it effective for a wide range of neural network architectures and tasks, thus it has become one of the most popular optimization methods of deep neural networks.

In summary, these optimization methods are essential for training neural networks effectively. Choosing the appropriate optimization algorithm and tuning its hyperparameters can significantly impact convergence speed and final performance.

1.2.3. Regularization in Neural Networks

Regularization is a fundamental technique in neural networks to prevent overfitting, where the model performs well on the training data but fails to generalize to new, unseen data. Overfitting occurs when the model captures noise in the training data, leading to poor performance on validation or test sets. Regularization methods aim to constrain the model's complexity, encouraging it to learn essential patterns rather than memorizing noise.
L2 Regularization (Weight Decay). L2 regularization, also known as weight decay, adds a penalty term to the loss function based on the squared magnitudes of the model's weights. This encourages the model to use smaller weights, preventing them from growing excessively. The modified loss function is

$$L_{\text{regularized}} = L_{\text{original}} + \frac{\lambda}{2}\sum_{i=1}^{N} w_i^2$$

where L_{original} is the original loss function, λ is the regularization parameter, N is the number of model parameters, and w_i represents the weights.
Dropout. Dropout (Hinton et al., 2014) is a widely used regularization technique that randomly drops a fraction of the neurons during each training iteration. This prevents individual neurons from relying too heavily on specific input features and encourages the network to learn robust representations. The dropout regularization can be applied to hidden layers using the following equation $a_{\text{dropout}} = a \odot d$ where a_{dropout} is the output after dropout, a is the original activation, and d is a binary mask that determines which neurons to drop.
Early Stopping. Early stopping is a form of regularization that monitors the model's performance on a validation set during training. It halts training when the validation loss stops decreasing or starts increasing, preventing the model from fitting noise in the training data.

Regularization methods are often used in combination to achieve better generalization. For example, a neural network can be trained with both L2 regularization and dropout to simultaneously control the complexity of weights and neuron interactions. Regularization techniques play a vital role in improving a neural network's generalization and robustness. By constraining the model's behavior, these methods help ensure that the network learns meaningful patterns and performs well on new, unseen data.

1.2.4. Representation Learning

Representation learning lies at the heart of deep learning, enabling models to automatically learn data representations that uncover essential patterns and features within the data. In deep learning, the objective is to find representations that simplify the task at hand, making it more amenable to computation and manipulation.

In essence, representation learning strives to transform raw input data into a form that exposes relevant characteristics in a more expressive and structured manner. Instead of relying on handcrafted features, representation learning empowers models to autonomously learn features directly from the data. Through this automatic feature extraction, the model repeatedly maps the input in various ways to a space in which solving the task at hand becomes easier. Classic examples include the classification of data where in the original domain they cannot be linearly separated, but there exist transformations under which the data becomes easily linearly separable.

Mathematically, representation learning in deep networks involves constructing a series of transformations applied to the input data. Each layer captures increasingly abstract and higher-level features:$h^{(1)} = f^{(1)}(x), h^{(2)} = f^{(2)}(h^{(1)}), \dots, h^{(L)} = f^{(L)}(h^{(L-1)})$. The process of representation learning constructs features that are not only discriminative for the task at hand but also transferable across related tasks. By learning invariant and robust features, deep networks are less likely to overfit to noisy or irrelevant aspects of the data, thus improving their ability to generalize to diverse scenarios. Pretrained models on large datasets can be repurposed for specific tasks with limited data, as the initial layers capture general features that are useful across domains. Fine-tuning the later layers to task-specific data further refines the model's performance.

Furthermore, deep representations often have the potential to disentangle factors of variation in the data, allowing the model to manipulate and interpret the learned features independently. This disentanglement enhances the interpretability of the model's decisions, promoting more informed and controlled generalization (chapters 3,4).

Representation learning in deep learning has revolutionized various applications by empowering models to generalize more effectively by capturing essential patterns, promoting transfer learning, disentangling factors of variation, and hierarchically structuring information. In natural language processing, for example, deep neural networks can learn word embeddings that capture semantic relationships between words, enabling more effective text analysis. Similarly, in computer vision, deep networks can autonomously learn to extract intricate features from images, leading to impressive image recognition capabilities.

1.3. Out-of-Distribution Generalization

Most of the contents in this and the following section are based on Liu et al. (2021); Schölkopf et al. (2021).

The landscape of contemporary machine learning methodologies has showcased remarkable advancements across diverse domains such as natural language processing, computer vision, and recommendation systems. While these techniques have demonstrated superiority in controlled experimental conditions, their vulnerability to data distribution shifts has emerged as a pressing concern. The potential consequences of such errors span a spectrum from minor inconveniences to severe implications in high-stakes domains like healthcare and autonomous driving. Below we will define the problem of out-of-distribution generalization formally and categorize the important methods that have been proposed to address this challenge.

Problem Definition. Consider a supervised learning scenario in which data is collected from distinct environments, each characterized by its own underlying probability distribution. Let $(X_e, Y_e) \sim \mathbb{P}^e$, where $X_e \in \mathcal{X}$ represents the feature random variable and $Y_e \in \mathcal{Y}$ signifies the corresponding label. Here, $e \in \mathcal{E} = \{1, \ldots, E\}$ denotes the index of environments, and $\mathcal{E}$ encompasses all potential environments. The collection $\mathcal{E}$ is partitioned into two subsets: $\mathcal{E}_{\text{seen}}$ representing observed environments and $\mathcal{E}_{\text{unseen}}$ encompassing unobserved ones ($\mathcal{E} = \mathcal{E}_{\text{seen}} \cup \mathcal{E}_{\text{unseen}}$). The training dataset consists of samples originating from $\mathcal{E}_{\text{seen}}$. Data from environment e is denoted as $\mathcal{D}_e = \{(x_i^e, y_i^e)\}_{i=1}^{n_e}$, where each data point (x_i^e, y_i^e) is an independent and identically distributed (i.i.d.) sample drawn from $\mathbb{P}^e$, and n_e represents the number of samples in environment e. The training dataset is the combination of all $\mathcal{D}_e$ for $e \in \mathcal{E}_{\text{seen}}$, which can be expressed as $\mathcal{D}_{\text{train}} = \bigcup_{e \in \mathcal{E}_{\text{seen}}} \mathcal{D}_e$.

Let $f_\theta : \mathcal{X} \to \mathcal{Y}$ denote a parametric model with parameters $\theta \in \Theta$. Define the risk associated with the model as $\mathcal{R}_\theta^e = \mathbb{E}_{(X_e, Y_e) \sim \mathbb{P}^e}[\ell(f_\theta(X_e), Y_e)]$, where ℓ represents the per-sample loss function (e.g., cross-entropy, squared loss). The objective of the Out-of-Distribution generalization problem is to learn a model that minimizes the maximum risk across different environments:

$$\min_{\theta \in \Theta} \max_{e \in \mathcal{E}} \mathcal{R}_\theta^e.$$

Since the training data is drawn only from $\mathcal{D}_{\text{train}}$ and does not include samples from unobserved environments, solving the above problem becomes a challenging task.

This problem can be partitioned into three components: (1) Feature representation of X (e.g., denoted as $g(X)$); (2) The mapping function $f_\theta(X)$ from features X to the label Y, often referred to as the model; (3) The optimization objective. There has been a surge of interest in tackling this problem from various perspectives toward this pipeline, which can be divided into the following categories:

- Unsupervised Representation Learning for OOD Generalization: This category includes methods involving unsupervised domain generalization and disentangled representation learning. These techniques utilize unsupervised representation learning approaches to enhance the initialization of downstream OOD generalization tasks, leading to improved feature representations.
- Supervised Model Learning for OOD Generalization: This group encompasses strategies like invariant representation learning, training tactics, causal learning, invariant risk minimization, stable learning, and heterogeneity-aware invariant learning. Various model architectures and learning strategies are designed within this category to enable OOD generalization.
- Optimization for OOD Generalization: This category considers methods that find distributionally robust optimization.

The contributions of this thesis fall under the first two categories, therefore, we will expand further on those in what follows.

Disentangled Representation Learning. Disentangled representation learning endeavors to acquire representations in which distinct and meaningful aspects of data variation are disentangled from each other Bengio et al. (2012); Locatello et al. (2019). This characteristic is indicative of representations of high quality and holds potential advantages for generalizing beyond the original distribution. The prevailing strategies for achieving disentanglement are primarily based on Variational Autoencoders (VAE Higgins et al. (2016); Kim and Mnih (2018)). These techniques are executed in an entirely unsupervised manner within a single environment, without requiring supplementary information. Both interpretability and sparsity are emphasized by these methods. In this context, "sparsity" pertains to the notion that minor alterations in distribution usually manifest in a sparse or localized manner within the disentangled decomposition Schölkopf et al. (2021).

Causal Learning. Causal learning approaches aspire to uncover the underlying causal structure inherent in the data and predict outcome variables based on the identified causal factors. By accurately discerning cause-and-effect relationships, these techniques are anticipated to exhibit strong performance even amidst changes in data distribution. This is rooted in the assumption that the fundamental causal structure remains invariant across different environments or domains.

We now delve deeper into the realm of causal learning, introducing its foundational concepts. At the heart of causal learning lies Assumption A, originating from the causal inference literature. This assumption posits a causally invariant relationship between the target variable Y and its direct causes $X_{\mathrm{pa}(Y)}$. Under this assumption, causal variables $X_{\mathrm{pa}(Y)}$ are anticipated to remain stable across various environments or data biases, fueling investigations into leveraging these causal variables exclusively.

Assumption A. (Causality Assumption Bühlmann (2018)). The structural equation models:

$$Y_e = f_Y(X^e_{\mathrm{pa}(Y)}, \epsilon^e_Y), \quad \epsilon^e_Y \perp X^e_{\mathrm{pa}(Y)}$$

remain consistent across all environments $e \in \mathrm{supp}(\mathcal{E}_{\mathrm{all}})$, signifying that ϵ^e_Y maintains the same distribution for all environments. Here, $\mathrm{pa}(Y)$ represents the direct causes of Y.

We proceed to examine methods tied to causal inference, which aim to extract causal variables from heterogeneous data. Although randomized experiments, such as A/B testing, are the gold standard for identifying causal effects, their practicality diminishes in real-world settings due to their cost and complexity.

Hence, the development of techniques that provide a "causal explanation" beyond standard regression or classification while offering some degree of invariance across environments is more pragmatic. As motivated by this idea, a series of methods has been proposed, including those by Peters et al. (2016); Pfister et al. (2018); Rothenhäusler et al. (2018); Heinze-Deml et al. (2018); Gamella and Heinze-Deml (2020); Oberst et al. (2021), which exploit the inherent heterogeneity within data across multiple environments.

Assumption B. (Invariance Assumption) There exists a subset $S^* \subseteq \{1, \ldots, p\}$ of covariate indices (including the empty set) such that

$$P(Y^e | X^e_{S^*}) \quad \text{is the same, for all } e \in \mathcal{E}.$$

This implies that the conditional distribution remains invariant across all environments when conditioning on covariates from S^*.

Peters et al. Peters et al. (2016) explore the concept that "invariance" can infer causal structure under specific conditions and introduce the Invariant Causal Prediction (ICP) approach. They leverage the observation that when considering all direct causes of a target variable, the conditional distribution of the target given these direct causes remains unchanged even when intervening on all other variables except the target itself. A statistical test is conducted to assess if a covariate subset S satisfies the invariance assumption B for the observed environments in $\mathcal{E}$. For more details see Liu et al. (2021). Under the assumption of a structural equation model with Gaussian residuals (Peters et al., 2016), ICP employs the Chow test (Chow, 1960) to identify subsets of true causal variables. Then ICP is capable of uncovering subsets of true causal variables (with some probability).

Now that we have established a foundational understanding of the key concepts surrounding Out-of-Distribution (OOD) generalization and explored the important concepts around it, we are well-prepared to delve into the articles and learn about the ways in which causal structure and representation learning can offer solutions to address this central challenge.

References

Andrews, S., Tsochantaridis, I., and Hofmann, T. (2003). Support vector machines for multiple-instance learning. In *Advances in neural information processing systems*, pages 561–568.

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv: 1607.06450.*

Bengio, Y., Courville, A., and Vincent, P. (2012). Representation learning: A review and new perspectives. *arXiv preprint arXiv: 1206.5538.*

Bühlmann, P. (2018). Invariance, causality and robustness. *arXiv preprint arXiv: 1812.08233.*

Chow, G. C. (1960). Tests of equality between sets of coefficients in two linear regressions. *Econometrica*, 28(3):591–605.

Dietterich, T. G., Lathrop, R. H., and Lozano-Pérez, T. (1997). Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89(1-2):31–71.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Gamella, J. L. and Heinze-Deml, C. (2020). Active invariant causal prediction: Experiment selection through stability. In *Advances in Neural Information Processing Systems*.

Goldberger, J., Gordon, S., and Greenspan, H. (2016). Training deep models with imbalanced and noisy labels. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning-Volume 48*, pages 2070–2079. JMLR. org.

Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. (2016). *Deep learning*, volume 1. MIT Press.

Heinze-Deml, C., Peters, J., and Meinshausen, N. (2018). Invariant causal prediction for nonlinear models. *Journal of Causal Inference.*

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. (2016). beta-vae: Learning basic visual concepts with a constrained variational framework.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 448–456.

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

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings.*

Liu, J., Shen, Z., He, Y., Zhang, X., Xu, R., Yu, H., and Cui, P. (2021). Towards out-of-distribution generalization: A survey. *arXiv preprint arXiv: 2108.13624.*

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

Maron, O. and Lozano-Pérez, T. (1998). A framework for multiple-instance learning. In *Advances in neural information processing systems*, pages 570–576.

Oberst, M., Thams, N., Peters, J., and Sontag, D. (2021). Regularizing towards causal invariance: Linear models with proxies. In *International Conference on Machine Learning.*

Patel, G. and Dolz, J. (2022). Weakly supervised segmentation with cross-modality equivariant constraints. *Medical Image Analysis*, 77:102374.

Pathak, D., Krähenbühl, P., and Darrell, T. (2015). Constrained convolutional neural networks for weakly supervised segmentation. *arXiv preprint arXiv: 1506.03648.*

Peters, J., Bühlmann, P., and Meinshausen, N. (2016). Causal inference by using invariant prediction: identification and confidence intervals. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 78(5):947–1012.

Pfister, N., Bühlmann, P., and Peters, J. (2018). Invariant causal prediction for sequential data. *Journal of the American Statistical Association.*

Reed, S., Lee, H., Anguelov, D., Szegedy, C., Erhan, D., and Hua, W. (2014). Training deep neural networks on noisy labels with bootstrapping. *arXiv preprint arXiv:1412.6596.*

Rothenhäusler, D., Meinshausen, N., Bühlmann, P., and Peters, J. (2018). Anchor regression: Heterogeneous data meets causality. *arXiv preprint arXiv:1801.06229.*

Schölkopf, B., Locatello, F., Bauer, S., Ke, N. R., Kalchbrenner, N., Goyal, A., and Bengio, Y. (2021). Toward causal representation learning. *Proceedings of the IEEE*, 109(5):612–634.

Sukhbaatar, S., Bruna, J., Paluri, M., Bourdev, L., and Fergus, R. (2014). Training convolutional networks with noisy labels. *arXiv preprint arXiv:1406.2080.*

Ulyanov, D., Vedaldi, A., and Lempitsky, V. (2016). Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv: 1607.08022.*

Chapter 2

Structure Learning of Reusable Factor Graphs

by

Amin Mansouri[1], Kartik Ahuja[1], Irina Rish[1], and Yoshua Bengio[1]

(1) Mila, Quebec AI Institute and Université de Montréal

This work has not yet been published but is ready to be submitted, and it serves as the cornerstone for the articles presented in chapters 3 and 4. Chapter 3 draws direct inspiration from the core ideas introduced in this work, effectively reversing the problem's perspective. Additionally, chapter 4 further builds upon this foundation and insights from chapter 3, extending the results to more realistic scenarios. Unlike the current article, which lays down the theoretical groundwork accompanied by evaluations in synthetic environments, chapter 4 focuses on jointly learning the representation and reusable underlying structure of the observations in various real-world tasks. Given its pivotal role in shaping the thesis, this work has been presented as a standalone chapter.

Contributions. I have had a strong interest in tackling the challenge of OOD generalization using invariance principles to learn better structures, as highlighted in my earlier work (Mansouri et al., 2021). This curiosity led me to delve deeper into explicit structure learning within Directed Acyclic Graphs (DAGs). The inception of the notion of Reusable Factor Graphs should be attributed to Kartik Ahuja, who also deserves full credit for the accompanying theoretical findings. When it came to practical aspects such as designing and conducting experiments, Kartik and I collaborated closely to fine-tune our approach. I then took the lead in implementing our ideas, creating the necessary models, overseeing the experimentation process, and performing subsequent analyses. My involvement extended to mathematical derivations establishing connections to the Expectation-Maximization (EM) technique. The resultant article was a joint effort between Kartik and me, with his guidance being a cornerstone of the project's development. It is pivotal to acknowledge the invaluable high-level guidance provided by Irina Rish and Yoshua Bengio throughout the project's progression.

Personal Contributions.

- The inception of the idea of RFG should be credited to Kartik, however, the final version of RFG, and TD-RFG is the result of discussions between me and Kartik
- I implemented and ran all the experiments (including hyperparameter search) with the exception of those resulting in figures 2.6, 2.7
- I produced and analyzed all the plots with the exception of those in figures 2.6, 2.7
- I established the connection to EM and wrote the corresponding section.
- I came up with how to infer the number of factors when it is unknown, using soft Hamming Distance.
- I co-authored the paper together with Kartik.

RÉSUMÉ. La découverte de la structure causale qui sous-tend les données observées a émergé en tant que voie prometteuse pour renforcer les capacités de transfert des modèles d'apprentissage profond, améliorant leur capacité à généraliser vers de nouvelles distributions non observées. Cette étude introduit une approche novatrice appelée Reusable Factor Graphs (RFG), un nouvel algorithme d'apprentissage de structure qui se distingue des méthodes existantes en adoptant une représentation de graphe de facteurs pour les mécanismes causaux, par opposition à l'approche classique des graphes acycliques dirigés (DAG).

L'adoption du graphe de facteurs repose sur des biais inductifs précédemment inexplorés, notamment la réutilisabilité des mécanismes causaux et la sparsité inhérente de leur espace d'entrée. Ces biais, informés à la fois par la causalité et la cognition humaine de haut niveau, constituent le cœur des RFG. L'algorithme proposé exploite l'accès aux représentations causales de la distribution génératrice des données et révèle le potentiel de gains significatifs en efficacité d'échantillonnage grâce à la réutilisation des mécanismes causaux dans un cadre de graphe de facteurs.

À travers des évaluations approfondies sur des ensembles de données synthétiques, ce travail confirme l'efficacité de deux variantes de RFG, offrant une preuve convaincante de sa faisabilité. De plus, l'étude établit des connexions significatives entre l'apprentissage de structure des RFG et l'algorithme EM classique pour le regroupement (clustering).
Mots clés : apprentissage causal, apprentissage de structure, graphes de facteurs, mécanisme causal réutilisable

ABSTRACT. Discovering the causal structure that underlies observed data has emerged as a promising avenue to bolster the transfer capabilities of deep learning models, enhancing their ability to generalize to new and unseen distributions. This study introduces an innovative approach called Reusable Factor Graphs (RFG), a novel structure learning algorithm that distinguishes itself from existing methods by adopting a factor graph representation of causal mechanisms, as opposed to the conventional Directed Acyclic Graph (DAG) approach.

The adoption of the factor graph is rooted in previously unexplored inductive biases, including the reusability of causal mechanisms and the inherent sparsity within their input space. These biases, informed by both causality and higher-level human cognition, constitute the heart of RFG. The proposed algorithm leverages access to causal representations of the data-generating distribution and reveals the potential for significant gains in sample efficiency through the reuse of causal mechanisms within a factor graph framework.

Through comprehensive evaluations on synthetic datasets, this work substantiates the efficacy of two variants of RFGs, offering a compelling proof of concept for its practicality. Furthermore, the study establishes meaningful connections between the structure learning of RFGs and the classical EM algorithm for clustering.
Keywords: causal learning, structure learning, factor graphs, reusable causal mechanism

2.1. Introduction

A subject of rising interest in deep learning theory is the ability of learned models to generalize their performance outside of the distribution on which they were trained. Commonly referred to as Out-of-Distribution (OOD) generalization, this has been the subject of many recent studies that try to approach this challenge from various perspectives. Discovering the causal structure that underlies observed data has emerged as a compelling avenue in enhancing the transfer capabilities of deep learning models, enabling them to generalize effectively to novel and unseen distributions. Causality, as the backbone of relationships among variables, provides a deeper understanding of the underlying mechanisms driving data distributions. Uncovering such causal structures in practice can be decomposed as the learning of independent causal mechanisms along with the learning of invariant features (instead of spurious features) that give rise to the observed data distribution (Parascandolo et al., 2020; Goyal et al., 2020; Goyal and Bengio, 2020; Arjovsky, 2020). By identifying causal relationships, models can distinguish between true causal factors and spurious correlations, thereby capturing the fundamental drivers of data distribution. Such causal relationships encapsulate the invariances that models need to generalize effectively. Causal relationships are expected to remain stable across different environments, enabling models to leverage this stability for improved OOD performance. It is important to emphasize that causal mechanisms govern the relations among some (causal) *latent* variables, and it is only in the correct space that we could learn such parsimonious models; thus, the problem of *causal representation learning* should be considered closely to structure learning, and as a matter of fact, these two inform each other deeply (Ahuja et al., 2022b,a). In this work, similar to the prevalent approach in the structure learning literature, we assume access to such representations and focus on finding efficient ways of structure learning.

The realm of causal structure learning has established its significance across various scientific domains, including genetics, biology, and economics (Koller and Friedman, 2009; Peters et al., 2017; Sachs et al., 2005; Pearl, 2009). Bayesian networks (BNs), characterized by directed acyclic graphs (DAGs), are influential models renowned for their interpretability and computational feasibility. Causal graphical models (CGMs) (Peters et al., 2016) enable the exploration of interventional queries, empowering us to probe the consequences of external interventions on variables. In the context of machine learning models, we believe models with the ability to comprehend and reason about the dynamics of entities would be expected to exhibit improved robustness and generalization in novel scenarios.

Causal structure learning entails the inference of causal graphical models, frequently taking the form of directed acyclic graphs (DAGs). This learning is complicated by the fact that multiple causal DAG models can yield identical observational distributions. While interventions help reduce this ambiguity, their effectiveness depends on the availability of interventional data. Various methods have been proposed for causal structure learning, with certain algorithms assuming causal sufficiency in the absence of latent confounders.

This study introduces an innovative approach named Reusable Factor Graphs (RFG), a novel structure learning algorithm that diverges from prevailing methods by adopting a factor graph representation of causal mechanisms, in contrast to the conventional Directed Acyclic Graph (DAG) paradigm. The adoption of the factor graph is rooted in previously unexplored inductive biases, including the reusability of causal mechanisms and the inherent sparsity within their input space. These biases, informed by both causality and higher-level human cognition, constitute the heart of RFG. The proposed algorithm leverages access to causal representations of the data-generating distribution and reveals the potential for significant gains in sample efficiency through the reuse of causal mechanisms within a factor graph framework.

We combine ideas from Kahneman (2011), the “Consciousness Prior” (Bengio, 2019), and its relation to Global Workspace Theory (GST) (Baars, 2005), a hypothesis regarding the working mechanism of the human brain in reasoning tasks. According to Kahneman (2011) system one mode of thinking or thinking fast is intuitive and instinctive. On the other hand, system two, or thinking slow, is rational and based on logic, and indecisive. The former is the domain in which current Deep Learning is good, and the latter is what motivates this work. These works conjecture that conscious thought is conceived based on the conscious thought at the previous moment and an attention mechanism that acts as a bottleneck on the output of all sensory modules. The probabilistic and logical nature of this mode of thinking inspires us to explore structure learning in the form of a particular modification of the graphical models, i.e., a factor graph, and its compositional nature motivates thinking in terms of independent mechanisms. The consciousness prior (Bengio, 2019) brings the importance of two crucial inductive biases to our attention—reusability and sparsity of interactions. These hypotheses and intuitions can be modeled neatly by a sparse factor graph representation. A factor graph is a bipartite graph, where on one side, we keep the nodes corresponding to *factors* or *mechanisms*, and on one side, we keep the *subsets of random variables* that our factor nodes can operate on to generate the observations. However, we enforce the number of factor nodes to be small (enforcing *reusability*), and the size of each subset of random variables be limited too (enforcing *sparsity*, i.e., resembling the attention bottleneck in GST). Our goal is to learn the structure of this bipartite graph, i.e., both the parameters of the mechanisms and the connections (edges) between these mechanisms and the causal variables (i.e., the variables a mechanism takes as input). We do not consider DAGs as they have been explored Abbeel et al. (2006); Brouillard et al. (2020); Lachapelle et al. (2019); Zheng et al. (2018) since they do not allow the inductive bias of *reusability* to be exploited efficiently and explicitly, meaning that while a parent-child relation in a DAG could reappear in many other edges of a causal DAG, settings such as earlier methods learn all of them from scratch and do not exploit the reusability arising from decomposing the causal structure into repeating factors.

This approach enables the learning of reusable mechanisms, and Bengio et al. (2019) suggests that through slight modifications our method could be capable of translating the RFG to a causal DAG by learning the causal directions. We establish the theory and carry out experiments for learning a factor graph comprising a small number of factors operating on small subsets of random variables. The former encourages reusability of mechanisms (as opposed to learning a new mechanism for every edge in a causal DAG), and the latter encourages sparsity (a typical causal mechanism most likely cannot manipulate a large set of elements in an environment simultaneously). We establish the connection of our proposed algorithm with the Expectation-Maximization algorithm in appendix A.3.

2.2. Related works

Structure learning. Abbeel et al. (2006) is one of the most important works on learning factor graphs. Most of the existing works on structure learning in factor graphs are discrete optimization approaches. There has been a lot of work recently on structure learning in DAGs using continuous optimization-based approaches for linear settings Zheng et al. (2018) and non-linear settings exploiting neural networks Lachapelle et al. (2019). Factor graphs offer many advantages that standard DAGs-based models may not. To the best of our knowledge, continuous optimization-based approaches for structure learning of factor graphs have not been extensively explored.

Importance of reusability. Ours is the first work to employ and exploit reusability constraints for structure learning in factor graphs. These constraints allow us to approximate the partition function. If the reusability degree is a constant, then the number of distinct terms in the partition function grows exponentially in the reusability degree which can be a small value in many problems of interest (see section 2.3 for the definitions of reusability degree and the partition function). Existing ideas on reusing parameters in structure learning such as dynamic Bayesian networks hard code the parameter sharing across time steps. In contrast, in the current setup, we learn which subsets of variables share a common factor.

Other reusability based representations. Efficient representation of factor graphs was first presented in the form of plated factor graphs Obermeyer et al. (2019), which expresses reusability in the form of plates; the representation in their paper and our work are not the same. In Obermeyer et al. (2019) the authors showed the advantage of such a representation in terms of inference but the advantages from the structure learning point of view were not explored.

2.3. Methodology

We start by describing the limitations of structural causal models (SCMs) Goyal and Bengio (2020). SCMs can be inflexible in their ability to capture independent knowledge factors thus making it hard to learn the independent causal mechanisms. Moreover, the problem is exacerbated to a great degree if each causal mechanism is composed of some simpler rules that are generic and are reused in other mechanisms. In Goyal and Bengio (2020), the authors propose that building factor graphs can alleviate some of these key concerns.

We first discuss how the standard representation of factor graphs has certain important limitations that can also prohibit it from learning reusable generic rules. Consider a standard factor graph, one side of vertices corresponding to factors $\{f_1, \dots, f_M\}$ and the other side corresponds to the random variables $\{X_1,...,X_N\}$. The joint probability distribution $P(X_1, \dots ,X_N)$ can be written as a factorization over factors in $\{f_1, \dots, f_M\}$ as follows.

$$P(X_1, \dots ,X_N) = \frac{1}{Z}\Pi_j f_j(S_j)$$

where $S_j \subseteq \{X_1, \dots ,X_N\}$ and Z is a normalization constant. S_j contains all the random variables that share an edge with f_j. In Figure 2.1, we show the example of a factor graph with a distribution $P(X_1, \dots, X_4) = f_1(X_1)f_2(X_1,X_2)f_3(X_2,X_3)f_4(X_2,X_3)$. The bipartite graph in Figure 2.1 looks at each function f_i as a separate vertex even if some of the functions are the same. As a result, the graphical representation is ineffective in capturing the reusability of the factors. For instance, if the factors f_2 and f_3 are equal, then it can be easier to learn such a shared rule from the data. Moreover, learning such shared factors can be useful from the point of view of identifying some generic mechanisms that are used in

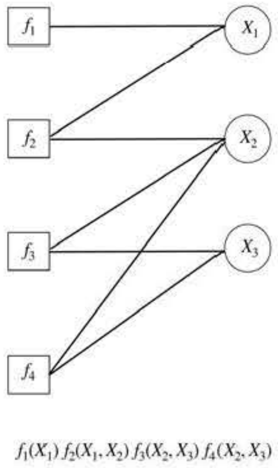


Fig. 2.1. Example of a standard factor graph with factorized joint distribution $P(X_1,X_2,X_3) = f_1(X_1)f_2(X_1,X_2)f_3(X_2,X_3)f_4(X_2,X_3)$.

data generation. Keeping this in mind, we propose an alternate way to represent the factor graphs. We construct a bipartite graph, where on one side each vertex corresponds to a unique function and on the other side we have the subsets S_j on which the factor operates. Therefore, we can rewrite the distribution to highlight this fact.

$$P(X_1,\ldots,X_N) = \frac{1}{Z}\Pi_j \Pi_{S_i \in \mathsf{N}(j)} f_j(S_j)$$

In the above factorization, each function f_j is unique and the set $\mathsf{N}(j)$ are the neighbors corresponding to the sets S_j on which the factor operates. In Figure 2.2, we present the example in Figure 2.1 under the new representation taking into account the fact that $f_2 = f_3$.

What can we directly infer from the new factor graph representation? The number of vertices on the factor side expresses how many distinct functions need to be learned. The number of neighbors of each factor vertex describes how many times the factor is used. The number of neighbors of each vertex on the right side describes how many times the same subset of variables appears in the factorization. The maximum number of nodes inside a vertex on the variable side describes the sparsity in the graph.

2.3.1. Structure learning of reusable factor graphs

We described some of the important advantages that the new representation brings. What are the advantages of the above representation from the perspective of structure learning? The assumptions made on the graph representation form the inductive bias for structure learning. For instance, a graph is directed acyclic, graph has a bounded degree form different

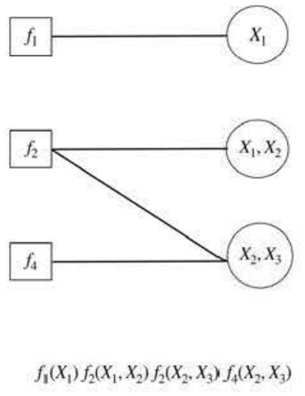


Fig. 2.2. Example of bipartite graph of Figure 2.1 under the new representation taking into account the fact that $f_2 = f_3$.

types of inductive biases. The representation proposed above can be used to encourage the reusability of factors. We should try to learn a graph with three properties: a) the number of factor vertices is small, b) allows for some factor vertices to have a high degree thus encouraging reusability, and c) the maximum size of the subset S_j on the variable side should be small.

Each data sample $\boldsymbol{X} = [X_1, \dots X_p]$ is drawn i.i.d. from a distribution P. Suppose we have m factor vertices (m is a hyperparameter) denoted by set $\mathcal{F} = \{1, \dots m\}$. We model the i^{th} factor vertex using a neural network f_{θ_i} with parameters θ_i. We restrict the maximum cardinality of a variable vertex S_j to be d. The vertices on the variable side correspond to all possible subsets S_j of $\boldsymbol{X}$ of cardinality up to d, where d controls the sparsity in the factor graph. Therefore, the total number of variable vertices are $Q = \sum_{k=1}^{d} \binom{p}{k}$, which grows as $\mathcal{O}(p^d)$. We index the vertices to form a set $\mathcal{V}$ such that from the index j of the vertex we can identify the subset S_j of the random variables. We treat the adjacency matrix M of the bipartite graph as a random, where each entry M_{ij} of the matrix is a Bernoulli random variable with success probability $\sigma(\lambda_{ij})$, where σ is a sigmoid function and λ_{ij} is a parameter that we learn. We write the parameters for all the edges in the form of a matrix $\Lambda \in \mathbb{R}^{M \times Q}$. We can write the loss function that we want to optimize under this parameterization as follows. For each neural network i with parameters θ_i, we assign the dimension of the input that it operates on. For simplicity, let us assume that for each input size there are q neural networks. Therefore, $q * d = m$.

$$L(\{\theta_j\}_{j\in\mathcal{F}}, \Lambda) = -\mathbb{E}_{M\sim\sigma(\Lambda)}\mathbb{E}_{\boldsymbol{X}\sim P}\Big[\sum_{j\in\mathcal{F}}\sum_{k\in\mathcal{V}} \log(f_{\theta_j}(S_k))M_{jk}I_{jk} - \log(Z(\{\theta_j\}_{j\in\mathcal{F}}, \Lambda)\Big] \quad (2.3.1)$$

where $Z(\{\theta_j\}_{j\in\mathcal{F}}, \Lambda)$ is the normalization constant and I_{jk} is an indicator function which takes a value 1 is the size of S_k matches the input dimension that neural network i can take and otherwise it is zero. Note that optimization of Λ can be handled using Gumbel-softmax reparameterization trick Brouillard et al. (2020). One main challenge in trying to minimize the above loss is how to compute $Z(\{\theta_j\}_{j\in\mathcal{F}}, \Lambda)$. One possible approach is to use standard Monte-Carlo integration. The approach in Abbeel et al. (2006) focused on learning factor graphs used a way to factorize where we do not need to compute the normalization constant. However, the approach in Abbeel et al. (2006) is only applicable to discrete random variables. Moving forward, we can apply the above ideas to directed factor graphs Frey (2012) in a similar fashion. Other than parameterizing the adjacency matrix with parameters λ_{ij} and taking the expectation above w.r.t. $M \sim \sigma(\Lambda)$, we could also use a softmax over the outputs of the neural nets modeling the factors. This way, the softmax operation encourages competition among the neural nets and naturally provides a normalized set of weights that could replace M_{jk} in the above equation. We denote this method in the experiment section (as well as plot legends) by the *softmax* method, and use the Λ method to refer to the former.

2.3.2. Equivalence classes representation

In order to avoid choosing subsets of variables that are incompatible with the *signature* of a factor (which is enforced by the indicator function I_{jk} in Equation 2.3.1), we can break down the set of possible unique factors $\mathcal{F}$ into subsets of factors that share the same signature. Concretely, if we assume that d is the maximum number of variables in any given factor, we can define $\mathcal{F} = \bigcup_{k=1}^{d} \mathcal{F}_d$, where $\mathcal{F}$ is the set of unique factors that have k inputs; we also call $Q_k = |\mathcal{F}_k|$. One consequence is that we can also break down the set of subsets of variables (i.e. the vertices on the left-hand side of Figure 2.2) into subsets of *exactly* k variables; we call this set $\mathcal{S}_k = \{S \in 2^{\mathcal{V}} \mid |S| = k\}$, and $M_k = |\mathcal{S}_k| = \binom{p}{k}$. The adjacency matrix is now parametrized by $\Lambda \in \mathbb{R}^{M_1\times Q_1} \times \ldots \times \mathbb{R}^{M_d\times Q_d}$, as opposed to an $M \times Q = \sum_k M_k \times \sum_k Q_k$ matrix, reducing significantly the number of parameters. The objective from Equation 2.3.1 can now be written as

$$\mathcal{L}(\{\theta_j\}_{j\in\mathcal{F}}, \Lambda) = -\mathbb{E}_{M\sim\sigma(\Lambda)}\mathbb{E}_{\boldsymbol{X}\sim P}\left[\sum_{k=1}^{d}\sum_{j\in\mathcal{F}_k}\sum_{i\in\mathcal{S}_k} \log f_{\theta_j}(S_i)M_{ji}^{(k)} - \log Z(\{\theta_j\}_{j\in\mathcal{F}}, M)\right] \quad (2.3.2)$$

2.3.3. Partition computation advantage for reusable factor graphs

Here we briefly try to lay down some intuition that can help explain why the partition function computation can be tractable in reusable factor graphs with a larger reusability degree than the standard factor graphs. Consider a very simple setting where there are K factors that take as input a scalar binary valued variable. The partition function is given as $Z = \sum_{\boldsymbol{X}} \Pi_{m=1}^{K} f_m(X_i)$ The above summation involves 2^K terms. However, let us consider the case when all the functions f_m were identical. In such a case, we only need to compute the product function over K distinct sequences and re-express it as $Z = \sum_{\boldsymbol{X}} \binom{K}{c} f(0)^c f(1)^{K-c}$. If we were to construct an estimate of the partition function using Monte Carlo integration, then the integral would require fewer random samples to estimate the Z.

We now describe a more general case of the setting described above. Let us assume that the size of the input to each factor is the same and is equal to d. Also, let us assume that no two subsets S_j and S_k intersect. The total number of subsets is $\frac{p}{d}$. In addition, we say that each S_j is only associated with one factor f_i. Therefore, the total number of factors is $m = \frac{p}{d}$. Suppose there are only r distinct factors $\{g_1, \cdots, g_r\}$. Suppose g_i is repeated m_i times, which implies $\sum_{i=1}^{r} m_i = m$. The total number of possible values g_i can take assuming binary valued input is $\tilde{d} = 2^d$. The total number of possible values the product of factors g_i can take given that g_i are multiplied m_i times is $m_i + \binom{\tilde{d}-1}{\tilde{d}-1}$.

$$m_i + \binom{\tilde{d}-1}{\tilde{d}-1} \leq \Big(\frac{e(m_i + \tilde{d} - 1)}{\tilde{d} - 1}\Big)^{\tilde{d}-1} \tag{2.3.3}$$

The total number of possible terms across the factors are

$$\Pi_{i=1}^{r} \Big(\frac{e(m_i + \tilde{d} - 1)}{\tilde{d} - 1}\Big)^{\tilde{d}-1} \approx \Big(\frac{e(\frac{m}{r} + \tilde{d} - 1)}{\tilde{d} - 1}\Big)^{r\tilde{d}-r} \tag{2.3.4}$$

2.4. Time-directed reusable factor graphs

In this work, we aim to show as a proof of concept why the above framework is powerful, and that the above formulation can lead to more effective sample complexity in structure learning. Therefore, we will make additional assumptions that will allow us to simplify the objective defined in equation 2.3.2 further. In this section, we discuss a class of reusable factor graphs, which we refer to as time-directed reusable factor graphs.

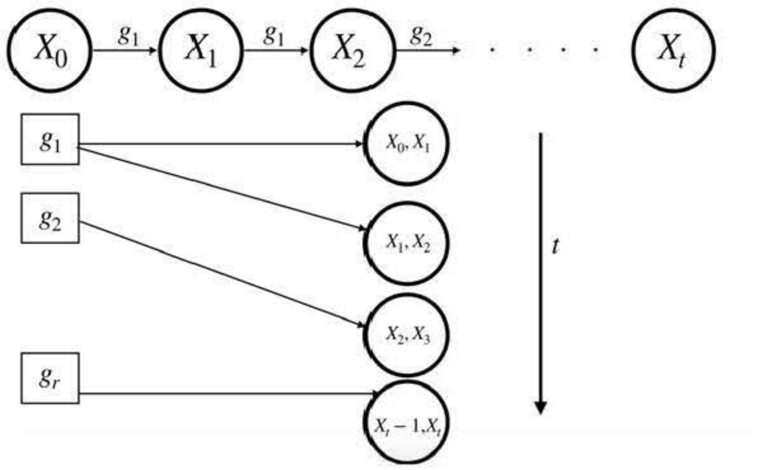


Fig. 2.3. Illustrating time-directed reusable factor graph

2.4.1. Reusable mechanisms in a Markov chain

Suppose $\boldsymbol{X} = [X_1, \dots, X_p]$ can be factorized as a Markov chain.

$$\mathbb{P}(\boldsymbol{X}) = \mathbb{P}(X_1)\Pi_{i=1}^{p-1}\mathbb{P}(X_{i+1}|X_i) \tag{2.4.1}$$

We can write the factorization in factor graph notation as follows.

$$\mathbb{P}(\boldsymbol{X}) = \mathbb{P}(X_1)\Pi_{i=1}^{p-1}\mathbb{P}(X_{i+1}|X_i) = f_1(X_1)\Pi_{i=1}^{p-1}f_{i+1}(X_i, X_{i+1}) \tag{2.4.2}$$

If the Markov chain was stationary then all the factors $f_{i+1}(X_i, X_{i+1})$ would be the same say $f(X_i, X_{i+1})$. We consider the setting where each of the factors $f_{i+1}(X_i, X_{i+1})$ are drawn from the set of reusable mechanisms $\{g_1, \dots, g_r\}$, i.e., $\forall i \in \{1, \dots, p\}$, $f_i \in \{g_1, \dots, g_r\}$. In Figure 2.3 we illustrate an example of a reusable mechanism-based Markov chain.

Learning. Here, we describe how we can set up a simple continuous relaxation of the likelihood objective (similar to the one described in the previous section). Assume that we have m neural networks $\{f_{\theta_1}, \dots, f_{\theta_m}\}$. Let Θ be the set of parameters characterizing the different neural networks. Each neural network operates on the pair X_i, X_{i+1} and models a conditional density given as $\tilde{f}_{\theta_j}(X_i, X_{i+1}) = \frac{f_{\theta_j}(X_i, X_{i+1})}{\sum_x f_{\theta_j}(X_i, X_{i+1}=x)}$ (we assume discrete support for each random variable to simplify the normalization). Recall that M is the adjacency matrix with the number of rows equal to the number of factors m and the number of columns equal to p. Observe that the number of parameters in M grows with the length of the chain and we will describe how to tackle this issue in a bit. We define the likelihood next.

$$L(\{\theta_i\}_{i=1}^m, M, \boldsymbol{X}) = \Pi_{i=1}^{p-1}\Pi_{j=1}^m\tilde{f}_{\theta_j}(X_i, X_{i+1})^{M_{ji}} \tag{2.4.3}$$

In the above likelihood, we assume that each column of the matrix M adds up to 1. Define a constrained maximization of the likelihood as follows

$$\begin{aligned} &\max_{\{\theta_i\}_{i=1}^m, M} \mathbb{E}\big[\log\big[L(\{\theta_i\}_{i=1}^m, M, \boldsymbol{X})\big]\big] \\ &\text{s.t.} \forall i,j \; M_{ij} \in \{0,1\}, \sum_{i=1}^{m} M_{ij} = 1 \end{aligned} \tag{2.4.4}$$

We define a continuous relaxation of the above objective as follows. We model M as a softmax over a real-valued matrix Λ with softmax taken over each column of the matrix, which ensures that each column adds up to one and each entry is positive and less than one.

$$\max_{\{\theta_i\}_{i=1}^m, \mathsf{softmax}(\Lambda)} \mathbb{E}\big[\log\big[L(\{\theta_i\}_{i=1}^m, \mathsf{softmax}(\Lambda), \boldsymbol{X})\big]\big] \tag{2.4.5}$$

In Figure 2.4, we show an example to illustrate the graph being learned.

Assumption 1. *$\exists\, \theta' \in \Theta$ such that for all $k \in \{1,\cdots,r\}$ $\tilde{f}_{\theta'}(X_i, X_{i+1}) = g_k(X_i,X_{i+1})$*

Proposition 1. *If $m \geq r$ and Assumption 1 holds, then from the solution of equation equation 2.4.5, we can exactly recover the true factors $\{g_1, \cdots, g_r\}$.*

Proof. Observe that solving equation 2.4.5 and equation 2.4.4 are equivalent (from the solutions of one problem we can recover all the solutions to the other problem). Also, note that the optimal value of the objective in equation 2.4.5 and equation 2.4.4 are the same (due to the linearity of the objective in Λ and M). Since Assumption 1 holds and $m \geq r$ we can construct one of the optimal solutions. Select θ_j values for the different NNs such that there is at least one NN $\tilde{f}_j$ that imitates g_j. Select Λ_{ij}^s such that at each time step i assigns the entire weight to the factor that is used at that time to generate the data. As a result, the term inside equation equation 2.4.5 becomes the exact distribution that generates the data. We need to show the other side of the result that if the maximum is achieved, we can construct the factors $\{g_1, \ldots g_r\}$. We argue that we can below.

$$\begin{aligned} &\mathbb{E}\big[\log\big[L(\{\theta_i\}_{i=1}^m, \mathsf{softmax}(\Lambda), \boldsymbol{X})\big]\big] = \mathbb{E}\big[\sum_{i=1}^{p-1}\sum_{j=1}^{m} \Lambda_{ij}^s \log\big(\tilde{f}_{\theta_j}(X_i, X_{i+1})\big)\big] \\ &\sum_{i=1}^{p-1}\sum_{j=1}^{m} \Lambda_{ij}^s \mathbb{E}\big[\log\big(\tilde{f}_{\theta_j}(X_i, X_{i+1})\big)\big] = \sum_{i=1}^{p-1}\sum_{j=1}^{m} \Lambda_{ij}^s L_{ij}^*(\theta_j) \end{aligned} \tag{2.4.6}$$

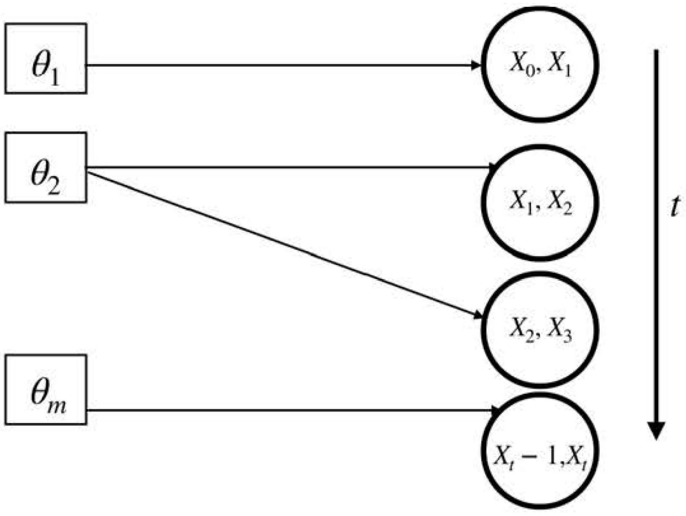


Fig. 2.4. Illustrating time-directed reusable factor graph parameterized by $\{\theta_1, \dots, \theta_m\}$

where $L^*_{ij}(\theta_j) = \mathbb{E}\big[\log\big(\tilde{f}_{\theta_j}(X_i, X_{i+1})\big)\big]$. Consider one of the optimal set of values for $\{\theta^*_j\}_{j=1}^m$, define the set of indices that are in the set $\mathcal{S} = \arg\max_j L^*_{ij}(\theta_j)$. The optimal values of Λ^s_{ij} have a zero weight on all the indices outside the set $\mathcal{S}$. Since all the values Λ^s_{ij} corresponding to $j \in \mathcal{S}$, without loss of generality consider the solutions in which the entire weight Λ^s_{ij} is assigned to one of the elements in $\mathcal{S}$. Each term in the optimal solution corresponds to a valid conditional probability distribution. At the optimum, the distribution learned and the true distribution are the same. By marginalizing the LHS and RHS of the distributions, we can show that the factors are exactly recovered.

□

Define log likelihood for each time step as $L_{ji} = \mathbb{E}[\log(\tilde{f}_{\theta_j}(X_i, X_{i+1}))]$. For each time step i, L_{ji} is the expected log likelihood that we observe from using NN parametrized by θ_j.
Proposition 2. *Suppose we fix $\{\theta_j\}_{j=1}^m$, then the solution to optimal M in equation equation 2.4.4 is simply the row corresponding to the maximum likelihood, i.e., let $j^* = \arg\max_{j\in\{1,\dots,m\}} L_{ji}$, $M_{j^*i} = 1$ and for all $j \neq j^*$, $M_{ji} = 0$.*

Proof. Follows from the linearity of the likelihood in M. The same argument as shown in the proof of the previous proposition. □

Inferring mechanisms. Suppose we are given more samples from the chain $\{X_{p+1}, \dots, X_q\}$ and our goal is to infer what are the mechanisms that cause the transitions. We parametrize the likelihood only in terms of the adjacency matrix $\tilde{M}$ and assume that we know the reusable mechanisms $\{g_1, \dots, g_r\}$. If we know these mechanisms and we only want to learn what the adjacency matrix is we can write the likelihood as follows

$$L(\tilde{M}, \{X_{p+1}, \dots, X_q\}) = \mathbb{P}(X_{p+1})\Pi_{i=1}^{p-1}\Pi_{j=1}^{r}\left[g_j(X_i, X_{i+1})\right]^{\tilde{M}_{ji}} \tag{2.4.7}$$

In the above, it is assumed that the columns of $\tilde{M}$ add up to one. Consider the i^{th} column of $\tilde{M}$. Pick a $j^* \in \arg\max g_{ji}(X_i, X_{i+1})$ and set $\tilde{M}^*_{j^*i} = 1$ and $\tilde{M}^*_{ji} = 0$ for all $j \neq j^*$, then $\tilde{M}^*$ maximizes the likelihood above.

2.5. Experiments

2.5.1. A simple illustrative example

In this section, we provide an illustrative example on which we carry out the experiments. Recall $\boldsymbol{X} = [X_1, \dots, X_p]$. Suppose there are two functions $a : \mathbb{R} \to \mathbb{R}$ and $b : \mathbb{R} \to \mathbb{R}$. We will assume that one of these two mechanisms a or b that are used generate X_k from X_{k-1}. For a $k \geq 1$,

$$\begin{gathered} X_k \leftarrow a(X_{k-1}) + N_k \\ \text{or} \\ X_k \leftarrow b(X_{k-1}) + N_k \end{gathered} \tag{2.5.1}$$

In Figure 2.5, we illustrate the above model. We will use extra structural knowledge of the above model to simplify the factor graph learning. Assume that the model knows X_k generates X_{k+1}. In addition to the above structural assumptions, we will also assume that the learner knows that X_k and X_{k-1} are related through additive noise. Therefore, we can simplify the learning further for this example and say that instead of learning the conditional probability distributions, we can try to learn the mean of the conditional distribution. We use the extra knowledge in this setting to simplify the objective in equation equation 2.3.1 as follows. We use each neural network f_{θ_i} to model the conditional expectation $\mathbb{E}[X_k|X_{k-1}]$. Recall $\Lambda \in \mathbb{R}^{M \times p}$, which was used to model the i.i.d. Bernoulli random variables in the adjacency matrix. Since we know that each variable X_k is associated with exactly one neural network, we need to make sure that the sum of the probabilities adds up to one for each row and we use a softmax function to ensure that such a constraint is met.

$$L(\{\theta_j\}_{j=1}^{M}, \Lambda) = \sum_{i=1}^{p}\sum_{j=1}^{M}\mathbb{E}_X[(X_i - f_{\theta_j}(X_{i-1}))^2]\mathsf{softmax}(\Lambda[:,i])[j] \tag{2.5.2}$$

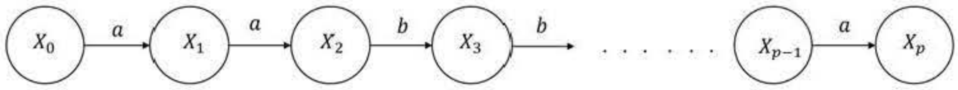


Fig. 2.5. Illustrative example: Markov model

where $\Lambda[: ,i]$ is the i^{th} column of the matrix Λ and $\mathsf{softmax}(\Lambda[: ,i])$ is the probability vector corresponding to it.

Now, we describe the results of experiments on the illustrative model we described. We assume a linear generative model given as

$$X_k = A_i X_{k-1} + N_k \tag{2.5.3}$$

where A_i is i^{th} generating mechanism and $X_k \in \mathbb{R}^u$. The set of all the models that are reused is $\{A_i\}_{i=1}^r$. Our goal is to minimize the objective in equation equation 2.5.2 and learn the matrices A_i. We also need to correctly identify which matrix is associated with which variable X_k. We sample A_i to be a random unitary matrix and N_k is a Gaussian noise vector with unit variance. Indicator vector $\boldsymbol{I}$ dictates which unitary matrix is associated with which variable, i.e. X_k maps to the unitary matrix $A_{\boldsymbol{I}_k}$.

We give an example of the case when the length of the chain $p = 10$, the dimension of each X_k $u = 2$, and the number of reusable factors $r = 2$ and number of samples is 100.

$$A_1 = \begin{bmatrix} -0.325 & 0.945 \\ -0.945 & -0.325 \end{bmatrix}, A_2 = \begin{bmatrix} 0.996 & 0.086 \\ 0.086 & -0.996 \end{bmatrix} \tag{2.5.4}$$

$\boldsymbol{I} = [2,1,2,1,1,2,1,2,\mathbf{1}]$

Case 1. We first start with the case when the number of matrices to be learned (use a one-layer neural network for this purpose) exactly match the number of true distinct matrices, which is two in the above case. The estimated outputs are

$$\hat{A}_1 = \begin{bmatrix} -0.318 & 0.941 \\ -0.941 & -0.328 \end{bmatrix}, \hat{A}_2 = \begin{bmatrix} 0.985 & 0.094 \\ 0.078 & -0.991 \end{bmatrix} \tag{2.5.5}$$

$\hat{\boldsymbol{I}} = [2,1,2,1,1,2,1,2,\mathbf{1}]$

Case 2. Next, we move to the case when the number of matrices to be learned (we set them to four) is greater than the number of generating matrices that are distinct. The estimated outputs are

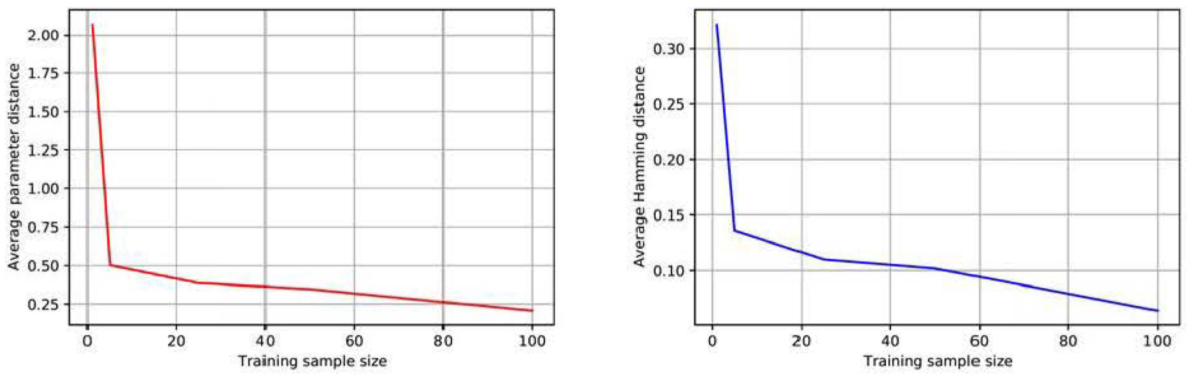


Fig. 2.6. Comparison of the method as a function of the number of samples. $p = 10$, $r = 2$, $u = 2$

$$\hat{A}_1 = \begin{bmatrix} -0.304 & 0.932 \\ -0.929 & -0.335 \end{bmatrix}, \hat{A}_2 = \begin{bmatrix} 0.985 & 0.094 \\ 0.078 & -0.991 \end{bmatrix} \tag{2.5.6}$$

$$\hat{A}_3 = \begin{bmatrix} -0.025 & 0.751 \\ -0.709 & -0.478 \end{bmatrix}, \hat{A}_4 = \begin{bmatrix} -0.319 & 0.942 \\ -0.041 & -0.327 \end{bmatrix} \tag{2.5.7}$$

$\hat{\boldsymbol{I}}$ = [2,4,2,4,4,2,4,2,1] In the above cases, we saw that the proposed objective is almost perfectly able to recover the structure, i.e., the matrices and where in the chain they operate. For the same setting, we averaged the results over fifty trials and plotted the performance as a function of the training sample size. We measure the performance in terms of the parameter distance – the distance between the estimated matrices and the average Hamming distance between the indicator vectors. In Figures 2.6 ,2.7, we show the plots showing the performance of the proposed approach as a function of the number of training samples.

Given the initial success in the above example, below we scale up the experiment to very long chains and high dimensions of X. As we want to showcase the benefit of reusability, we will use only 1 sample in the training set for which the Markov Chain Length (L) varies in [20,40,60,80,100,200,400,600,800,1000]. We also vary the complexity of the task by letting $d \in$ [20,40,60,80,100]. We will show how increasingly longer chains will exploit the reusability of the single sample to both recover the underlying data generation parameters, as well as the indices $\boldsymbol{I}$ of where the mechanisms operate along the chain. In this section we let the number of matrices to be learned exactly matches the number of true distinct matrices. We will see in the following sections how varying these numbers affects the learned parameters. For further training details please refer to appendix A.

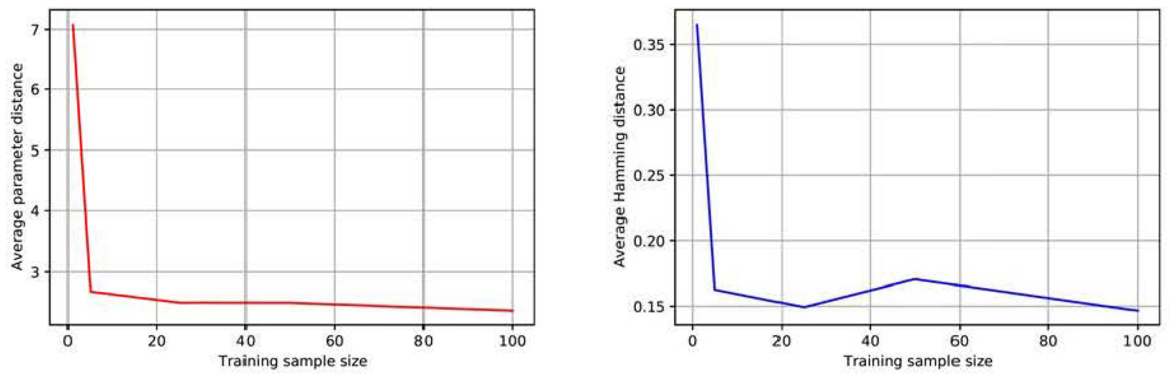


Fig. 2.7. Comparison of the method as a function of the number of samples. $p = 25$, $r = 2$, $u = 5$

Figures 2.8, 2.9 show the average parameter distance and test Hamming distance over the Markov chain length as metrics for the performance of our structure learning algorithm. For training Hamming distances please see the appendix A. We can clearly observe the benefit of reusability when the parameter and Hamming distances both diminish as the chain length increases and we have only access to a single sample.

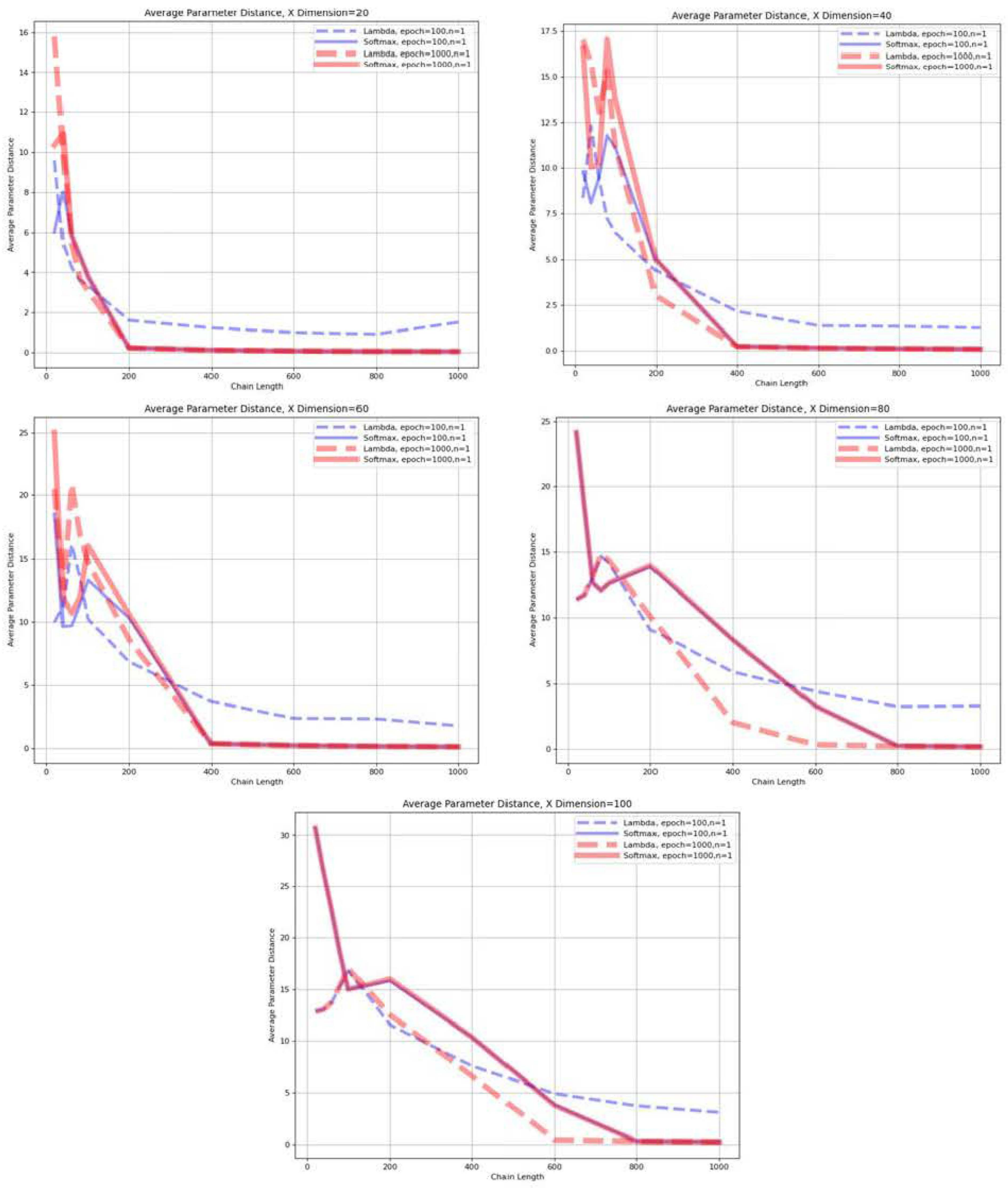


Fig. 2.8. Average parameter distance between the learned models and the ground truth matrices vs. the chain length. Each plot corresponds to some $d \in [20,40,60,80,100]$.

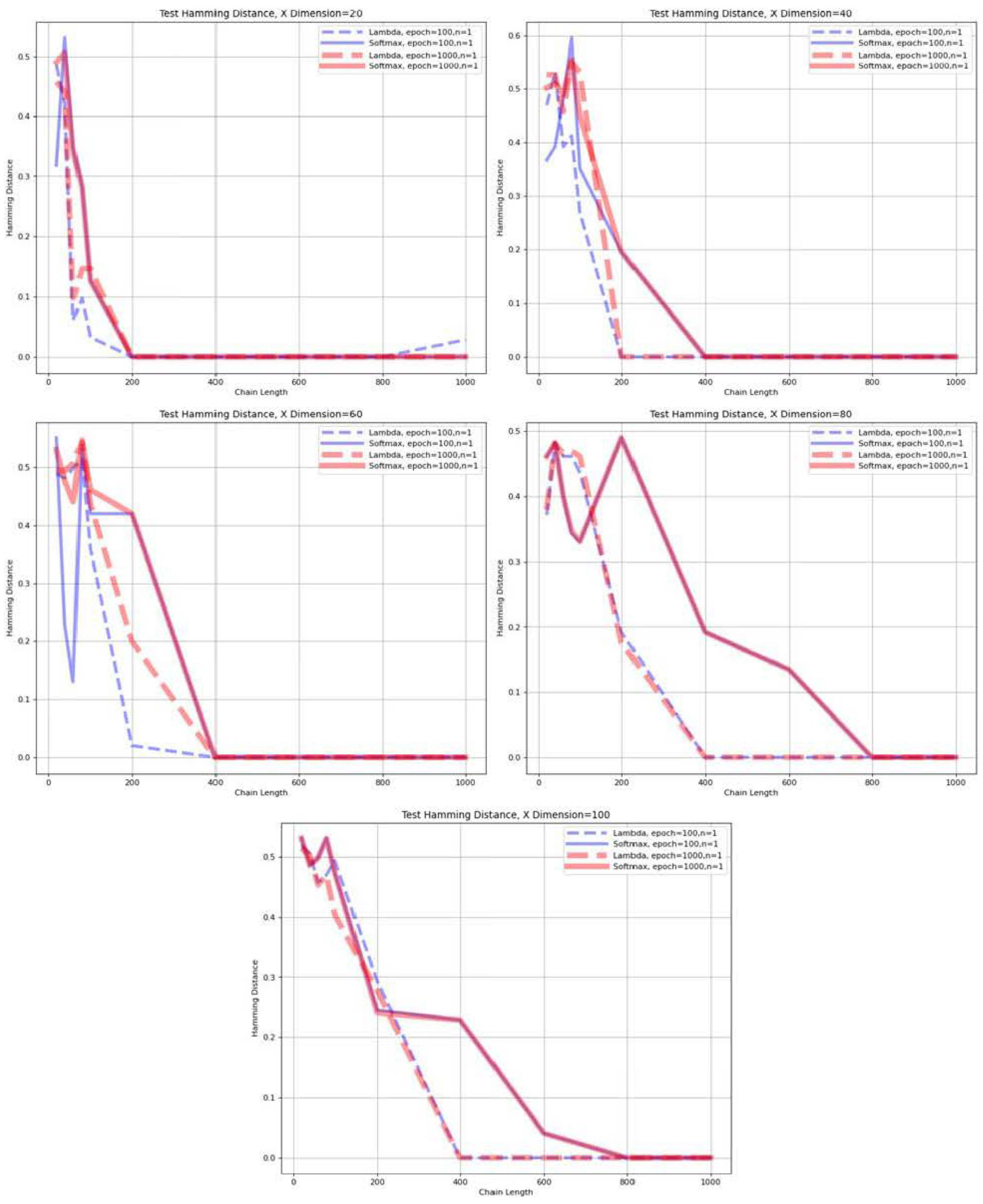


Fig. 2.9. Test Hamming distance vs. the chain length. Each plot corresponds to some $d \in [20,40,60,80,100]$.

2.6. Unknown number of reusable factors

In this section, we observe how learned models behave in a setting where the number of ground truth factors is fixed ($r = 6$), but that number is unknown to the model, therefore we have to train with varying numbers of neural nets modeling the factors. We will explore the behavior of the learned models, and study if there exists a way to infer the correct number of neural nets to be used for modeling the reusable structure. To make sure each factor appears with enough frequency in the chain, we set the chain length to $L = 100$ in the generation process, and keep the dimension low at $d = 2$. For the training dataset, we sample 100 chains of length L. Noise standard deviation σ equals to 15% of the elements of $\|AX\|$. We only use the Λ matrix approach here. When we employ 9 neural nets when there are actually 6 factors, figures 2.10, 2.11 that show the pairwise distances of neural networks and ground truth factors, clearly show that for each factor, there is exactly one neural net that perfectly models them. Figures 2.12, 2.13 reflect the situation where we have fewer neural nets to model all the ground truth factors, and as is expected it should not be possible, which is confirmed by the pairwise parameter distances confirming that no neural net perfectly models any factor. Instead, they all converge to some mixture of the factors to minimize the objective as much as possible. Figures 2.14, 2.15 show the successful recovery of all factors uniquely by each neural net when there are as many neural nets available for modeling as there are ground truth factors. In the next section, we will propose a metric that will assist us in finding the correct number of factors r.

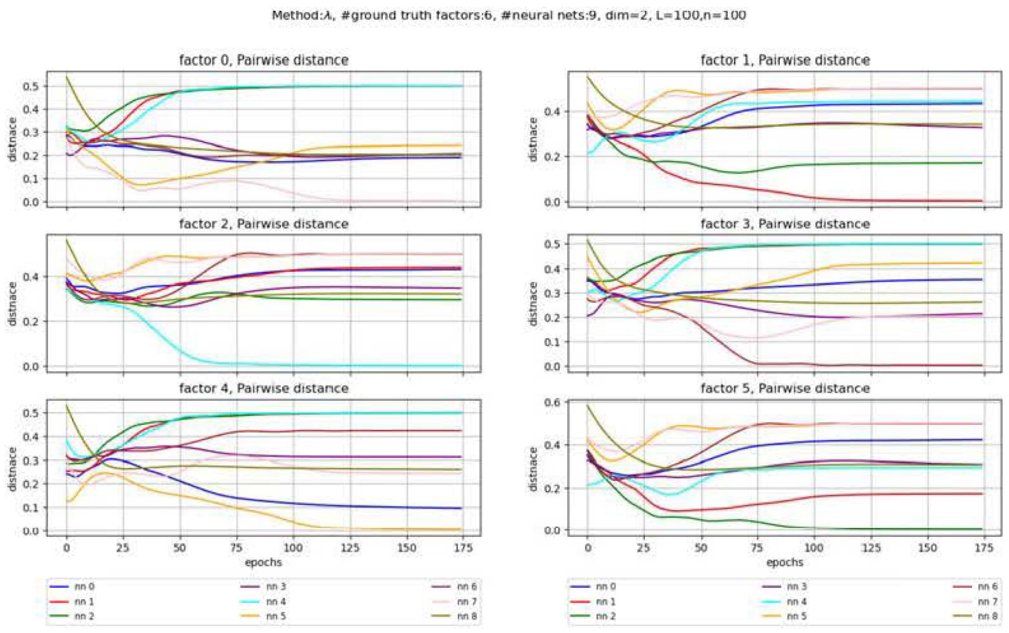


Fig. 2.10. Each plot, corresponds to the pairwise parameter distance (over training steps) of one ground truth factor to all the neural networks modeling the reusable structure. There are 6 factors and 9 neural networks.

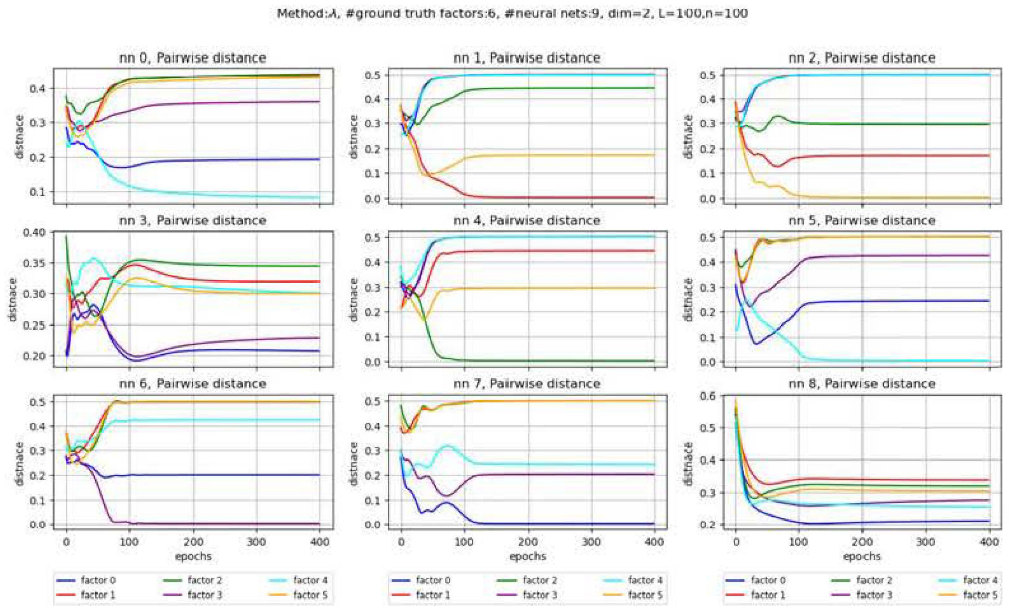


Fig. 2.11. Each plot corresponds to the pairwise parameter distance of one neural network modeling a factor compared to all the ground truth factors. There are 9 neural networks and 6 factors.

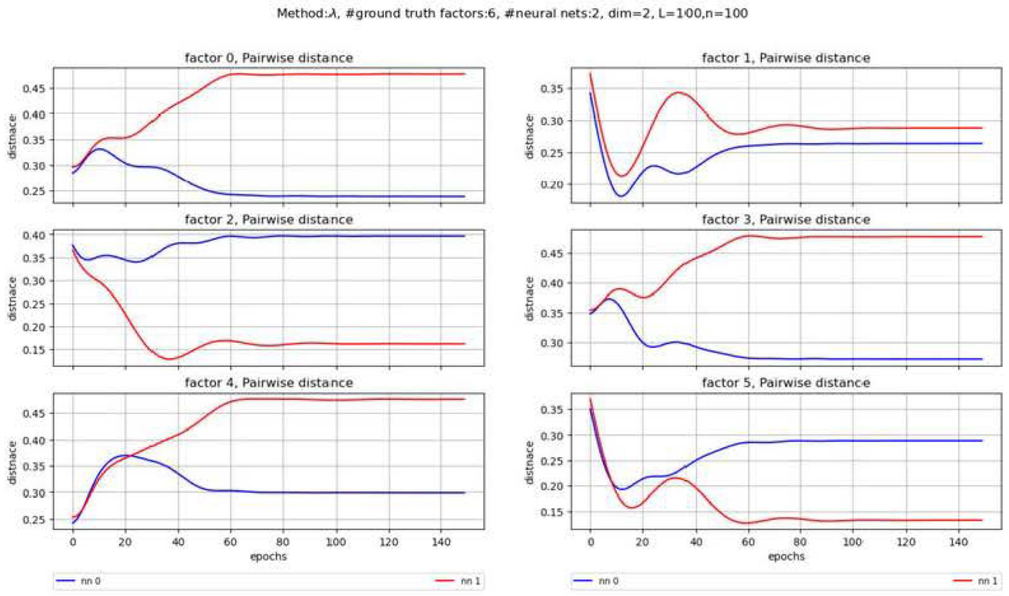


Fig. 2.12. Each plot, corresponds to the pairwise parameter distance (over training steps) of one ground truth factor to all the neural networks modeling the reusable structure. There are 6 factors and 2 neural networks.

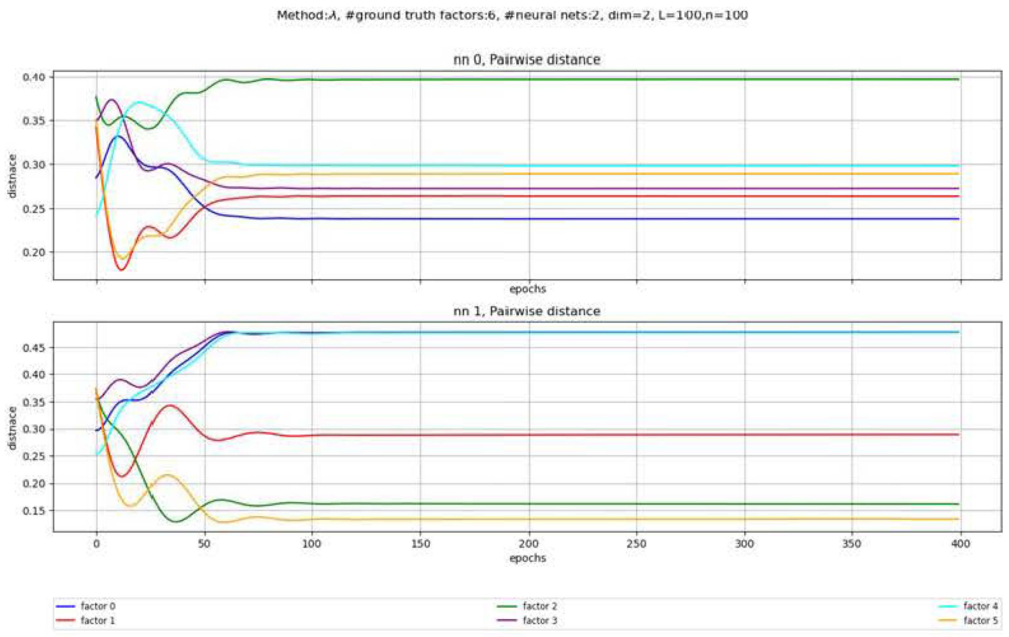


Fig. 2.13. Each plot, corresponds to the pairwise parameter distance of one neural network modeling a factor compared to all the ground truth factors. There are 2 neural networks and 6 factors.

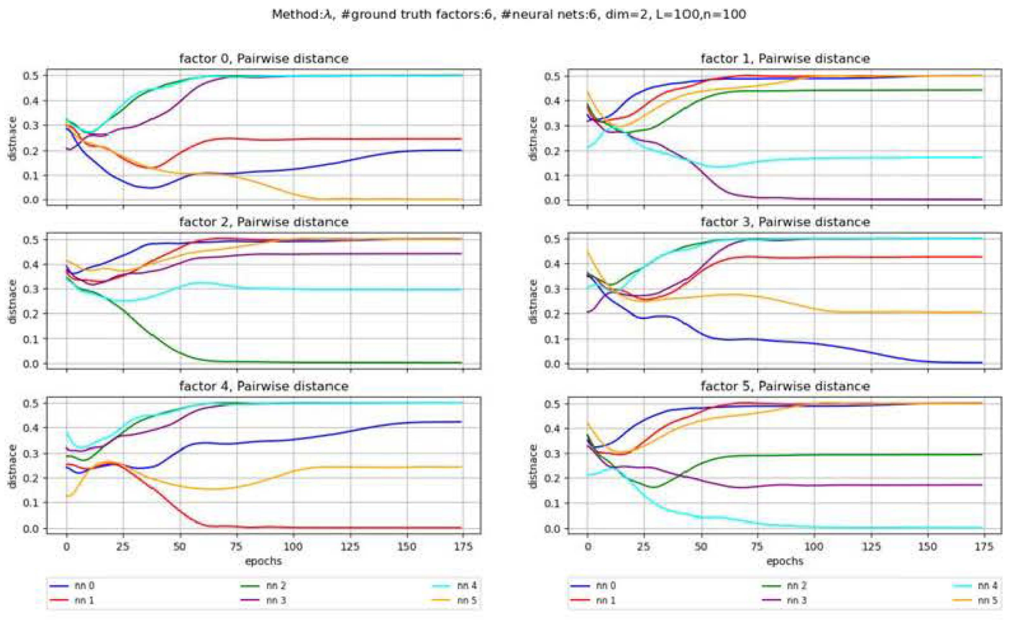


Fig. 2.14. Each plot, corresponds to the pairwise parameter distance (over training steps) of one ground truth factor to all the neural networks modeling the reusable structure. There are 6 factors and 6 neural networks.

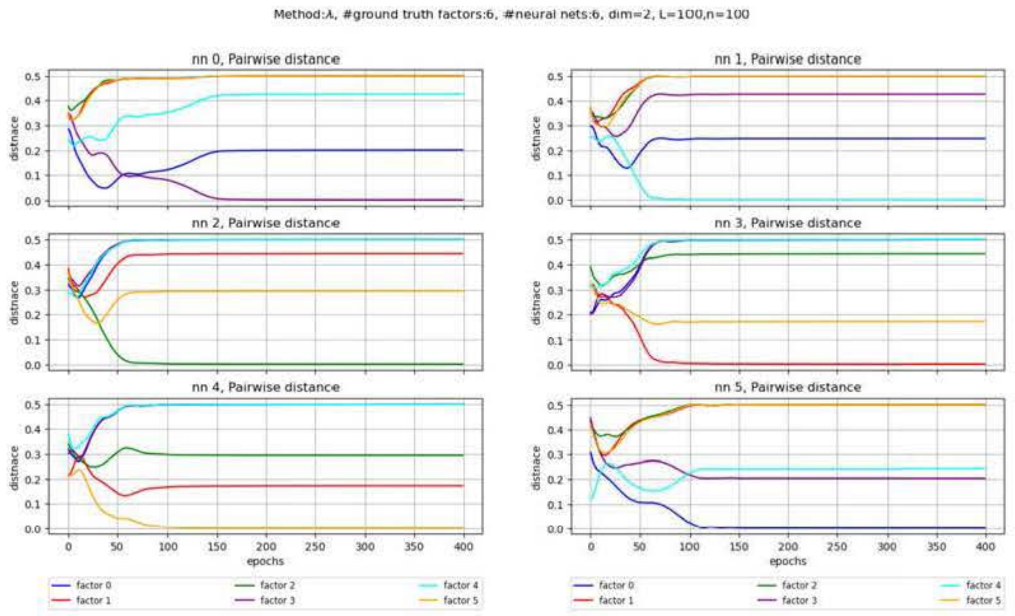


Fig. 2.15. Each plot, corresponds to the pairwise parameter distance of one neural network modeling a factor compared to all the ground truth factors. There are 6 neural networks and 6 factors.

2.6.1. Inferring r with Soft Hamming Distance

Notice that when there are fewer neural nets k than the number of factors r, there is no way we can have an estimate of the Hamming distance because we do not have enough models to construct a sequence similar to the ground truth. Hence, we introduce a *soft* Hamming distance. So far, we have measured the average parameter distance between a factor and its corresponding neural net model. Separately, we predict a sequence that represents our estimate of the position that each neural net operates on. But a natural step from here would be to measure the parameter distance between the ground truth factor that operates at position i with the neural net model we predict for that position and sum these distances over all positions $i \in [1, \ldots, L]$. This way we have the benefits of both. It also gives a more robust and accurate estimate of how close we are to the true underlying reusable structure. Because neither the Hamming distance nor parameter distance alone lack such expression; We can only learn the parameters to some approximation but recover the sequence, and vice versa. To see where both measures do well, we need to combine them as suggested. Doing so, we see a very clear distinction between data points related to $k < r$ and $k > r$ in the plot for test soft Hamming distance versus k (Figure 2.16). We see that the measure plateaus and we can confidently determine the number of ground truth factors. See appendix A for experiments that stress test the proposed algorithm by scaling the number of factors and the latent dimension even further.

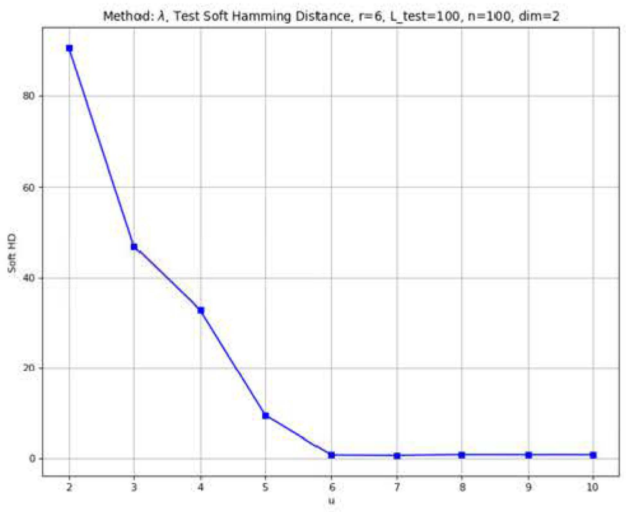


Fig. 2.16. Test set soft Hamming distance versus k, the number of neural nets to model the reusable factor graph when there are $r = 6$ factors. The metric introduced can pinpoint the correct value of r by a clear *knee point.*

2.7. Conclusion

In pursuit of improving generalization capabilities in deep learning models, we proposed a novel structure learning method for Reusable Factor Graphs. RFG departs from the conventional Directed Acyclic Graph (DAG) paradigm. By embracing factor graphs as an effective representation of causal structures, RFG not only harnesses previously unexplored inductive biases from causality and human cognition but also sets the stage for more sample-efficient and effective structure learning.

We showcased the efficacy of RFG which exploits the principles of reusability and sparsity both theoretically, and experimentally in comprehensive evaluations on synthetic data, and demonstrated how we can successfully infer the number of ground truths. We established connections between our work and the classical EM algorithm, augmenting this work with theoretical insights. As we move forward, the exploration of new variants of RFG and its application in real-world scenarios holds promise for advancing our understanding of causal relationships and enhancing the generalization capacities of deep learning models. We build on this work and the results of chapter 3 to arrive at an end-to-end model capable of jointly learning the structure and the representations in chapter 4 to tackle challenging problems with realistic environments and various tasks.

References

Abbeel, P., Koller, D., and Ng, A. Y. (2006). Learning factor graphs in polynomial time and sample complexity. *Journal of Machine Learning Research*, 7(Aug):1743–1788.

Ahuja, K., Hartford, J., and Bengio, Y. (2022a). Properties from mechanisms: an equivariance perspective on identifiable representation learning. In *International Conference on Learning Representations*.

Ahuja, K., Hartford, J., and Bengio, Y. (2022b). Weakly supervised representation learning with sparse perturbations. In *Neural Information Processing Systems*.

Arjovsky, M. (2020). *Out of Distribution Generalization in Machine Learning*. PhD thesis.

Baars, B. J. (2005). Global workspace theory of consciousness: toward a cognitive neuroscience of human experience. In Laureys, S., editor, *The Boundaries of Consciousness: Neurobiology and Neuropathology*, volume 150 of *Progress in Brain Research*, pages 45–53. Elsevier. ISSN: 0079-6123.

Bengio, Y. (2019). The consciousness prior.

Bengio, Y., Deleu, T., Rahaman, N., Ke, R., Lachapelle, S., Bilaniuk, O., Goyal, A., and Pal, C. (2019). A meta-transfer objective for learning to disentangle causal mechanisms.

Brouillard, P., Lachapelle, S., Lacoste, A., Lacoste-Julien, S., and Drouin, A. (2020). Differentiable causal discovery from interventional data. *arXiv preprint arXiv:2007.01754*.

Frey, B. J. (2012). Extending factor graphs so as to unify directed and undirected graphical models. *arXiv preprint arXiv:1212.2486*.

Goyal, A. and Bengio, Y. (2020). Inductive biases for deep learning of higher-level cognition. *arXiv preprint arXiv:2011.15091*.

Goyal, A., Lamb, A., Hoffmann, J., Sodhani, S., Levine, S., Bengio, Y., and Schölkopf, B. (2020). Recurrent independent mechanisms.

Kahneman, D. (2011). *Thinking, fast and slow*. Farrar, Straus and Giroux, New York.

Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. MIT Press.

Lachapelle, S., Brouillard, P., Deleu, T., and Lacoste-Julien, S. (2019). Gradient-based neural dag learning. *arXiv preprint arXiv:1906.02226*.

Mansouri, A., Spinney, S., Memarian, A., Conrod, P., and Rish, I. (2021). Identifying invariant and sparse predictors in high-dimensional data. *Presented at the ICML 2021 Workshop on Uncertainty and Robustness in Deep Learning*.

Obermeyer, F., Bingham, E., Jankowiak, M., Pradhan, N., Chiu, J., Rush, A., and Goodman, N. (2019). Tensor variable elimination for plated factor graphs. In *International Conference on Machine Learning*, pages 4871–4880. PMLR.

Parascandolo, G., Neitz, A., Orvieto, A., Gresele, L., and Schölkopf, B. (2020). Learning explanations that are hard to vary.

Pearl, J. (2009). *Causality*. Cambridge university press.

Peters, J., Bühlmann, P., and Meinshausen, N. (2016). Causal inference by using invariant prediction: identification and confidence intervals. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 78(5):947–1012.

Peters, J., Janzing, D., and Schölkopf, B. (2017). *Elements of causal inference: foundations and learning algorithms*.

Sachs, K., Perez, O., Pe'er, D., Lauffenburger, D. A., and Nolan, G. P. (2005). Causal protein-signaling networks derived from multiparameter single-cell data. *Science*, 308(5721):523–529.

Zheng, X., Aragam, B., Ravikumar, P. K., and Xing, E. P. (2018). Dags with no tears: Continuous optimization for structure learning. *Advances in Neural Information Processing Systems*, 31:9472–9483.

Chapter 3

Object-Centric Causal Representation Learning

by

Amin Mansouri[1], Jason Hartford[1], Yan Zhang[2], Irina Rish[1], and Yoshua Bengio[1]

(1) Mila, Quebec AI Institute and Université de Montréal

(2) Samsung, SAIT AI Lab Montréal

A shorter version of this work was initially published and presented at NeurIPS 2022 Workshop on Symmetry and Geometry in Neural Representations. An expanded version of it was accepted at NeurIPS 2023 Workshop on Causal Representation Learning. The full version of this work as appearing in this chapter is under review in the main track of ICLR 2024.

Context and Contributions. In the preceding chapter (2), our focal point rested on acquiring a reusable structure through access to causal representations. However, it was evident from the outset that an eventual relaxation of this assumption would be necessary. Meanwhile, concurrent advancements have emerged in identifying causal representations through partial structural knowledge (Ahuja et al., 2022a) and in the domain of object-centric learning (Locatello et al., 2020b). With these new ideas in play, we decided to use these insights to push our current efforts beyond conventional assumptions. Our present pursuit involves a departure from established conventions in the non-linear ICA literature, delving more deeply into the realm of causal representation learning. This study was set to strengthen the groundwork established in chapter 2 and set the stage for chapter 4.

Hence, the genesis of this project emerged from a collaborative effort between Kartik Ahuja, Jason Hartford, and myself. Notably, Yoshua Bengio provided guidance and encouraged this exploration (Kartik contributed significantly to the initial phase resulting in the workshop version, before moving to Meta AI Research). We collectively delved into relevant literature—Jason, Kartik, and I—to absorb theories and devise experiments. I took charge of translating ideas into practical experiments and implemented all the variants of our models and baselines, as well as creating a vast number of diverse vision datasets each purposed to evaluate a specific aspect of our model's capabilities. I then carried out all the experiments and subsequent analysis. My contributions extended to theoretical insights during troubleshooting the model failures. The theoretical framework, a result of discussions with Jason, Yan, and me, shaped our work profoundly. Yan's contribution through his work (Zhang et al., 2022b) was pivotal, stabilizing our training and introducing us to its working mechanisms and advantages over (Locatello et al., 2020b). The manuscript was collectively written by me, Jason, and Yan. Yan and Jason (and Kartik during his engagement) consistently offered me priceless feedback at every step of the project's evolution, which was accompanied by guidance from Irina and Yoshua, which steered the project toward success. Everyone's collective insights and support were truly instrumental in the project's achievements.

Personal Contributions.

- I did the literature review surrounding object-centric learning and methods to use alongside Ahuja et al. (2022a) for disentanglement.
- I implemented an extensive and scalable codebase for the fast generation of numerous variations of both the 2D and 3D datasets, each designed to probe a specific aspect of our method and explore the possible failure modes. The datasets used different engines, therefore there was little transfer of code from one to another, and I carried out both from scratch.
- I implemented all the baselines, and our disentanglement method based on Slot Attention (SA) and SA-MESH. Then ran all the experiments with known and unknown

perturbations, with various combinations of parameters, as well as the hyperparameter search. This was achieved due to the expansive codebase I had written for swift integration of different models and datasets.

- I extensively and meticulously troubleshooted the failure modes of our method resulting in crucial theoretical insights about the necessary conditions on the perturbations.
- I came up with three solutions to address the problem of matching, implemented them, extensively compared their run-time complexities and performance in various settings, and carried out the experiments with the fastest and best-performing algorithm.
- I produces all the plots, figures, and result tables.
- I wrote all of the paper with the exception of these sections: 3.3, 3.4, B.1
- I co-authored the rest of the paper together with Jason, Yan, and Kartik.

RÉSUMÉ. Des progrès récents et significatifs ont été réalisés dans l'apprentissage de représentation causale, qui ont montré une variété de contextes dans lesquels nous pouvons démêler les variables latentes avec des garanties d'identifiabilité (jusqu'à une classe d'équivalence raisonnable). L'hypothèse commune à toutes ces approches est que (1) les variables latentes sont des vecteurs $d-$dimensionnels, et (2) que les observations sont le résultat d'une fonction d'observation injective de ces variables latentes. Bien que ces hypothèses semblent bénignes — elles reviennent à supposer que tout changement dans l'espace latent se reflète dans l'espace d'observation et que nous pouvons utiliser des encodeurs standards pour déduire les variables latentes — nous montrons que lorsque les observations sont de multiples objets, la fonction d'observation n'est plus injective et le démêlage échoue en pratique. Nous pouvons remédier à cet échec en combinant les développements récents en matière d'apprentissage centré sur les objets et d'apprentissage par représentation causale. En modifiant l'architecture Slot Attention (Locatello et al., 2020b), nous développons une architecture centrée sur les objets qui exploite une faible supervision des perturbations clairsemées pour démêler les propriétés de chaque objet. Nous soutenons que cette approche est plus efficace en matière de données dans le sens où elle nécessite beaucoup moins de perturbations qu'une approche comparable qui encode dans un espace euclidien et nous montrons que cette approche réussit à démêler les propriétés d'un ensemble d'objets dans une série de simples expériences de démêlage basées sur des images.

Mots clés : Apprentissage de représentations causales, Apprentissage centré sur les objets, Représentations désentrelacées, Attention par emplacement, Apprentissage faiblement supervisé

ABSTRACT. There has been significant recent progress in causal representation learning that has shown a variety of settings in which we can disentangle latent variables with identifiability guarantees (up to some reasonable equivalence class). Common to all of these approaches is the assumption that (1) the latent variables are $d-$dimensional vectors, and (2) that the observations are the output of some injective observation function of these latent variables. While these assumptions appear benign—they amount to assuming that any changes in the latent space are reflected in the observation space and that we can use standard encoders to infer the latent variables—we show that when the observations are of multiple objects, the observation function is no longer injective, and disentanglement fails in practice. We can address this failure by combining recent developments in object-centric learning and causal representation learning. By modifying the Slot Attention architecture (Locatello et al., 2020b), we develop an object-centric architecture that leverages weak supervision from sparse perturbations to disentangle each object's properties. We argue that this approach is more data-efficient in the sense that it requires significantly fewer perturbations than a comparable approach that encodes to a Euclidean space and, we show that this approach successfully disentangles the properties of a set of objects in a series of simple image-based disentanglement experiments.

Keywords: causal representation learning, object-centric learning, disentangled representations, slot attention, weakly-supervised learning

3.1. Introduction

Consider the image in Figure 3.1 (left). We can clearly see four different colored balls, each at a different position. But asking, "Which is the first shape? And which is the second?" does not have a clear answer: the image just depicts an unordered set of objects. This observation seems trivial, but it implies that there exist permutations of the objects which leave the image unchanged. For example, we could swap the positions of the two blue balls without changing a single pixel in the image.

In causal representation learning, the standard assumption is that our observations x are "rendered" by some generative function $g(\cdot)$ that maps the latent properties of the image z to pixel space (i.e. $x = g(z)$); the goal is to *disentangle* the image by finding an "inverse" map that recovers z from x up to some irrelevant transformation. The only constraint on $g(\cdot)$ that is assumed by all recent papers (for example Hyvarinen and Morioka, 2016, 2017; Locatello et al., 2020a; Khemakhem et al., 2020a,b; Lachapelle et al., 2022; Ahuja et al., 2022a,b, 2023), is that $g(\cdot)$ is injective[1], such that $g(z_1) = g(z_2)$ implies that $z_1 = z_2$. But notice

[1]Some papers place stronger constraints on $g(\cdot)$, such as linearity Hyvärinen and Oja, 2000; Squires et al., 2023, sparsity Moran et al., 2022; Zheng et al., 2022, or constraints on g's Jacobian Gresele et al., 2021; Brady et al., 2023 but injectivity is the weakest assumption common to all approaches.

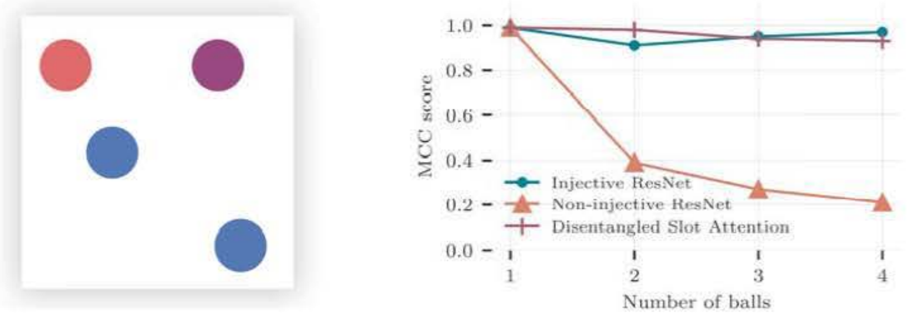


Fig. 3.1. *(Left)* An example image of simple objects. *(Right)* Mean correlation coefficient (MCC) score which measures the correlation between inferred latent variables and their associated ground truth values. Ahuja et al. (2022b)'s approach achieves almost perfect MCC scores (i.e. a score ≈ 1) when the ball color is used to make the generative function injective ("Injective ResNet"), but achieves an MCC score of at most $\frac{1}{k}$ where k is the number of objects when colors are selected randomly ("Non-injective ResNet"). We show that it is possible to recover the injective performance by disentangling object-centric representations ("Disentangled Slot Attention").

that if we represent the latents z as some d-dimensional vectors in Euclidean space, then whenever we observe objects like those shown in Figure 3.1, this injectivity assumption fails: symmetries in the objects' pixel representation imply that there exist non-trivial permutation matrices Π, such that $g(z) = g(\Pi z)$. This is not just a theoretical inconvenience: Figure 3.1 (right) shows that when the identity of the balls is not distinguishable, the disentanglement performance of a recent approach from Ahuja et al. (2022b) is upper-bounded by $1/k$ where k is the number of balls.

In parallel to this line of work, there has been significant progress in the object-centric learning literature (e.g. van Steenkiste et al., 2018a; Goyal et al., 2019; Locatello et al., 2020b; Goyal et al., 2020; Lin et al., 2020; Zhang et al., 2023; Chang et al., 2022) that has developed a suite of architectures that allow us to separate observations into sets of object representations. Two recent papers (Brady et al., 2023; Lachapelle et al., 2023) showed that the additive decoders used in these architectures give rise to provable object-wise disentanglement, but they did not address the task of disentangling the objects' associated properties. In this paper, we show that by leveraging object-centric architectures, *we effectively reduce the multi-object problem to a set of single-object disentanglement problems* which not only addresses injectivity failures but also results in a significant reduction in the number of perturbations we need to observe to disentangle properties using Ahuja et al. (2022b)'s approach.

We illustrate these results by developing a property disentanglement algorithm that combines Zhang et al. (2023)'s SA-MESH object-centric architecture with Ahuja et al. (2022b)'s approach to disentanglement and show that our approach is very effective at disentangling the properties of objects on both 2D and 3D synthetic benchmarks.

In summary, we make the following contributions:

- We highlight two problems that arise from objects that violate standard assumptions used to identify latent variables (Section 3.3).
- We show that these problems can be addressed by leveraging object-centric architectures, and that using object-centric architectures also enables us to use a factor of k fewer perturbations to disentangle properties, where k is the number of objects (Section 3.4).
- We implement the first object-centric disentanglement approach that disentangles object properties with identifiability guarantees (Section 3.5).
- We achieve strong empirical results[2] on both 2D and 3D synthetic benchmarks (Section 3.7).

3.2. Background

Causal representation learning (Schölkopf et al., 2021) seeks to reliably extract meaningful latent variables from unstructured observations such as images. This problem is impossible without additional structure because there are infinitely many latent distributions $p(z)$ that are consistent with the observed distribution, $p(x) = \int p(x|z)dp(z)$, only one of which corresponds to the ground truth distribution (Hyvärinen and Pajunen, 1999; Locatello et al., 2019). We therefore need to restrict the solution space either through distributional assumptions on the form of the latent distribution $p(z)$, or through assumptions on the functional form of the generative function $g : \mathcal{Z} \to \mathcal{X}$ that maps from the latent space to the observed space (Xi and Bloem-Reddy, 2023). A key assumption that (to the best of our knowledge) is leveraged by all papers that provide identifiability guarantees, is that $g(\cdot)$ is injective such that if we see identical images, the latents are identical (i.e. if $g(z_1) = g(z_2)$ then $z_1 = z_2$).

Given these restrictions, we can analyze the *identifiability* of latent variables for a given inference algorithm by considering the set of optimal solutions that satisfy these assumptions. We say latent variables are *identified* if the procedure will recover the latents exactly in the infinite data limit. Typically, some irreducible indeterminacy will remain, so latent variables will be identified *up to* some equivalence class $\mathcal{A}$. For example, if the true latent vector is

[2]The code to reproduce our results can be found at: https://github.com/amansouri3476/OC-CRL

z, and we have an algorithm for which all optimal solutions return a linear transformation of z such that, $\mathcal{A} = \{A : \hat{z} = Az\}$, then we say the algorithm is linearly identifies latent variables. We will call latent variables *disentangled* if the learning algorithm recovers the true latent variables up to a permutation (corresponding to a relabeling of the original variables), and element-wise transformation. That is, for all i, $z_i = h_i(z_{\pi(i)})$, where π is a permutation, and $h_i(\cdot)$ is an element-wise function; for the results we consider in this paper this function is simply a scaling and offset, $f_i(z) = a_i z_i + b_i$ corresponding to a change of units of measurement and intercept.

In this paper, we will build on a recent line of work that leverages paired samples from sparse perturbations to identify latent variables (Locatello et al., 2020a; Brehmer et al., 2022; Ahuja et al., 2022b). Our approach generalizes Ahuja et al. (2022b) to address the non-injectivity induced by objects, so we will briefly review their main results. Ahuja et al. assume that they have access to paired samples, (x, x') where $x = g(z)$, $x' = g(z')$, and z_i is perturbed by a set of sparse offsets $\Delta = \{\delta_1, \ldots, \delta_k\}$, such that $z'_i = z_i + \delta_i$ for all $i \in \{1, \ldots, k\}$. They show that if $g(\cdot)$ is an injective analytic function from $\mathbb{R}^d \to \mathcal{X}$, every $\delta \in \Delta$ is 1-sparse, and at least d linearly independent offsets are observed, then an encoder, f that minimizes the following objective recovers the true z up to permutations, scaling and an offset (Ahuja et al., 2022b, Theorem 1),

$$\hat{f} \in \arg\min_{f'} E_{x,x',\delta}\left[\left(f'(x) + \delta - f'(x')\right)^2\right] \quad \Rightarrow \quad \hat{f}(x) = \hat{z} = \Pi\Lambda z + c \tag{3.2.1}$$

where Π is a permutation matrix, Λ is an invertible diagonal matrix and c is an offset.

3.3. Objects result in non-identifiability

We begin by formally characterizing the challenges that arise when images contain multiple objects.

Data generating process. We assume that a set $Z := \{z_i\}_{i=1}^k$ of k objects is drawn from some joint distribution, $\mathbb{P}_Z$. In order to compare set and vector representations, let $\mathbf{vec}_\pi(Z)$ denote a flattened vector representation of Z ordered according to some permutation $\pi \in \mathrm{Sym}(k)$, the symmetric group of permutations of k objects; when π is omitted, $\mathbf{vec}(Z)$ simply refers to an arbitrary default ordering (i.e. the identity element of the group). Each object is described by a d-dimensional vector of properties[3] $z_i \in \mathbb{R}^d$, and hence $\mathbf{vec}(Z) \in \mathbb{R}^{kd}$. We

[3]A natural extension of the perspective we take in this paper is to also treat properties as sets rather than ordered vectors; for example, see Singh et al. (2023). We leave understanding the identifiability of these approaches to future work.

say objects have *shared properties* if the coordinates of z_i have consistent meaning across objects. For example, the objects in Figure 3.1 (left), each have x, y coordinates and a color which can be represented by its hue, so $z_i = [p^i_x, p^i_y, h^i]$. In general, the set of properties associated with an object can be different across objects, but for simplicity, our discussion will focus on properties that are fully shared between all objects.

The non-injectivity problem. We observe images x which are generated via a generative function $\mathfrak{g}(\cdot)$ that renders a set of object properties into a scene in pixel space, such that $x = \mathfrak{g}(Z)$. While $\mathfrak{g}(\cdot)$ is a set function, we can define an equivalent vector generative function, g, which, by definition, produces the same output as $\mathfrak{g}(Z)$; i.e. for all $\pi \in \mathrm{Sym}(k)$, $g(\mathbf{vec}_\pi(Z)) = \mathfrak{g}(Z)$. This generative function g taking vectors as input is consistent with standard disentanglement assumptions except that it is not injective:

Proposition 3. *If $g(\boldsymbol{vec}_\pi(Z)) = \mathfrak{g}(Z)$ for all $\pi \in Sym(k)$, then $g(\cdot)$ is not injective.*

Proof. The contrapositive of the definition of injectivity states that $z_1 \neq z_2$ implies $g(z_1) \neq g(z_2)$, but by definition of $g(\cdot)$, there exist $z_1 \neq z_2$ such that $g(z_1) = g(z_2)$. In particular, for any set Z and permutations $\pi_1 \neq \pi_2 \in \mathrm{Sym}(k)$, the vectors $\mathbf{vec}_{\pi_1}(Z) = z_1 \neq z_2 = \mathbf{vec}_{\pi_2}(Z)$. □

This proposition simply states that if images are composed of *sets* of objects, then if we model the generative function as a map from a Euclidean space, this map will not be injective by construction.

With the exception of Lachapelle et al. (2023), all of the causal representation learning papers cited in section 3.6 assume the generative function g is injective. To see why injectivity is necessary in general, consider an image with two objects. If the two objects are identical, then there are two disentangled solutions corresponding to the two permutations, so it is not possible to identify a unique solution.

The object identity problem. When applying sparse perturbations on Z (see section 3.2), we are effectively perturbing one coordinate of one object. However, how can we know which object of the multiple possible objects in Z we have perturbed? In the case of injective mappings, this is simple: since there is a consistent ordering for them, we know that a coordinate in $\mathbf{vec}(Z)$ corresponds to the same object before and after the perturbation.

However, this is no longer the case in our setting. Since the objects are actually part of a set, we cannot rely on their ordering: the perturbed object can, in principle, freely swap order with other objects; there is no guarantee that the ordering before and after the perturbation remains the same. In fact, we know that these ordering changes *must* be present due to the *responsibility problem*:

Proposition 4 (Zhang et al. (2020); Hayes et al. (2023))**.** *If the data is generated according to the data generating process described above with $g(\mathbf{vec}_\pi(Z)) := \mathfrak{g}(Z)$ and $k > 1$, then $f(\cdot)$ is discontinuous.*

Proof Sketch. Consider Figure 3.2, notice that if we perform a 90° rotation in the pixel space of the image, the image is identical, but the latent space has been permuted since each ball has swapped positions. Because the image on the left and the image on the right are identical in pixel space, any encoder, $f : \mathcal{X} \rightarrow \mathbb{R}^{kd}$, will map them them to identical latents. There exists a continuous pixel-space rotation from 0° to 90°, but it must entail a discontinuous swap in which latent is *responsible* for which part of pixel-space according to the encoder. □

A general proof can be found in Hayes et al. (2023). These discontinuities manifest themselves as changes in permutation from one $\mathbf{vec}_{\pi_1}(Z)$ to another $\mathbf{vec}_{\pi_2 \neq \pi_1}(Z)$. In disentanglement approaches that leverage paired samples (e.g. Ahuja et al., 2022b; Brehmer et al., 2022), continuity enables the learning algorithm to implicitly rely on the object identities to stay consistent. Without continuity, one cannot rely on the fact that $\mathbf{vec}(Z)$ and $\mathbf{vec}(Z) + \delta$ should be the same up to the perturbation vector δ, because the perturbation may result in a discontinuous change of $\mathbf{vec}(Z) + \delta$ when an observation is encoded back to latent space. As a consequence, we lose track of which object we have perturbed in the first place, so naïve use of existing disentanglement methods fails.

Another challenge is that the encoder f (Equation 3.2.1) has to map observations to $\mathbf{vec}(Z)$ in a *discontinuous* way, which is traditionally difficult to model with standard machine learning techniques.

In summary, the unordered nature of objects in Z results in non-injectivity, losing track of object identities, and the need for learning discontinuous functions. These all contribute to the non-identifiability of traditional disentanglement methods in theory and practice.

3.4. Object-centric causal representation learning

A natural solution to this problem is to recognize that the latent representations of multi-object images are sets and should be treated as such by our encoders and decoders in order to enforce invariance among these permutations. Both Brady et al. (2023) and Lachapelle et al. (2023) showed that architectures that enforce an appropriate object-wise decomposition in their decoders *provably* disentangle images into object-wise blocks of latent variables. These

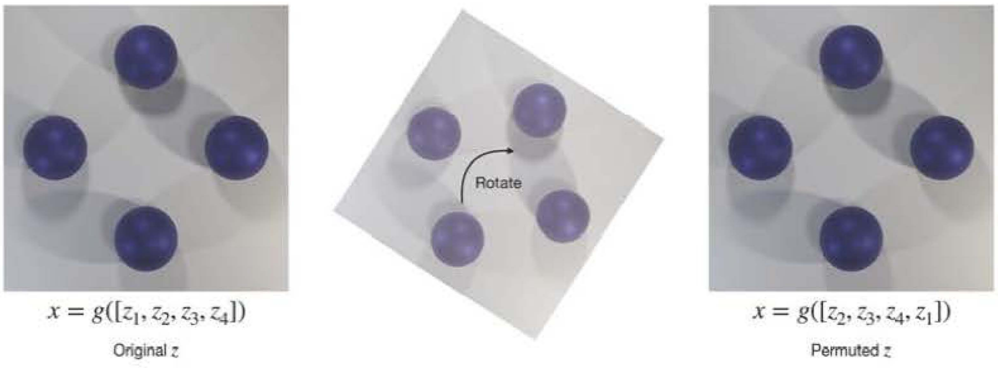


Fig. 3.2. An illustration of the object identity problem. Permuting the order of the latents $[z_1, z_2, z_3, z_3]$ is equivalent to a 90-degree rotation in pixel-space.

results do not disentangle the properties of objects, but they solve an important precursor: the assumption that there exists an object-wise decomposition of the generative function is sufficient to partition the latents into objects.

Like these two papers, we will assume that natural images can be decomposed into objects,[4] each of which occupies a disjoint set of pixels. When this is the case, we say that an image is *object-separable*. To define object separability formally, we will need to consider a partition P of an image into k disjoint subsets of pixels $P = \{x^{(1)}, \dots, x^{(k)}\}$ indexed by an index set $\mathcal{I}_P = \{1, \dots, k\}$; further, denote an index set that indexes the set of latent variables Z as $\mathcal{I}_Z$. We can then say,

Definition 1. *An image, x, is* ***object-separable*** *if there exists an* ***object-wise*** *partition P and a bijection $\sigma : \mathcal{I}_P \to \mathcal{I}_Z$ that associates each subset of pixels in P with a particular element of the set of latents, z_i, such that each subset of pixels $x^{(i)} \in P$ is the output of an injective map with respect to its associated latent $z_{\sigma(i)}$. That is, for all i, $(x^{(i)\prime} \subset \mathfrak{g}(Z'), x^{(i)} \subset \mathfrak{g}(Z))$, we have that $x^{(i)\prime} = x^{(i)}$ implies $z'_{\sigma(i)} = z_{\sigma(i)}$.*

This definition says that an image can be separated into objects if it can be partitioned into parts such that each part is rendered via an injective map from some latent z_i. We can think of each $x^{(i)}$ as a patch of pixels, with a bijection σ that relates each of the k patches of pixels in the partition $\{x^{(i)}\}_{i=1}^{k}$ to a latent variable in $Z = \{z_i\}_{i=1}^{k}$. Each patch "depends" on its associated latent via an injective map.

[4]This is a pragmatic approximation that suffices for the purposes of this paper, but a careful treatment of objects is far more subtle because what we interpret as an "object" often depends on a task or a choice of hierarchy; for a more nuanced treatment, Smith (2019)'s Chapter 8 is an excellent introduction into the subtleties around demarcating an "object".

Brady et al. (2023) and Lachapelle et al. (2023) give two different formal characterizations of partitions P that are consistent with our object-wise definition. Brady et al.'s characterization requires that a differentiable generative function $\mathfrak{g}$ is *compositional*, in the sense that each $x^{(i)} \in P$ only functionally depends[5] on a single $z_j \in Z$, and *irreducible* in the sense no $x^{(i)} \in P$ can be further decomposed into non-trivial subsets that have functionally independent latents. Lachapelle et al.'s assumption is weaker than ours in that they only require that the generative function is defined as $\mathfrak{g}(Z) = \sigma(\sum_{z_i \in Z} g_i(z_i))$ where σ is an invertible function and that $\mathfrak{g}$ is a diffeomorphism that is "sufficiently nonlinear" (see Assumption 2 Lachapelle et al., 2023); object-separable images are a special case with σ as the identity function and each $g_i(\cdot)$ rendering a disjoint subset of x, and hence their results apply to our setting.

Disentangling properties with object-centric encoding. In section 3.3 we showed that the assumptions underlying the sparse perturbation-based disentanglement approach are violated in multi-object scenes. But, the results from Brady et al. (2023) and Lachapelle et al. (2023) show that the objects can be separated into disjoint (but entangled) sets of latent variables. This suggests a natural approach to disentangling properties in multi-object scenes:

- we can reduce the multi-object disentanglement problem to a single-object problem with an object-wise partition of the image. Within each patch of pixels $x^{(i)} \in P$ injectivity holds, and so we no longer have multiple solutions at a patch level. This partition is identifiable and we can use an object-centric architecture to learn the object-wise partition. We require that this object-centric architecture can handle the responsibility problem.
- we leverage Ahuja et al. (2022b)'s approach to using weak supervision to disentangle the properties of each object individually. Since we assume that properties between objects are shared, this requires a factor of k fewer perturbations in the perturbation set Δ, where k is the number of objects.
- we address the object identity problem where we lose track of object identities after perturbations through an explicit matching procedure that re-identifies the object being perturbed.

See section 3.5 for details of how we implement this. This approach not only addresses the challenges outlined in Section 3.3, but it also significantly reduces the number of perturbations that we have to apply in order to disentangle shared properties.

[5]Functional dependence is defined by non-zero partial derivatives, i.e. $\frac{\partial x^i}{\partial z_j} \neq 0$.

Theorem 1 (informal). *If a data generating process outputs observations with k objects that have shared properties, then an object-centric architecture of the form $F(x) := \{\mathfrak{f}(x^{(i)})\}_{x^{(i)} \in P}$ where P is an object-wise partition and $\mathbf{f} : \mathcal{X} \to \mathbb{R}^d$ will disentangle in k times fewer perturbations than an encoder of the form $f : \mathcal{X} \to \mathbb{R}^{kd}$.*

The proof is given in Appendix B.1. The main insight is that if we have an object-centric architecture that learns an object-wise partition P and uses the same encoding function $\mathfrak{f}$ on every patch, then every perturbation provides weak supervision to every object, despite the fact that only one was perturbed. As a result, we do not need to disentangle each object's properties separately, and hence we reduce the number of required interventions by a factor of k.

3.5. Method

Object-wise partitions. There exist a number of ways to decompose an image into objects, but for our purposes, pixel segmentation-based approaches (Greff et al., 2019; Locatello et al., 2020b; Zhang et al., 2023) let us directly adapt existing disentanglement techniques to work with object-centric encoders. A pixel segmentation encoder $\hat{f}$ maps from images x to a set of slot vectors $\{s_1, \ldots, s_k\}$, each of which depends on a subset of the pixels $x^{(i)} \in P$. Images are then reconstructed using a slot decoder $\hat{g}$ that maps from the set of slot representations back to pixel space. The dependence between slots and patches of pixels is typically controlled by a soft-attention matrix, which will typically not result in a partition of the pixels. In our implementation, we use Zhang et al.'s SA-MESH modification of the original Locatello et al. slot attention architecture, which adds an entropy regularization term based on Sinkhorn and Knopp (1967); Cuturi (2013) to learn sparse attention matrices that do approximately partition the input by encouraging the subsets of pixels $x^{(i)}$ to be disjoint (for details on the architectures, see Appendix B.2). Importantly for us, Zhang et al. (2023) is *exclusively multiset-equivariant* (Zhang et al., 2022a), which allows it to model discontinuous functions, thus handling the responsibility problem.

Slot attention is usually trained with a reconstruction loss from relatively high-dimensional per-object slot representations, $s_i \in \mathbb{R}^D$, but for the images that we work with, we want a relatively low dimensional latent description (in the simplest case, just the two dimensions representing the (x, y) coordinates of each object). To disentangle these high-dimensional slot representations, we simply add a projection head, $\hat{p} : s_i \to \hat{z}_i$, that is trained by a latent space loss.

Disentanglement via weak supervision with matching. Ahuja et al. assume access to pairs of images (x, x') that differ by a sparse offset δ. They enforce this assumption via a disentanglement loss that requires that the latent representations of this pair of images differ by δ, such that $\hat{f}(x) + \delta = \hat{f}(x')$. When using a slot attention architecture, we introduce a *matching step* to the loss to infer the object to which the offset δ was applied. With 1-sparse δ vectors, the matching step reduces to a simple minimization over a cost matrix that measures $\|\hat{z}(x'^{(j)}) - (\hat{z}(x^{(i)}) + \delta)\|^2$ for all pairs of slots i,j. In Appendix B.4, we provide a more general matching procedure that applies to settings with dense offsets δ. We jointly optimize the following reconstruction and disentanglement loss,

$$\hat{f}, \hat{g}, \hat{p} \in \arg\min_{f,g,p} E_{\boldsymbol{x}}[\|x - g(f(x))\|^2] + E_{x,x',\delta}[\min_{i,j} \|(p(f(x')^{(j)}) - (p(f(x)^{(i)}) + \delta_t)\|^2] \quad (3.5.1)$$

The first term in this loss enforces that the encoder / decoder pair $\hat{f}, \hat{g}$ capture enough information in the slot representations s_i to reconstruct x. The second term contains the matching term and ensures that the function that projects from slot representation to latents $\hat{p}$ disentangles the slot representations into individual properties. The offset δ could be known or unknown to the model, and for the remainder of this paper, we focus on the more challenging and natural case of unknown offsets. See appendix B.3 for more details.

3.6. Related work

Causal representation learning. Our work builds on the nascent field of causal representation learning (Schölkopf et al., 2021). In particular, our disentanglement approaches builds on ideas in Ahuja et al. (2022b) which uses the same assumptions as Locatello et al. (2020a) but relaxes the requirement that the latent variables are independently distributed. These approaches form part of a larger body of recent work that shows the importance of sparsity and weak supervision from actions in disentanglement (Lachapelle et al., 2022; Lachapelle and Lacoste-Julien, 2022; Brehmer et al., 2022; Lippe et al., 2022, 2023b,a). In the appendix, we also show how known mechanisms from Ahuja et al. (2022a) can be dealt with in our framework. A closely related, but more general setting, is the recent progress on disentanglement from interventional distributions which do not require paired samples (Ahuja et al., 2023; Buchholz et al., 2023; von Kügelgen et al., 2023); we believe a useful extension of our approach would consider these settings. This literature builds on the foundational work from the nonlinear independent component analysis (ICA) literature (Hyvarinen and Morioka, 2016, 2017; Hyvarinen et al., 2019; Khemakhem et al., 2020a).

Object-centric learning. Natural data can often be decomposed into smaller entities—objects—that explain the data. The overarching goal of object-centric learning is to model such data in terms of these multiple objects. The reason for this is simple: it is usually easier to reason over a small set of relevant objects rather than, for example, a large grid of feature vectors. Representing data in this way has downstream benefits like better robustness (Huang et al., 2020). An important line of research in this area is how to obtain such objects from data like images and video in the first place. Typically, a reconstruction setup is used: given an image input, the model learns the objects in the latent space, which are then decoded back into the original image with a standard reconstruction loss (Locatello et al., 2020b; van Steenkiste et al., 2018b). Nguyen et al. (2023) propose RSM, a conceptually close idea to our work. They jointly learn object-centric representations with a modular dynamics model by minimizing a rolled-out reconstruction loss. However, they do not obtain any disentanglement of object properties, and the form of our proposed weak supervision provides insights into the effectiveness of their method for improving generalization.

We use slot attention since it makes very few assumptions about the desired data. For instance, some methods model foreground differently from background. Additionally, DINOSAUR (Seitzer et al., 2022) shows recent success on more complex images, which demonstrates the versatility of the slot attention approach. While in general object-centric models operate on image inputs and thus identify visual objects, it is in principle applicable to other domains like audio (Reddy et al., 2023) as well.

3.7. Empirical evaluation

Setup. We evaluate our method on 2D and 3D synthetic image datasets generated using Shinners (2011); Greff et al. (2022) that allow us to carefully control various aspects of the environment, such as the number of objects, their sizes, shapes, colors, relative positions, and dynamics. Such a controllable environment is an essential first step as it enables us to easily iterate and find the sources of non-identifiability for the proposed method. Examples of our 2D and 3D datasets are shown in figures 3.1,3.3 respectively. The object-wise true latents in either dataset consist of $z = (p_x, p_y, h, s, r, \phi)$, where p_x, p_y denote the coordinates of the center of an object, followed by color hue h, shape s, size r, and rotation angle ϕ about the z-axis. Based on object properties, they are each rendered and placed on a white background (for the 2D dataset) or placed on a floor that is illuminated by source lights and is being visited from somewhere above the floor (for the 3D dataset) and then aggregated to produce x_t. We use a 1D parameterization for colour by fixing colour saturation and value and only altering the colour hues. We also discretize the range of colour hues so we can test the model's ability to obtain disentangled representations in the simultaneous presence of both continuous (position, size, and rotation angle) and discrete properties (colour and shape). Our goal is to identify (up to irrelevant transformations) at the object level such true latents $z \in \mathbb{R}^d$ that give rise to the model's observations x by exploiting weak supervision from sparse perturbations. The model receives x_t,x_{t+1} along with some knowledge about the subset of mechanisms that caused the perturbations and is tasked to jointly reconstruct the image at both t,$t+1$ as well as to minimize an objective function in the latent space (see Equation 3.2.1). We show that this objective gives rise to disentangling the properties p_x,p_y,h,s,r,ϕ at the object level. Note that the model is agnostic to the continuous or discrete nature of the true latents, and the objective regardless produces a disentangled representation. All experiments in this section are carried out in the unknown mechanism setting with fully sparse perturbations. For results under known mechanisms and fully dense perturbations see appendix B.6.

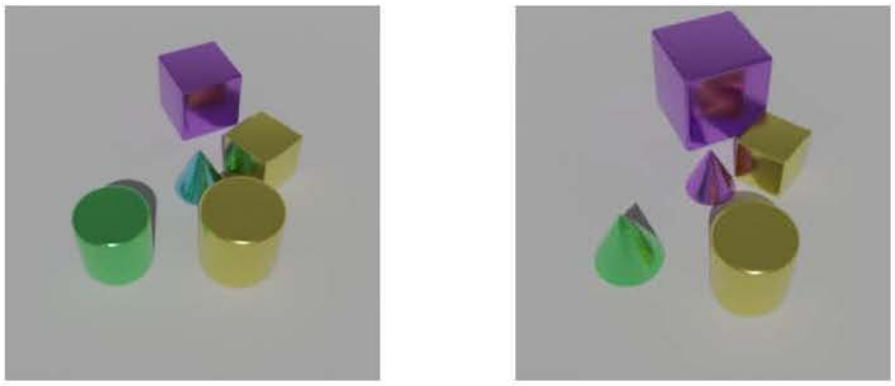


Fig. 3.3. *(Left)* An example image before any augmentations. *(Right)* Possible augmentations in the synthetic 3D dataset i.e., change in size, orientation, colour, position, and shape.

Disentanglement Metrics. We compared $\hat{z}$—the projections of non-background slots—to the true latents z of objects to measure the disentanglement of the properties in $\hat{z}$. We evaluated the identifiability of the learned representations either up to affine transformations or up to permutation and scaling. These two metrics were computed by fitting a linear regression between z,$\hat{z}$ and reporting the coefficient of determination R^2, and using the mean correlation coefficient (MCC) (Hyvarinen and Morioka, 2016, 2017).

Baselines. Our method projects slots to a latent space that has the same dimensionality d as the true latents. The model does not know anything about the structure of the true latent space z and throughout the training converges to an equivalence class of it. However, slot attention, our object-centric baseline, does not have such a projection, thus we need to have a meaningful extraction of features from high-dimensional slot representations so that we can measure if slot representations encode object properties in a disentangled manner. We use the following approaches to project slots to a $d-$dimensional space. For all slot attention baselines we match the ground truth segmentation masks to slots decoder masks to align slot projections and object-wise true latents to fit the projections. We use the same matching only for the *evaluation* of representations learned by our proposed method.

Random Projections (RP): Random projections of high-dimensional representations preserve distances in the projected space, so we can use $d-$dimensional random projections of slots that correspond to objects (requires matching with the ground truth segmentation masks) to obtain a crude estimate of vanilla slot attention's disentanglement.

Principal Components (PC): If the slots contain reasonably disentangled representations, then most of the variance of slots is expected to be due to object properties, therefore extracting the top d principal components of slot representations is a justified approach for the computation of the disentanglement scores. This should give a finer estimate than random projections.

Linear Regression (LR): We can also directly use the true latents and learn a linear mapping from non-background slots to the z space and use this projection with disentanglement metrics. Note however, that this gives an upper bound on what slot attention could achieve under linear transformations as it is completely supervised by the knowledge of the true latents, i.e., the representation we aim to discover through a much weaker signal (partial knowledge of perturbations); we only use the true latents for evaluation and not in any way we exploit them for training.

ConvNet: As the main reason for adopting set-based representations was to relax the assumption of an injective observation function, we need to compare our method against a conventional CNN encoder that can act as the inverse of an injective observation function. Concretely, if the objects in the scene are never identical and we define an ordering over objects in the scene and organize z according to that order in the dataset, then the observation function is injective, and a CNN encoder with enough capacity (such as ResNet18 (He et al., 2015)) should in principle be able to recover z up to irrelevant transformations (Ahuja et al., 2022b) by following the same disentanglement procedure we use (same as Ahuja et al. (2022a,b)). If object latents z are not organized according to a fixed ordering, then the observation function is no longer injective and we expect this baseline to fail. We denote the injective and non-injective variations by CNN† and CNN in the results table, respectively.

2D Shapes. In this section we present the empirical results that compare slot attention-based architectures with a ResNet18 trained on a non-injective and injective DGP[6] of 2,3,4 objects. Tables 3.1,3.2 confirm that as long as the observation function is injective we can empirically achieve identification (see CNN†). But the moment we drop any ordering over the objects and render x via a non-injective function, then identification via ResNet18, which is suited only to injective renderers fails disastrously (see the row corresponding to CNN in table 3.1. Also see figure 3.1). On the other hand, we can see that our method has no difficulty identifying object properties because it treats them as a set by leveraging slot attention and a matching procedure. It is also noteworthy to mention that an injective encoder requires about n times more samples to achieve the same performance because it encodes the scene onto a monolithic $\mathbb{R}^{nd}$ space where none of the $d-$dimensional sub-spaces share any representation causing the model to re-learn every property for every object exhaustively, while one key aspect of our method is to share such property representations across objects.
Table B.5 in the appendix shows the results for a particularly difficult training setting in which all of the objects are identical and have the same color, so the model cannot solely rely on colour cues to separate objects. This setting demonstrates that the failure of the injectivity assumption is not just a theoretical inconvenience, and the matching has to be successful to enable disentanglement. It is needless to say this scenario amounts to a non-injective g, and as can be seen, ResNet18 completely fails on any number of objects, whereas our method keeps achieving perfect disentanglement. For the results on other combinations of properties please see appendix B.5.1.
3D Shapes. Figure 3.3 shows examples of perturbations that the model observes and uses for disentangling object properties. We present the disentanglement scores for various combinations of properties and environments with $n = \{2,3,4\}$ number of objects in the scene. Since ConvNet (ResNet18) failed consistently in the simpler dataset of 2D shapes, we do not employ it with 3D shapes. In tables 3.3,3.4 the results for our method are reported under unknown fully sparse perturbations (Note that the SA baselines do not use any mechanisms for disentanglement.). Results for our method are averaged over 3 seeds, but since the baselines require training SA-MESH from scratch, they were trained only once as it is computationally expensive to obtain excellent reconstructions with Slot Attention (and its

[6] We make g injective by using the properties that are not the target of disentanglement, i.e., if x,y are the target properties, we will uniquely color each object based on its order in the default permutation. If x,y,c are targets, we will use unique shapes for each object based on its order in the permutation. The same logic follows for other property combinations.

derivative architecture SA-MESH). These results essentially confirm our foundations in the simpler 2D dataset and demonstrate how treating the scene as a set with our method results in perfect disentanglement of object properties. For the results on other combinations of properties please see appendix B.5.2.

Table 3.1. Linear Disentanglement (LD) scores on 2D shapes test set under *unknown* fully sparse perturbations. All results are averaged over 3 seeds except those requiring training SA-MESH from scratch that were trained only once. SA-LR, which is supervised by the ground truth latents, and is an upper bound on the disentanglement performance, achieve a score of 1.0 in all settings.

	$\text{pos}_x,\text{pos}_y$			$\text{pos}_x,\text{pos}_y$,color,size,rotation		
Model	$n = 2$	$n = 3$	$n = 4$	$n = 2$	$n = 3$	$n = 4$
Ours	**1.00** ±0.00	**1.00** ±0.00	**1.00** ±0.00	**0.95** ±0.01	**0.93** ±0.01	**0.94** ±0.02
SA-RP	0.92	0.96	0.94	0.75	0.70	0.68
SA-PC	1.00	1.00	1.00	0.93	0.88	0.86
CNN†	0.94 ±0.05	0.99 ±0.00	0.96 ±0.03	0.87 ±0.01	0.84 ±0.01	0.86 ±0.01
CNN	0.24 ±0.01	0.13 ±0.01	0.07 ±0.01	0.35 ±0.00	0.19 ±0.00	0.08 ±0.01

Table 3.2. Permutation Disentanglement (MCC) scores on 2D shapes test set under *unknown* fully sparse perturbations. All results are averaged over 3 seeds except those requiring training SA-MESH from scratch that were trained only once. SA-LR (supervised) achieves a score of 1.0 in all settings.

	$\text{pos}_x,\text{pos}_y$			$\text{pos}_x,\text{pos}_y$,color,size,rotation		
Model	$n = 2$	$n = 3$	$n = 4$	$n = 2$	$n = 3$	$n = 4$
Ours	**1.00** ±0.01	**1.00** ±0.01	**0.98** ±0.01	**0.95** ±0.01	**0.93** ±0.00	**0.94** ±0.01
SA-RP	0.80	0.90	0.82	0.58	0.52	0.50
SA-PC	1.00	1.00	1.00	0.86	0.85	0.84
CNN†	0.96 ±0.02	0.99 ±0.01	0.98 ±0.02	0.91 ±0.01	0.89 ±0.01	0.90 ±0.01
CNN	0.40 ±0.01	0.25 ±0.03	0.21 ±0.01	0.58 ±0.00	0.42 ±0.00	0.27 ±0.01

Table 3.3. LD scores on 3D shapes test set under *unknown* fully sparse perturbations. SA-LR achieves a score of 1.0 in all settings.

	$\text{pos}_x,\text{pos}_y,\text{color}$			$\text{pos}_x,\text{pos}_y,\text{color,size,rotation}$		
Model	$n = 2$	$n = 3$	$n = 4$	$n = 2$	$n = 3$	$n = 4$
Ours	**0.99** ±0.01	**0.99** ±0.00	**1.00** ±0.01	**0.91** ±0.03	**0.95** ±0.01	**0.93** ±0.01
SA-RP	0.67	0.58	0.58	0.51	0.56	0.60
SA-PC	0.64	0.62	0.64	0.56	0.76	0.76

Table 3.4. MCC scores on 3D shapes test set under *unknown* fully sparse perturbations. SA-LR achieves a score of 1.0 in all settings.

	$\text{pos}_x,\text{pos}_y,\text{color}$			$\text{pos}_x,\text{pos}_y,\text{color,size,rotation}$		
Model	$n = 2$	$n = 3$	$n = 4$	$n = 2$	$n = 3$	$n = 4$
Ours	**0.99** ±0.01	**0.99** ±0.00	**0.99** ±0.01	**0.89** ±0.02	**0.92** ±0.03	**0.92** ±0.02
SA-RP	0.62	0.54	0.54	0.49	0.50	0.46
SA-PC	0.69	0.68	0.70	0.64	0.77	0.78

3.8. Conclusion

This study establishes a connection between causal representation learning and object-centric learning, and (to the best of our knowledge) for the first time shows how to achieve disentangled representations in environments with multiple interchangeable objects. The importance of recognizing this synergy is two-fold. Firstly, causal representation learning has largely ignored the subtleties of objects in assuming injectivity and fixed $\mathbb{R}^d$ representations. Conversely, object-centric learning has not dealt with the challenge of unsupervised disentanglement. Yet disentangled representations can significantly improve a model's generalization capabilities under distribution shifts, and could also allow for learning parsimonious models of the dynamics when such proper representations are achieved, which we deem as important avenues for future research. In this study, we provided empirical evidence showcasing the successful disentanglement of object-centric representations through the fusion of slot attention with recent advances in causal representation learning.

3.9. Limitations

Our study focuses on showing when disentanglement is possible when treating object-centric environments as a set of representations instead of fixed-size vectors. We have analyzed the performance of our model comprehensively on two synthetic datasets that are relatively limited in capturing the complexities of real-world scenarios. Yet, we believe and showed such analysis is a necessary first step to identify the intricacies involved in making our algorithm work. Our analysis has been limited in a number of directions. First, while we do consider a wide range of continuous and discrete properties to be disentangled, the number of objects we use is rather low, which ideally should be scaled to real-world scenes containing more objects. Second, although our experiments include artifacts related to occlusion, depth, and lighting, in all of our experiments we simplify the problem by having the objects situated on homogenous backgrounds, whereas real-world scenes would comprise more complex backgrounds. Such decisions were mainly due to (1) generating datasets of size more than 5k for each combination of properties being a computationally heavy task on its own, (2) training SA-MESH from scratch for each combination of properties and number of objects would quickly add up as each training takes $\sim$ 12 hours on a single A100 GPU to achieve nice reconstructions, (3) details related to the background and the number of objects are tangential to the focus of this study, which is to demonstrate how to disentangle the causal factors in an object-centric environment.

References

Ahuja, K., Hartford, J., and Bengio, Y. (2022a). Properties from mechanisms: an equivariance perspective on identifiable representation learning. In *International Conference on Learning Representations*.

Ahuja, K., Hartford, J., and Bengio, Y. (2022b). Weakly supervised representation learning with sparse perturbations. In *Neural Information Processing Systems*.

Ahuja, K., Mahajan, D., Wang, Y., and Bengio, Y. (2023). Interventional causal representation learning.

Brady, J., Zimmermann, R. S., Sharma, Y., Schölkopf, B., von Kügelgen, J., and Brendel, W. (2023). Provably learning object-centric representations. *arXiv preprint arXiv: 2305.14229*.

Brehmer, J., De Haan, P., Lippe, P., and Cohen, T. (2022). Weakly supervised causal representation learning. *arXiv preprint arXiv:2203.16437*.

Buchholz, S., Rajendran, G., Rosenfeld, E., Aragam, B., Schölkopf, B., and Ravikumar, P. (2023). Learning linear causal representations from interventions under general nonlinear mixing.

Chang, M., Griffiths, T. L., and Levine, S. (2022). Object representations as fixed points: Training iterative inference algorithms with implicit differentiation. In *ICLR Workshop on Deep Generative Models for Highly Structured Data*.

Cuturi, M. (2013). Sinkhorn distances: Lightspeed computation of optimal transportation distances. *NEURIPS*.

Goyal, A., Lamb, A., Gampa, P., Beaudoin, P., Levine, S., Blundell, C., Bengio, Y., and Mozer, M. (2020). Object files and schemata: Factorizing declarative and procedural knowledge in dynamical systems. *arXiv preprint arXiv:2006.16225*.

Goyal, A., Lamb, A., Hoffmann, J., Sodhani, S., Levine, S., Bengio, Y., and Schölkopf, B. (2019). Recurrent independent mechanisms. *arXiv preprint arXiv:1909.10893*.

Greff, K., Belletti, F., Beyer, L., Doersch, C., Du, Y., Duckworth, D., Fleet, D. J., Gnanapragasam, D., Golemo, F., Herrmann, C., Kipf, T., Kundu, A., Lagun, D., Laradji, I., Liu, H.-T. D., Meyer, H., Miao, Y., Nowrouzezahrai, D., Oztireli, C., Pot, E., Radwan, N., Rebain, D., Sabour, S., Sajjadi, M. S. M., Sela, M., Sitzmann, V., Stone, A., Sun, D., Vora, S., Wang, Z., Wu, T., Yi, K. M., Zhong, F., and Tagliasacchi, A. (2022). Kubric: a scalable dataset generator.

Greff, K., Kaufman, R. L., Kabra, R., Watters, N., Burgess, C., Zoran, D., Matthey, L., Botvinick, M., and Lerchner, A. (2019). Multi-object representation learning with iterative variational inference. In *International Conference on Machine Learning*, pages 2424–2433. PMLR.

Gresele, L., Kügelgen, J. V., Stimper, V., Schölkopf, B., and Besserve, M. (2021). Independent mechanism analysis, a new concept? In *Advances in Neural Information Processing Systems*.

Hayes, B., Saitis, C., and Fazekas, G. (2023). The responsibility problem in neural networks with unordered targets.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition. *CoRR*, abs/1512.03385.

Huang, Q., He, H., Singh, A., Zhang, Y., Lim, S.-N., and Benson, A. R. (2020). Better set representations for relational reasoning. In *NeurIPS*.

Hyvarinen, A. and Morioka, H. (2016). Unsupervised feature extraction by time-contrastive learning and nonlinear ica. *Advances in Neural Information Processing Systems*, 29.

Hyvarinen, A. and Morioka, H. (2017). Nonlinear ica of temporally dependent stationary sources. In *Artificial Intelligence and Statistics*, pages 460–469. PMLR.

Hyvärinen, A. and Pajunen, P. (1999). Nonlinear independent component analysis: Existence and uniqueness results. *Neural networks*, 12(3):429–439.

Hyvarinen, A., Sasaki, H., and Turner, R. (2019). Nonlinear ica using auxiliary variables and generalized contrastive learning. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 859–868. PMLR.

Hyvärinen, A. and Oja, E. (2000). Independent component analysis: algorithms and applications. *Neural Networks*, 13(4):411–430.

Khemakhem, I., Kingma, D., Monti, R., and Hyvarinen, A. (2020a). Variational autoencoders and nonlinear ica: A unifying framework. In *International Conference on Artificial Intelligence and Statistics*, pages 2207–2217. PMLR.

Khemakhem, I., Monti, R., Kingma, D., and Hyvarinen, A. (2020b). Ice-beem: Identifiable conditional energy-based deep models based on nonlinear ica. *Advances in Neural Information Processing Systems*, 33:12768–12778.

Lachapelle, S. and Lacoste-Julien, S. (2022). Partial disentanglement via mechanism sparsity.

Lachapelle, S., Mahajan, D., Mitliagkas, I., and Lacoste-Julien, S. (2023). Additive Decoders for Latent Variables Identification and Cartesian-Product Extrapolation. arXiv:2307.02598 [cs, stat].

Lachapelle, S., Rodriguez, P., Sharma, Y., Everett, K. E., PRIOL, R. L., Lacoste, A., and Lacoste-Julien, S. (2022). Disentanglement via mechanism sparsity regularization: A new principle for nonlinear ICA. In *First Conference on Causal Learning and Reasoning*.

Lin, Z., Wu, Y., Peri, S. V., Sun, W., Singh, G., Deng, F., Jiang, J., and Ahn, S. (2020). SPACE: unsupervised object-oriented scene representation via spatial attention and decomposition. *CoRR*, abs/2001.02407.

Lippe, P., Magliacane, S., Löwe, S., Asano, Y. M., Cohen, T., and Gavves, E. (2022). Citris: Causal identifiability from temporal intervened sequences. *arXiv preprint arXiv:2202.03169*.

Lippe, P., Magliacane, S., Löwe, S., Asano, Y. M., Cohen, T., and Gavves, E. (2023a). Biscuit: Causal representation learning from binary interactions. In *The 39th Conference on Uncertainty in Artificial Intelligence*.

Lippe, P., Magliacane, S., Löwe, S., Asano, Y. M., Cohen, T., and Gavves, E. (2023b). Causal representation learning for instantaneous and temporal effects in interactive systems.

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

Locatello, F., Poole, B., Rätsch, G., Schölkopf, B., Bachem, O., and Tschannen, M. (2020a). Weakly-supervised disentanglement without compromises. In *International Conference on Machine Learning*, pages 6348–6359. PMLR.

Locatello, F., Weissenborn, D., Unterthiner, T., Mahendran, A., Heigold, G., Uszkoreit, J., Dosovitskiy, A., and Kipf, T. (2020b). Object-centric learning with slot attention. *CoRR*, abs/2006.15055.

Moran, G. E., Sridhar, D., Wang, Y., and Blei, D. (2022). Identifiable deep generative models via sparse decoding. *Transactions on Machine Learning Research.*

Nguyen, T., Mansouri, A., Madan, K., Khuong, N. D., Ahuja, K., Liu, D., and Bengio, Y. (2023). Reusable slotwise mechanisms. In *Thirty-seventh Conference on Neural Information Processing Systems.*

Reddy, P., Wisdom, S., Greff, K., Hershey, J. R., and Kipf, T. (2023). Audioslots: A slot-centric generative model for audio separation. *arXiv preprint arXiv: 2305.05591.*

Schölkopf, B., Locatello, F., Bauer, S., Ke, N. R., Kalchbrenner, N., Goyal, A., and Bengio, Y. (2021). Toward causal representation learning. *Proceedings of the IEEE*, 109(5):612–634.

Seitzer, M., Horn, M., Zadaianchuk, A., Zietlow, D., Xiao, T., Simon-Gabriel, C.-J., He, T., Zhang, Z., Schölkopf, B., Brox, T., and Locatello, F. (2022). Bridging the gap to real-world object-centric learning. *arXiv preprint arXiv: Arxiv-2209.14860.*

Shinners, P. (2011). Pygame. `http://pygame.org/`.

Singh, G., Kim, Y., and Ahn, S. (2023). Neural systematic binder. In *The Eleventh International Conference on Learning Representations.*

Sinkhorn, R. and Knopp, P. (1967). Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 21:343–348.

Smith, B. C. (2019). *The promise of artificial intelligence: reckoning and judgment.*

Squires, C., Seigal, A., Bhate, S. S., and Uhler, C. (2023). Linear causal disentanglement via interventions. In *International Conference on Machine Learning*. PMLR.

van Steenkiste, S., Chang, M., Greff, K., and Schmidhuber, J. (2018a). Relational neural expectation maximization: Unsupervised discovery of objects and their interactions. *CoRR*, abs/1802.10353.

van Steenkiste, S., Chang, M., Greff, K., and Schmidhuber, J. (2018b). Relational neural expectation maximization: Unsupervised discovery of objects and their interactions.

von Kügelgen, J., Besserve, M., Liang, W., Gresele, L., Kekić, A., Bareinboim, E., Blei, D. M., and Schölkopf, B. (2023). Nonparametric identifiability of causal representations from unknown interventions.

Xi, Q. and Bloem-Reddy, B. (2023). Indeterminacy in generative models: Characterization and strong identifiability. In Ruiz, F., Dy, J., and van de Meent, J.-W., editors, *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pages 6912–6939. PMLR.

Zhang, Y., Hare, J., and Prügel-Bennett, A. (2020). FSPool: Learning set representations with featurewise sort pooling. In *International Conference on Learning Representations*.

Zhang, Y., Zhang, D. W., Lacoste-Julien, S., Burghouts, G., and Snoek, C. G. M. (2023). Unlocking slot attention by changing optimal transport costs. *ARXIV.ORG*.

Zhang, Y., Zhang, D. W., Lacoste-Julien, S., Burghouts, G. J., and Snoek, C. G. (2022a). Multiset-equivariant set prediction with approximate implicit differentiation. In *International Conference on Learning Representations*.

Zhang, Y., Zhang, D. W., Lacoste-Julien, S., Burghouts, G. J., and Snoek, C. G. M. (2022b). Unlocking slot attention by changing optimal transport costs. In *NeurIPS 2022 Workshop on Neuro Causal and Symbolic AI (nCSI)*.

Zheng, Y., Ng, I., and Zhang, K. (2022). On the identifiability of nonlinear ica: Sparsity and beyond. *Advances in Neural Information Processing Systems*, 35:16411–16422.

Chapter 4

Reusable Slotwise Mechanisms

by

Trang Nguyen[1,2,3], Amin Mansouri[2], Kanika Madan[2], Nguyen Duy Khuong[1], Kartik Ahuja[2], Dianbo Liu[2], and Yoshua Bengio[2]

(1) FPT Software AI Center, Vietnam
(2) Mila, Quebec AI Institute and Université de Montréal
(3) Tokyo Institute of Technology, Tokyo, Japan

This work has been accepted to the main track of NeurIPS 2023.

As highlighted in preceding chapters, this body of work serves as the culmination of the thesis, synthesizing all antecedent concepts and solutions addressing the multifaceted challenges of structure and representation learning. It adeptly illustrates how these concepts can significantly enhance out-of-distribution generalization within real-world contexts and downstream tasks. Building upon the foundations laid in chapter 2, the collaboration with Trang Nguyen gained momentum, propelling the experimentation process to new heights. This collaborative effort allowed me to allocate more focused time to the endeavors delineated in chapter 3. On the other hand, while Trang diligently carried forward our collective efforts, I facilitated her rapid integration by equipping her with an expansive, adaptable, and scalable codebase, optimized for swift prototyping and experimentation. As the second co-author of this work, I embraced an exciting dual responsibility. Firstly, I served as a mentor to Trang, who was also an intern under Yoshua's guidance. I frequently provided her with abundant feedback spanning implementation intricacies and troubleshooting hurdles in various stages of the project. On a different front, I assumed a more significant role in orchestrating the broader perspective and charting the course towards achieving the most impactful outcomes. Collaborating closely with Kartik, Dianbo, and Kanika, I contributed to high-level analyses pertaining to experimental design, dataset selection, algorithmic choices, and establishment of baseline benchmarks. The collaborative effort with Trang stands as a source of pride for me and Trang as this endeavor afforded me the opportunity to engage with brilliant researchers and exert influence across both strategic vision and implementation intricacies. Provided with the codebase and feedback from me, Trang meticulously executed numerous experiments, and we analyzed them together. The composition process involved the collective contributions of Trang, myself, Dianbo, Yoshua, and Kanika. Valuable input from Kartik and Nguyen Duy further enriched our manuscript. Notably, Yoshua provided unwavering support throughout this journey, offering pivotal insights and probing questions that fundamentally guided the project to its successful fruition.

Personal Contributions.

- I did the literature review surrounding object-centric learning and dynamics modeling and the baselines NPS, C-SWM.
- I implemented an extensive and scalable codebase for swift prototyping of various models in conjunctions with different datasets. However, the implementation of the final version of RSM and fitting that alongside baselines in the codebase I provided, was done by Trang.
- I oversaw smooth conduction of the experiment, providing frequent implementation and troubleshooting feedback to Trang. The troubleshooting includes delving into low-level details of coding, and suggesting experimentation to probe the various aspects of the model.
- The final version of the RSM algorithm was the result of discussions among me, Dianbo, Kartik, and Yoshua.
- I provided guidance regarding the implementation and usage of object-centric methods.
- Trang and I produced figure 4.1.

- Throughout the various stages of the project, I mentored Trang in implementation, writing, presenting results, co-authoring author rebuttals for our submission, as well as presenting Trang with the ideas and math behind RSM and guiding her for realizing those in practice.
- I wrote the Introduction, Related Work, and Conclusion, and extensively edited the rest of the paper toward its final version.

RÉSUMÉ. Les agents capables de comprendre et de raisonner sur la dynamique des objets devraient faire preuve d'une robustesse et d'une généralisation améliorées dans de nouveaux scénarios. Cependant, pour parvenir à cette capacité, il faut non seulement une représentation efficace de la scène, mais également une compréhension des mécanismes régissant les interactions entre les sous-ensembles d'objets. Des études récentes ont fait des progrès significatifs dans la représentation de scènes à l'aide d' emplacement d'objets. Dans ce travail, nous introduisons "Reusable Slotwise Mechanisms", ou "RSM", un cadre qui modélise la dynamique des objets en tirant parti de la communication entre les emplacements ainsi qu'une architecture modulaire capable de sélectionner dynamiquement des mécanismes réutilisables pour prédire les états futurs de chaque emplacement d'objet. De manière cruciale, "RSM" exploite les *Informations contextuelles centrales (CCI)*, permettant à des mécanismes sélectionnés d'accéder aux emplacements restants via un goulot d'étranglement, permettant ainsi la modélisation d'interactions complexes d'ordre supérieur qui pourraient nécessiter un sous-ensemble creux d'objets. Les résultats expérimentaux démontrent les performances supérieures de RSM par rapport aux méthodes de pointe pour diverses prédictions futures et tâches associées en aval, notamment la réponse visuelle aux questions et la planification d'actions. De plus, nous présentons la capacité de généralisation hors distribution de "RSM" à gérer des scènes dans des scénarios complexes.
Mots clés : Apprentissage centré sur les objets, Mécanismes indépendants réutilisables, Modélisation de la dynamique, Généralisation hors distribution

ABSTRACT. Agents with the ability to comprehend and reason about the dynamics of objects would be expected to exhibit improved robustness and generalization in novel scenarios. However, achieving this capability necessitates not only an effective scene representation but also an understanding of the mechanisms governing interactions among object subsets. Recent studies have made significant progress in representing scenes using object slots. In this work, we introduce Reusable Slotwise Mechanisms, or RSM, a framework that models object dynamics by leveraging communication among slots along with a modular architecture capable of dynamically selecting reusable mechanisms for predicting the future states of each object slot. Crucially, RSM leverages the *Central Contextual Information (CCI)*, enabling selected mechanisms to access the remaining slots through a bottleneck, effectively allowing for modeling of higher order and complex interactions that might require a sparse subset of objects. Experimental results demonstrate the superior performance of RSM compared to state-of-the-art methods across various future prediction and related downstream tasks, including Visual Question Answering and action planning. Furthermore, we showcase RSM's Out-of-Distribution generalization ability to handle scenes in intricate scenarios.
Keywords: object-centric learning, reusable independent mechanisms, dynamics modeling, out-of-distribution generalization

4.1. Introduction

Accurate prediction of future frames and reasoning over objects is crucial in various computer vision tasks. These capabilities are essential for constructing comprehensive world models in applications like autonomous driving and reinforcement learning for robots. Traditional deep learning-based representation learning methods compress entire scenes into monolithic representations, lacking compositionality and object-centric understanding. As a result, these representations struggle with systematic generalization, interpretability, and capturing interactions between objects. This limitation leads to poor generalization performance as causal variables become entangled in non-trivial ways.

There has been growing interest in slot-based and modular representations that decompose scenes into individual entities, deviating from fixed-size monolithic feature vector representations (Graves et al., 2014; Santoro et al., 2018; Goyal et al., 2020, 2021a; Goyal and Bengio, 2020; Goyal et al., 2021b; Madan et al., 2021; Henaff et al., 2017; Li et al., 2018; Rosenbaum et al., 2019; Shazeer et al., 2017; Zhao et al., 2021; Liu et al., 2022). These novel approaches offer significantly more flexibility when dealing with environments that comprise multiple objects. By employing an encoder that segments a scene into its independent constituent entities instead of compressing information into a fixed-size representation, these methods allow for greater flexibility and parameter sharing when learning object-centric representations, and their compositional nature enables better generalization. Compositional and object-centric representations can be effectively utilized alongside complex world models that accurately capture the interactions and dynamics of different entities in a scene.

These world models, when presented with proper representations, in principle, can model the transition functions that relate latent causal factors across consecutive time steps of a rollout. While monolithic blocks are still used occasionally with object-centric methods Wu et al. (2023), recent attempts have incorporated similar inductive biases related to the object-centricity of images in modeling interactions (Kipf et al., 2020). Structured world models and representations seem truly promising for systematically generalizing to novel scenes. Structured world models would ideally decompose the description of the evolution of a scene into causal and independent sub-modules, making it easy to recombine and repurpose those mechanisms in novel ways to solve challenges in unseen scenarios. Such separation of dynamics modeling makes structured world models more adaptable to distribution shifts, as only the parameters of a few mechanisms that have changed in a new environment would have to be retrained, and not all of the parameters in the case of a monolithic model (Bengio et al., 2019).

A major class of such structured world models aims at baking in some inductive bias about the nature of object interactions. On one extreme, there have been studies that employ Graph Neural Networks (GNNs) to capture object dependencies through dense connections (Kipf et al., 2020), while on the other hand, there has been contrasting work aiming at modeling the dynamics through only pairwise interactions (Goyal et al., 2021a). We believe, however, that ideally, an agent should be able to learn, select, and reuse a set of prediction rules based on contextual information and the characteristics of each object.

In this work, we argue that the assumptions made in previous attempts at learning the dynamics among slots may be insufficient in more realistic domains. To address these limitations, we propose Reusable Slotwise Mechanisms (RSM), a novel modular architecture incorporating a set of deep neural networks representing reusable mechanisms on top of slotwise representations (Locatello et al., 2020; Burgess et al., 2019). Drawing inspiration from the Global Workspace Theory (GWT) in the cognitive neuroscience of working memory Baars (2005, 2017), we introduce the concept of Central Contextual Information (CCI), which allows each reusable mechanism, *i.e.*, a possible explanation of state evolution, to access information from all other slots through a bottleneck, enabling accurate predictions of the next state for a specific slot. The CCI's bottleneck amounts to a relaxed inductive bias compared to the extreme cases of pairwise or fully dense interactions among slots. Finally, through comprehensive experiments, we demonstrate that RSM outperforms the state-of-the-art in various next-step prediction tasks, including independent and identically distributed (*i.i.d.*) and Out-of-Distribution (OOD) scenarios.

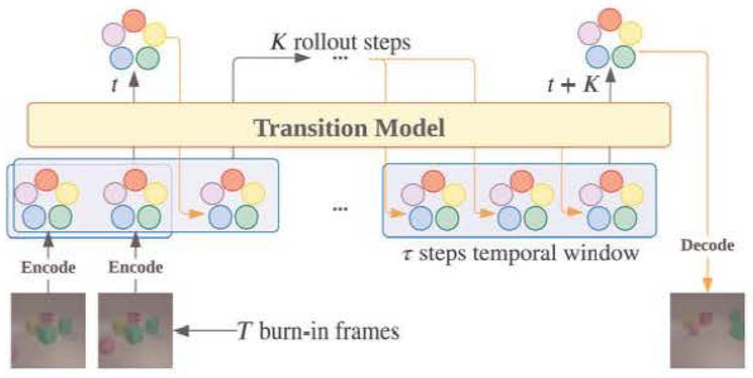
In summary, the presented work makes the following contributions: **(1)** RSM: A modular dynamics model comprising a set of reusable mechanisms that take as input slot representations through an attention bottleneck and sequential slot updates, **(2)** RSM achieves state-of-the-art OOD performance compared to baseline modular architectures in a range of long-term prediction tasks, based on ranking metrics in the latent space and as well as reconstruction loss, and **(3)** Ablation studies show how CCI benefits the mechanism selection and the prediction task, compared to the baselines.

4.2. Proposed Method: RSM - Reusable Slotwise Mechanisms

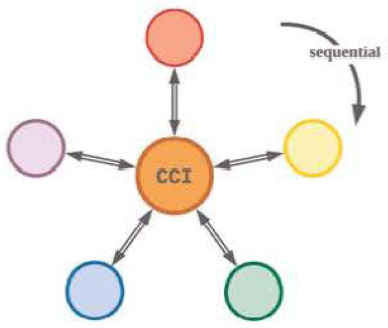
4.2.1. RSM Overview

We introduce RSM, a modular architecture consisting of a set of M Multilayer Perceptrons (MLPs) that act as reusable mechanisms, operating on slotwise representations to predict changes in the slots. What sets RSM apart from other architectures is incorporating the CCI buffer, which enhances its adaptability when dealing with environments characterized by varying levels of interaction complexity among objects by allowing the propagation of information about all other slots. Unlike previous approaches (Goyal et al., 2021b, 2020, 2021a; Ke et al., 2021), we enable a sparse subset of slots to transmit contextual information through a bottleneck for updating each slot. This inductive bias becomes helpful in environments where higher-order complex interactions need to be captured by reusable mechanisms as the CCI effectively modulates the complexity of the mechanisms and accommodates those that require one or two slots, as well as those that rely on a larger subset of slots.

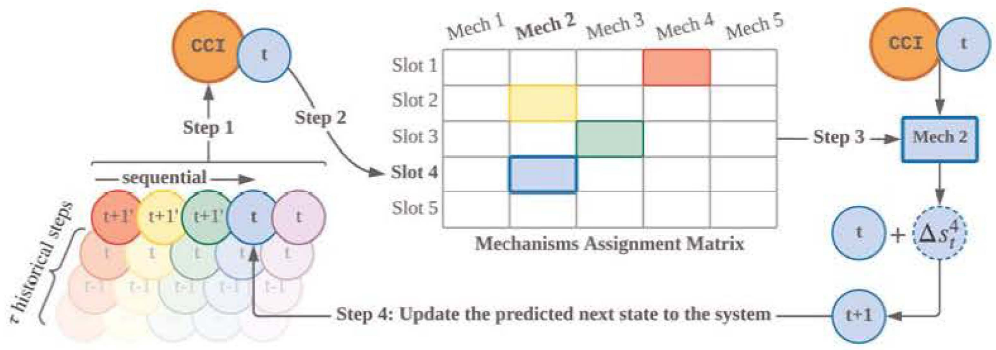
The training pipeline of RSM, which is visualized in Figure 1(a), is designed to predict K rollout steps of N object slots, based on T burn-in frames with a temporal window of maximum τ history steps for each prediction step. Prediction of the rollout begins by processing the given T burn-in frames to obtain the N slot representations for each of the T burn-in steps using an object-centric model. For any rollout step after the burn-in frames, the slots in a window of $\tau \leq T$ previous steps will be fed to the model as additional



(a) Spatial-temporal based Future Prediction Pipeline



(b) RSM Intuition



(c) Computational Flow in RSM from step t to $t+1$

Fig. 4.1. The future prediction pipeline (Figure 1(a)), RSM Intuition (Figure 1(b)), and Computational Flow (Figure 1(c)). The colored circles represent slots, with the dashed border denoting changes in the slot. The Central Contextual Information (CCI) is derived from all object slots as context, assists in selecting a mechanism for slots, and acts as an input of mechanisms. Slots are sequentially updated in four steps: **(1)** Compute CCI by unrolling slots (updated and non-updated) over the past τ steps, **(2)** Select a mechanism based on CCI and the slot of interest, **(3)** Predict the next state by the selected mechanism's dynamics, and **(4)** Update the predicted slot and prepare for the next object's turn.

context to predict the state (slots) at $t+1$. The model takes this input and updates the slots sequentially to output the slots at $t+1$. The sequential updating of slots means that updates from (according to some ordering) slots can influence the prediction of later slots within a prediction step, as illustrated in 1(b). It is worth highlighting that the sequential way of updating slots breaks the symmetries in mechanism selection, and also allows for a more expressive transition function, similar to how autoregressive models enable encoding of rich distributions. The prediction process is repeated until the slots are predicted for K rollout steps.

4.2.2. Computational Flow in RSM

This section describes the computational flow of RSM in more detail along with a 4-step process that will be repeated for all slots within a time-step t, in a sequential manner, as illustrated in figure 1(c) and Algorithm 1 in the Appendix. The following are the main components of the architecture, where d_s and d_{cci} denote the dimension of slots and the CCI, respectively:

(1) **MultiheadAttention**$(\cdot)$: $\mathbb{R}^{((\tau+1)\times N)\times d_s} \rightarrow \mathbb{R}^{d_s}$ followed by a **projection** $\phi(\cdot)$: $\mathbb{R}^{d_s} \rightarrow \mathbb{R}^{d_{\mathsf{cci}}}$ that computes the CCI, denoted as $\mathsf{cci} \in \mathbb{R}^{d_{\mathsf{cci}}}$, from all of the N slots in the past T steps concatenated. Keys, queries, and values all come from slots, so the CCI is not affected by the order of slot representations.

(2) The set of M **reusable mechanisms** $\{g_1, \dots, g_M\}$ where $g_i(\cdot)$: $\mathbb{R}^{d_{\mathsf{cci}}+d_s} \rightarrow \mathbb{R}^{d_s}$ are represented by independently parametrized MLPs implicitly trained to specialize in explaining different transitions. Each such $g_i(\cdot)$ takes as input one slot concatenated with the CCI.

(3) $\psi(\cdot)$: $\mathbb{R}^{d_{\mathsf{cci}}+d_s} \rightarrow \mathbb{R}^M$ that takes as input the CCI and the slot of interest s_t^i. It computes a categorical distribution over the possible choices of mechanisms for s_t^i, and outputs a sample of that distribution to be used for updating s_t^i.

Considering the N slots per each of the τ steps in the temporal window before t, $s_{\tau^*:t}^{1:N} = \{s_{t-\tau+1}^1, s_{t-\tau+1}^2, \dots, s_{t-\tau+1}^N, \dots, s_t^1, s_t^2, \dots, s_t^N\}$ with $\tau^* = t-\tau+1$, RSM predicts the next state of slots, denoted as $s_{t+1}^{\prime 1:N}$, using the following 4-step process, which is sequentially applied to each of the slots. Suppose an ordering has been fixed over the slots for a rollout, and according to that ordering, for some $0 < n \leq N$, we have that $n-1$ slots have been updated to their predicted values at $t+1$ and are denoted by $s_{t+1}^{\prime 1}, \dots, s_{t+1}^{\prime n-1}$. Below we explain the process of computing the next state for s_t^n (e.g. the blue slot in Figure 1(c)).

Step 1. Compute the CCI: We first append $s_t^{\prime 1:N}$ to the current temporal window to achieve $s_{\tau^*:t+1}^{1:N}$. The duplicated $s_t^{\prime 1:N}$ serves as a placeholder, which will be overwritten with the predicted values in subsequent steps. Subsequently, as presented in Equation 4.2.1, a **MultiheadAttention**$(\cdot)$ takes $s_{\tau^*:t+1}^{1:N}$ as input before passing through $\phi(\cdot)$ to produce the central contextual information cci.

$$\mathsf{cci} = \phi(\textbf{MultiheadAttention}(s_{\tau^*:t+1}^{1:N})) \tag{4.2.1}$$

Step 2. Select a mechanism for s_t^n: $\psi(\cdot)$ takes two arguments as inputs, cci from Step 1 and s_t^n, to outputs the logits of a categorical distribution $\pi_{1:M}$ over M possible choices of mechanisms. During training, we employ a Gumbel-softmax layer Maddison et al. (2016); Jang et al. (2017) on top of ψ's outputs, as described in Eq. 4.2.2, to select the mechanism. During inference, we simply choose argmax value of π_i as the selected mechanism.

$$\pi_{1:M} = \mathsf{Gumbel\text{-}softmax}(\psi(\mathsf{cci}, s_t^n)) \tag{4.2.2}$$

Step 3. Predict the changes of s_t^n: Let $\Delta s_t^{n,j}$ denote the change of s_t^n from the previous step, predicted by the selected mechanism g_j. The transformation of s_t^n, denoted as Δs_t^n, is the sum of $\Delta s_t^{n,1:M}$ weighted by $\pi_{1:M}$, as presented in Eq. 4.2.3.

$$\Delta s_t^{n,j} = g_j(\mathsf{cci}, s_t^n), \quad \Delta s_t^n = \sum_{j=1}^{M} (\Delta s_t^{n,j} \cdot \pi_j) \tag{4.2.3}$$

Step 4. Update and sync s_{t+1}^n: $s_{t+1}'^n$ is then computed by adding the predicted transformation from the previous step and replaces the value of s_{t+1}^n in the slots buffer, as described in Eq. 4.2.4.

$$s_{t+1}'^n = s_t^n + \Delta s_t^n \tag{4.2.4}$$

The process above is repeated for all slots at time t, to obtain the next state (slots) prediction $s_{t+1}^{1:N}$.

4.3. Experiments Setup

This study evaluates RSM's dynamics modeling and generalization capabilities through video prediction, VQA, and action planning tasks. We aim to provide empirical evidence supporting the underlying hypotheses that guided the architectural design of RSM.

- $\mathcal{H}_1$: Slots communication through the CCI and reusable mechanisms reduces information loss during prediction, resulting in an accurate prediction of future rollout frames (**Sec. 4.4.1**).
- $\mathcal{H}_2$: RSM produces more accurate results and effectively handles novel scenarios in the downstream tasks (**Sec. 4.4.2**), especially enhancing OOD generalization (**Sec. 4.4.3**).
- $\mathcal{H}_3$: The synergy between the CCI and the disentanglement of objects dynamics into reusable mechanisms is essential to RSM (qualitative analyses in **Sec. 4.4.4** and ablations in **Sec. 4.4.5**).

In the following subsections, we describe the experiments focusing on the transition of slots over rollout steps with pre-trained object-centric models. Additionally, Appendix C.5 provides experiments and analyses with an end-to-end training pipeline.

4.3.1. Environments

OBJ3D (Lin et al., 2020) contains dynamic scenes of a sphere colliding with static objects. Following Lin et al. (2020); Wu et al. (2023), we use 3 to 5 static objects and one launched sphere for interaction.

CLEVRER (Yi et al., 2020) shares similarities with OBJ3D, but additionally has multiple moving objects in various directions throughout the scene. For the VQA downstream task, CLEVRER offers four question types: descriptive, explanatory, predictive, and counterfactual, among which, in the spirit of improving video prediction, we focus on boosting the performance on answering predictive questions which require an understanding of future object interactions.

PHYRE (Bakhtin et al., 2019) is a 2D physics puzzle platform where the goal is to strategically place red objects such that the green object touches the blue or purple object. Bakhtin et al. (2019) design *templates* that describe such tasks with varying initial states. Subsequently, they define (1) *within-template* protocol where the test set contains the same *templates* as in training, and (2) *cross-template* protocol that tests on different *templates* that those in training. We report results both on *within-template* as iid and on *cross-template* to obtain the OOD evaluation.

Physion (Bear et al., 2021) is a VQA dataset that assesses a model's capability in predicting objects' movement and interaction in realistic simulated 3D environments in eight physical phenomena.

For further details and data visualization, we refer the readers to Appendix C.2.

4.3.2. Baselines

We compare RSM against three main baselines: **SlotFormer** (Wu et al., 2023), Switch Transformer (Fedus et al., 2021) denoted as **SwitchFormer**, and **NPS** (Goyal et al., 2021a). We show the efficacy of relaxing the inductive bias on communication density among slots by contrasting RSM with dense communication methods (SlotFormer and SwitchFormer) and a pair-wise communication method (NPS). Likewise, comparing RSM to SlotFormer highlights the role of disentangling objects' dynamics into mechanisms, while comparing to SwitchFormer and NPS emphasizes the vital role of communication among slots via the

CCI. Additionally, we compare to **SAVi-Dyn** (Wu et al., 2023), which is the SOTA on CLEVRER. In other experiments, we compare to **SlotFormer** (current SOTA).

In the tables, we present our reproduced SlotFormer (marked by "†") and our re-implemented SwitchFormer and NPS (marked by "*"), alongside SAVi-Dyn reported by Wu et al. (2023). [1]

4.3.3. Implementation Details

Following Wu et al. (2023), we focus on the transition of slots and take advantage of the pre-trained object-centric *encoder-decoder* pair that converts input frames into slots and vice versa. We use the pre-trained weights of SAVi and STEVE provided by Wu et al. (2023), including **SAVi** (Kipf et al., 2022a) for OBJ3D, CLEVRER, and PHYRE; and **STEVE** (Singh et al., 2022) for Physion.

We present the best validation set configuration of RSM for each dataset, along with fine-tuning results and model size scaling in Appendix C.4. In summary, (1) OBJ3D and CLEVRER include 7 mechanisms, while PHYRE and Physion use 5, and (2) the number of parameters in RSM is slightly lower than that of SlotFormer in corresponding experiments. Additionally, we re-implemented SwitchFormer and NPS with a similar number of parameters as in RSM and SlotFormer.

4.4. Experimental Results

We report mean and standard deviation across 5 different runs. Video visualizations of our experiments are provided in our repository [2]. See also Appendix C.1 on the reproducibility of our results.

4.4.1. Video Prediction Quality

To demonstrate $\mathcal{H}_1$, we provide the video prediction quality on OBJ3D and CLEVRER in Table 4.1 and Fig. 4.2. In general, RSM demonstrates its effectiveness in handling object dynamics in the long-term future prediction over baselines.

Evaluation Metrics We evaluate the quality and similarity of predicted rollout frames using **SSIM** (Wang et al., 2004) and **LPIPS** (Zhang et al., 2018) metrics. Since the range

[1] We adapt the code for Switch Transformer and NPS to ensure consistency of experimental setups and evaluations with SlotFormer and RSM (See Appendix C.3).

[2] github.com/trangnnp/RSM

Table 4.1. Future frame prediction quality on OBJ3D and CLEVRER. Bold scores indicate the best performance, with RSM consistently outperforming baselines by a remarkable margin.

Method	OBJ3D		CLEVRER				
	SSIM↑	LPIPS$_{\times 100}$↓	SSIM↑	LPIPS$_{\times 100}$↓	ARI↑	FG-ARI↑	FG-mIoU↑
SAVi-Dyn	0.91	12.00	0.89	19.00	8.64	64.32	18.25
NPS*	$0.90^{\pm 0.2}$	$8.24^{\pm 0.2}$	$0.89^{\pm 0.2}$	$12.51^{\pm 0.0}$	$62.84^{\pm 0.2}$	$64.62^{\pm 0.3}$	$30.39^{\pm 0.2}$
SwitchFormer*	$0.91^{\pm 0.2}$	$8.09^{\pm 0.3}$	$0.88^{\pm 0.3}$	$14.28^{\pm 0.1}$	$60.61^{\pm 0.4}$	$59.32^{\pm 0.3}$	$28.94^{\pm 0.2}$
SlotFormer[†]	$0.90^{\pm 0.2}$	$8.32^{\pm 0.2}$	$0.88^{\pm 0.2}$	$13.09^{\pm 0.1}$	$63.38^{\pm 0.3}$	$62.91^{\pm 0.2}$	$29.68^{\pm 0.3}$
RSM (Ours)	$\mathbf{0.92}^{\pm 0.1}$	$\mathbf{7.88}^{\pm 0.1}$	$\mathbf{0.91}^{\pm 0.1}$	$\mathbf{11.96}^{\pm 0.1}$	$\mathbf{67.72}^{\pm 0.2}$	$\mathbf{66.15}^{\pm 0.2}$	$\mathbf{32.73}^{\pm 0.2}$

of **LPIPS** metric is small, we report the actual values times 100, denoted as **LPIPS**$_{\times 100}$, to facilitate comparisons among the methods. Additionally, we also assess the performance using **ARI**, **FG-ARI**, and **FG-mIoU** metrics, which measure clustering similarity and foreground segmentation accuracy of object bounding boxes and segmentation masks. We evaluate the model's performance averaged over unrolling 44 steps on OBJ3D and 42 steps on CLEVRER with 6 burn-in frames in both datasets.

Table 4.1 exhibits that RSM outperforms other approaches and achieves the highest scores across evaluation metrics for both datasets. Notably, compared to SlotFormer, RSM improves LPIPS$_{\times 100}$ by 0.46 points in OBJ3D, 1.12 points in CLEVRER, and increases FG-mIoU by 3.05 points in CLEVRER. Following RSM, NPS consistently ranks second in performance among the baselines.

These results are supported by Fig. 4.2, which illustrates the rollout frames in OBJ3D. RSM's outputs' accurate rollout predictions with high visual fidelity demonstrate the efficacy of slot communication by having less error accumulated along time steps than any of the baselines. It is worth emphasizing that RSM excels in handling a significant series of complex movements in steps 20-40, particularly during the upward propulsion of the red metallic cube. In contrast, we find that the baselines struggle with complex object movements during this period, leading to inaccuracies in predicting the dynamics towards the end. Furthermore, RSM demonstrates flexible slot communication with relaxed inductive biases on interaction density, enabling it to adapt to environments comprising mechanisms with

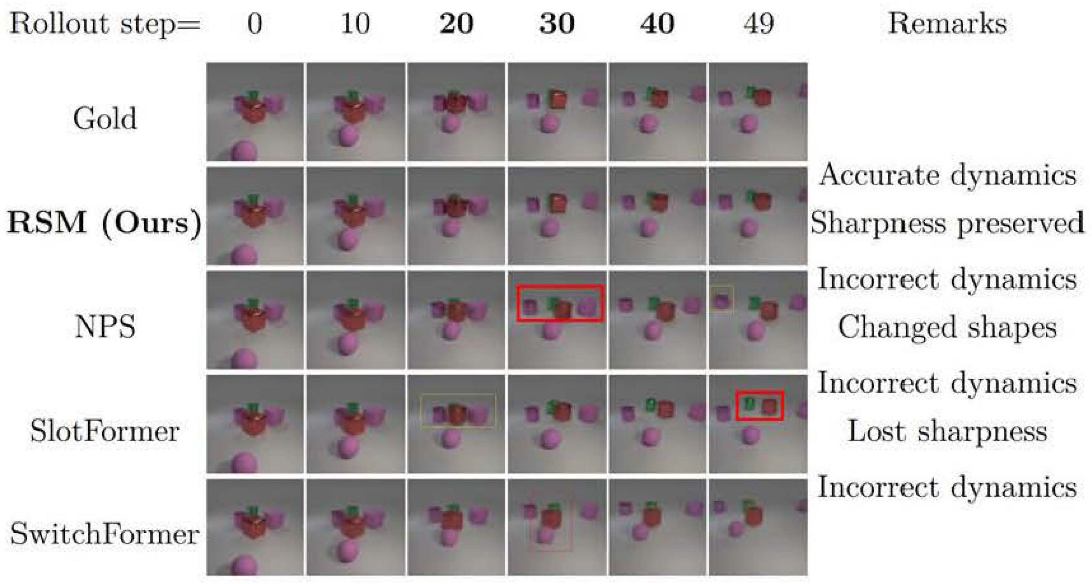


Fig. 4.2. Comparison of rollout frames in OBJ3D. RSM generates frames with precise dynamics and maintains visual quality, even during complex actions in steps 20 to 40. In contrast, the baselines produce ones with artifacts (yellow boxes) and inaccurate dynamics (red boxes).

varying levels of complexity. In contrast, we find that NPS with sparse interactions faces difficulties with close-by objects and rapid collisions (seen in OBJ3D), while SwitchFormer with dense communication struggles with distant objects (as in CLEVRER).

4.4.2. Downstream Tasks: Visual Question Answering and Action Planning

4.4.2.1. Visual Question Answering task. To demonstrate $\mathcal{H}_2$, we validate the performance of future frames generated by the models on the VQA task in CLEVRER and Physion, and the action planning task in PHYRE in the next section. The general pipeline is to solve the VQA task with the predicted rollout frames from the given input frames. Specifically, we employ **Aloe** (Ding et al., 2021) as the base reasoning model on top of the unrolled frames in CLEVRER. Likewise, in Physion, we adhere to the official protocol by training a linear model on the predicted future slots to detect objects' contact. In Physion, we also include the results obtained from human participants Bear et al. (2021) for reference; Likewise, we collect the results from observed frames (*Obs.*), which are obtained by training

Table 4.2. VQA performance on CLEVRER and Physion. Despite not surpassing human performance in Physion, RSM outperforms baselines in both datasets. All scores are in percentage.

Method	**CLEVRER**		**Physion**		
	per opt. ↑	per ques. ↑	Obs. ↑	Dyn. ↑	Gap ↑
Human	-	-	**74.7**	-	-
NPS*	$95.3^{\pm 0.3}$	$93.8^{\pm 0.2}$		$65.6^{\pm 0.3}$	+0.6
SwitchFormer*	$92.8^{\pm 0.3}$	$90.4^{\pm 0.2}$	65.0	$66.2^{\pm 0.1}$	+1.2
SlotFormer†	$96.1^{\pm 0.2}$	$93.3^{\pm 0.1}$		$66.9^{\pm 0.2}$	+1.9
RSM (Ours)	$\mathbf{96.8}^{\pm 0.1}$	$\mathbf{94.3}^{\pm 0.0}$		$\mathbf{68.1}^{\pm 0.0}$	**+3.1**

the VQA model on top of pre-trained burn-in slots and compare them to the performance of rollout slots (*Dyn.*).

In Table 4.2, RSM consistently outperforms all three baselines in VQA for CLEVRER and Physion. On CLEVRER, RSM achieves the highest scores of 96.8% (per option) and 94.3% (per question), surpassing SlotFormer and NPS. In Physion, RSM shows notable improvement, with a 3.1% increase from 65.0% in *Obs.* to 68.1% in *Dyn.*, outperforming all other baselines, indicating the benefit of enhancing the dynamics modeling to improve the downstream tasks. However, RSM is still far from human performance in Physion, showing room for further research into this class of algorithms.

4.4.2.2. Action Planning task. We adopt the approach from prior works (Qi et al., 2021; Wu et al., 2023) and construct a task success classifier as an action scoring function. This function is designed as a simple linear classifier, which considers the predicted object slots, to rank a pre-defined set of 10,000 actions from Bakhtin et al. (2019), which are executed accordingly. We utilize AUCCESS, which quantifies the success rate over the number of attempts curve's Area Under Curve (AUC) for the first 100 actions. We report the average score over 10 folds, with the best performance among 5 different runs on each fold.

The first line in Table 4.3 indicates the action planning results of the proposed RSM and baselines in the iid setting (*within-template* protocol). RSM achieves the highest performance compared to baselines, indicating the critical role of efficient communication of slots in complex tasks like action planning. In addition, Fig. 4.3 shows a successful case of RSM solving the planning task by strategically placing the red object at step 0, causing a collision between the green and blue objects at the end.

4.4.3. Out-of-Distribution (OOD) Generalization

To provide more evidence for $\mathcal{H}_2$, we resume analyzing the performance of the action planning task in PHYRE but with the *cross-template* protocol, with results indicated in the last line in Table 4.3. Overall, RSM demonstrates strong generalization and has the smallest gap between iid and OOD performance compared to the baselines. The *cross-template* is a natural method to evaluate the OOD generalization in PHYRE since scenes in the train and test sets are in distinct *templates* and contain dissimilar object sizes and objects in the background (Bakhtin et al., 2019). We refer the reader to Appendix C.2 for further details and visualizations, and Appendix C.4 for the discussion on the reproduced results.

4.4.4. The Ability to Disentangle Object Dynamics into Mechanisms

To demonstrate $\mathcal{H}_3$, we conduct qualitative analysis on the underlying mechanisms assignment within the 4 steps of Fig. 4.3 and visualize results in Fig. 4.4. While there is no explicit assignment of roles to mechanisms during training, we can infer their functionality at convergence based on slot visualizations and patterns of activation as follows: Mechanism **2** handles collisions, which can be observed in the collision between the green and red ball in step 1 and that of the red ball with the right-side wall in step 5 (blue boxes). Mechanism **3** controls the horizontal movements, observed in the green ball from steps 5 to 10 (green boxes). Mechanism **4** acts as an idleness mechanism. Mechanism **5** handles downward free-fall motion, observed from steps 2 to 3 of the two balls. Mechanism **1** does not seem to be doing anything meaningful and this could be due to the model using more capacity than it requires to model the dynamics in this environment. The inferred functions provide the following insights into understanding the efficacy of RSM: Firstly, RSM successfully disentangles the dynamics into reusable mechanisms, as described in Sec. 4.2.1. Secondly, RSM assigns proper such mechanisms to slots throughout the rollout steps, which not only helps to preserve the accuracy of prediction but also emphasizes the effectiveness of communication among slots in deciding mechanisms for each other. The automatic emergence of a `null` mechanism (mechanism 4) is also worth highlighting, which significantly helps reduce the error accumulation in action-free settings, such as idle objects in the background.

Table 4.3. Action planning task in PHYRE. RSM outperforms all baselines in both iid and OOD setups.

PHYRE-B	NPS*	SwitchFormer*	SlotFormer†	**RSM**
iid (*within-template*)	$80.52^{\pm 1.0}$	$78.27^{\pm 1.9}$	$76.4^{\pm 1.1}$	$\mathbf{82.89}^{\pm 0.6}$
OOD (*cross-template*)	$42.63^{\pm 1.3}$	$48.36^{\pm 1.4}$	$42.46^{\pm 1.7}$	$\mathbf{57.37}^{\pm 1.4}$

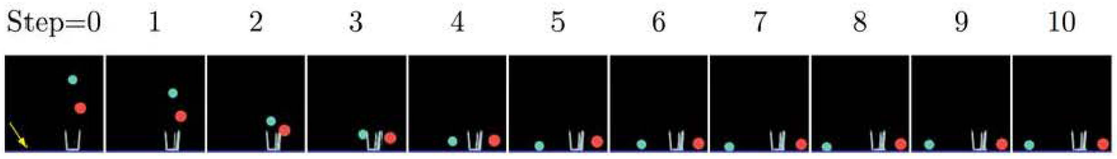


Fig. 4.3. Action planning task in PHYRE. RSM strategically positions a red ball at step 0 prevents the green ball from falling onto the glass by causing a collision that alters the original trajectory of the green ball and causing it to make contact with the blue floor (indicated by the arrow).

4.4.5. Ablation Studies

To provide more evidence for $\mathcal{H}_3$, we conduct ablations to understand the individual effects of (1) the CCI, (2) mechanisms and their disentanglement, and (3) the sequential slots updating in RSM, and visualize the results in Fig. C.1. In general, the ablation results confirm the superiority of the original RSM design compared to all variations, highlighting a significant disparity in the absence of CCI.

$\mathbf{RSM_{!2}}$ masks out the CCI in step 2 at inference. We observe that the lack of CCI leads to a degenerate selection of mechanisms for slots, with 4 out of 5 object slots being controlled by mechanism 3 (horizontal movement).

$\mathbf{RSM_{!3}}$ masks out the CCI in step 3 during inference. We have found that CCI not only captures comprehensive spatial-temporal information but also provides guidance regarding specific movement details, including directions. In particular, even when the correct mechanism is assigned (e.g. moving leftward), the slots become confused about the exact direction of movement. Additionally, slots encountered a color-related issue in the subsequent steps.

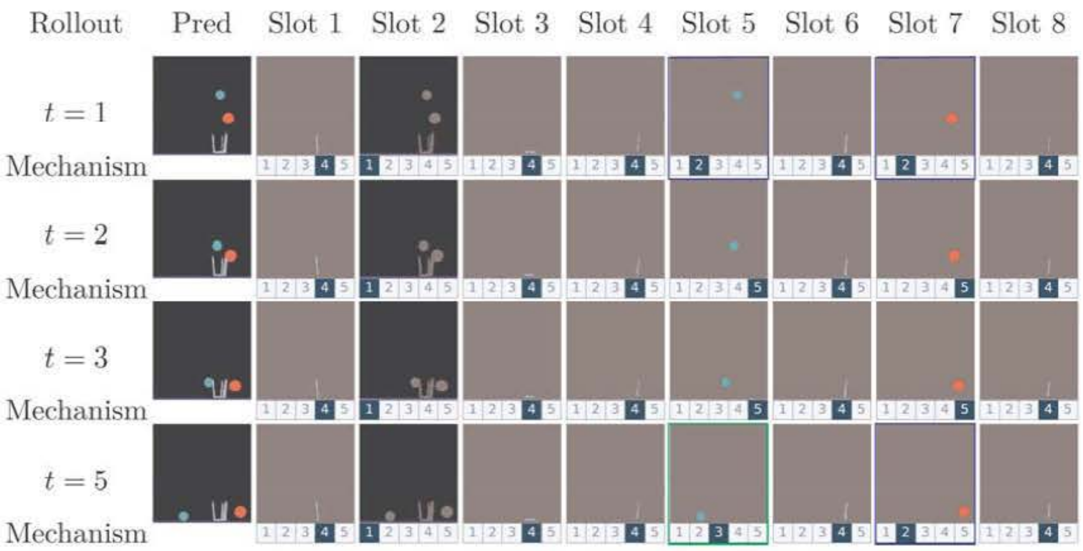


Fig. 4.4. The underlying mechanism assignment in PHYRE. Mechanisms are assigned to each slot at t to obtain the updates at $t + 1$. RSM disentangles objects' dynamics into reusable mechanisms, which can be expressed as Collision (2), Moving left or right (3), Idle (4), and Falling (5).

RSM$_{\tilde{\mathbf{k}}}$ randomly assigns mechanisms to slots by a randomized mechanism index, k, to replace the distribution π in Eq. 4.2.2. We observe that the launched ball moves in the wrong direction and exits the scene early, while other objects shake in their positions, underscoring the importance of correctly assigning mechanisms to slots.

RSM$_\mathbf{p}$ is the parallel version of RSM that modifies the model $\psi(\cdot)$ in Eq. 4.2.2 to assign mechanisms to N slots simultaneously, in both training and inference time. We find that **RSM$_\mathbf{p}$** stands out as a promising model, as it demonstrates a notable **LPIPS** performance; however, it is essential to note the partially inaccurate dynamics caused by the weaker communication among slots.

4.5. Related Work

Modular dynamics models. In the domain of modular neural networks, RIMs (Goyal et al., 2021b) pioneered the exploration of modularity for learning dynamics and long-term dependencies in non-stationary settings. However, RIMs suffer from conflating object and mechanism concepts, limiting their effectiveness. SCOFF (Goyal et al., 2020) introduced object files (OF) and schemata to address this limitation, but it struggles with generalizing out-of-distribution (OOD) scenarios. Key distinctions between RSM and SCOFF are

as follows: SCOFF schemas can handle only one OF at a time, while RSM allows multiple slots to be input to reusable mechanisms using an attention bottleneck. SCOFF and RIMs use sparse communication among OFs for rare events involving multiple objects, whereas RSM leverages the CCI to activate suitable reusable mechanisms. NPS (Goyal et al., 2021a) incorporates sparse interactions directly into its modular architecture, eliminating the need for sparse communication among slots or OFs. Their "production rules" handle rare events involving multiple objects by taking one primary slot and a contextual slot as input. A recent benchmark (Ke et al., 2021) evaluates causal discovery models and introduces a modular architecture with dense object interactions, similar to GNN-based methods, but assigns separate mechanisms to each object. Among the class of algorithms with less modularity in their dynamics model, R-NEM (van Steenkiste et al., 2018) was a pioneering unsupervised method for modeling dynamics using a learned object-centric latent space through iterative inference (see also Eslami et al. (2016)) which along similar approaches (Battaglia et al., 2018), (Battaglia et al., 2016) used Graph Neural Networks (GNNs) to model pairwise interactions and differ from our work in two key aspects. Firstly, we do not rely on GNNs to model interactions, as dense interactions are not always realistic in many environments. Secondly, we focus on learning a set of simple and reusable mechanisms that can be applied flexibly to different scenarios, rather than compressing information through shared node and edge updates in a GNN.
Unsupervised learning of object-centric representations. To decompose a scene into meaningful sub-parts, there have been lots of recent works on unsupervised learning of object-centric representations from static images (Zhao et al., 2021; Greff et al., 2017; Locatello et al., 2020; Zhang et al., 2023), videos (Pervez et al., 2022; Li et al., 2022; Kipf et al., 2022b; Elsayed et al., 2022; Wu et al., 2023), and theoretical results on the identifiability of such representations (Mansouri et al., 2022; Ahuja et al., 2022; Brady et al., 2023). Although lots of these methods work very well in practice, we decided to proceed with slot attention (Locatello et al., 2020) to be consistent with the baselines.

4.6. Conclusion

In this study, we developed RSM, a novel framework that leverages an efficient communication protocol among slots to model object dynamics. RSM comprises a set of reusable mechanisms that take as input slot representations passed through a bottleneck, the Central Contextual Information (CCI), and then they are processed sequentially to obtain slot updates. Through comprehensive empirical evaluations and analysis, we show RSM's advantages over the baselines in various tasks, including video prediction, visual question answering, and action planning tasks, especially in OOD settings. Our results suggest the importance of CCI, which integrates and coordinates knowledge from different slots for both mechanism assignment and predicting slot updates. We believe there is a promise for future research endeavors in exploring more sophisticated stochastic attention mechanisms for information integration, aligning with the principles of higher-level cognition, to be able to cope with a large number of slots and objects, and enable ways of uncertainty quantification in the predictions. In Appendix C.6, we delve into the limitations of this work and future directions.

References

Ahuja, K., Hartford, J., and Bengio, Y. (2022). Weakly supervised representation learning with sparse perturbations. *arXiv preprint arXiv: Arxiv-2206.01101.*

Baars, B. J. (2005). Global workspace theory of consciousness: toward a cognitive neuroscience of human experience. In Laureys, S., editor, *The Boundaries of Consciousness: Neurobiology and Neuropathology*, volume 150 of *Progress in Brain Research*, pages 45–53. Elsevier.

Baars, B. J. (2017). *The Global Workspace Theory of Consciousness.* John Wiley Sons, Ltd.

Bakhtin, A., van der Maaten, L., Johnson, J., Gustafson, L., and Girshick, R. (2019). Phyre: A new benchmark for physical reasoning.

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V. F., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., Gülçehre, Ç., Song, H. F., Ballard, A. J., Gilmer, J., Dahl, G. E., Vaswani, A., Allen, K. R., Nash, C., Langston, V., Dyer, C., Heess, N., Wierstra, D., Kohli, P., Botvinick, M. M., Vinyals, O., Li, Y., and Pascanu, R. (2018). Relational inductive biases, deep learning, and graph networks. *CoRR*, abs/1806.01261.

Battaglia, P. W., Pascanu, R., Lai, M., Rezende, D. J., and Kavukcuoglu, K. (2016). Interaction networks for learning about objects, relations and physics. In *NIPS*.

Bear, D. M., Wang, E., Mrowca, D., Binder, F. J., Tung, H. F., Pramod, R. T., Holdaway, C., Tao, S., Smith, K. A., Sun, F., Fei-Fei, L., Kanwisher, N., Tenenbaum, J. B., Yamins, D. L. K., and Fan, J. E. (2021). Physion: Evaluating physical prediction from vision in humans and machines. *CoRR*, abs/2106.08261.

Bengio, Y., Deleu, T., Rahaman, N., Ke, R., Lachapelle, S., Bilaniuk, O., Goyal, A., and Pal, C. (2019). A meta-transfer objective for learning to disentangle causal mechanisms. In *ICLR'2020, arXiv:1901.10912*.

Brady, J., Zimmermann, R. S., Sharma, Y., Schölkopf, B., von Kügelgen, J., and Brendel, W. (2023). Provably learning object-centric representations. *arXiv preprint arXiv: 2305.14229*.

Burgess, C. P., Matthey, L., Watters, N., Kabra, R., Higgins, I., Botvinick, M. M., and Lerchner, A. (2019). Monet: Unsupervised scene decomposition and representation. *CoRR*.

Ding, D., Hill, F., Santoro, A., Reynolds, M., and Botvinick, M. (2021). Attention over learned object embeddings enables complex visual reasoning. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems*.

Elsayed, G. F., Mahendran, A., van Steenkiste, S., Greff, K., Mozer, M. C., and Kipf, T. (2022). SAVi++: Towards end-to-end object-centric learning from real-world videos.

Eslami, S. M. A., Heess, N., Weber, T., Tassa, Y., Kavukcuoglu, K., and Hinton, G. E. (2016). Attend, infer, repeat: Fast scene understanding with generative models. *CoRR*, abs/1603.08575.

Fedus, W., Zoph, B., and Shazeer, N. (2021). Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *CoRR*, abs/2101.03961.

Goyal, A. and Bengio, Y. (2020). Inductive biases for deep learning of higher-level cognition. *CoRR*, abs/2011.15091.

Goyal, A., Didolkar, A., Ke, N. R., Blundell, C., Beaudoin, P., Heess, N., Mozer, M., and Bengio, Y. (2021a). Neural production systems. *CoRR*, abs/2103.01937.

Goyal, A., Lamb, A., Gampa, P., Beaudoin, P., Levine, S., Blundell, C., Bengio, Y., and Mozer, M. (2020). Object files and schemata: Factorizing declarative and procedural knowledge in dynamical systems. *arXiv preprint arXiv:2006.16225*.

Goyal, A., Lamb, A., Hoffmann, J., Sodhani, S., Levine, S., Bengio, Y., and Schölkopf, B. (2021b). Recurrent independent mechanisms.

Graves, A., Wayne, G., and Danihelka, I. (2014). Neural turing machines. *CoRR*, abs/1410.5401.

Greff, K., Van Steenkiste, S., and Schmidhuber, J. (2017). Neural expectation maximization. *Advances in Neural Information Processing Systems*.

Henaff, M., Weston, J., Szlam, A., Bordes, A., and LeCun, Y. (2017). Tracking the world state with recurrent entity networks.

Jang, E., Gu, S., and Poole, B. (2017). Categorical reparameterization with gumbel-softmax. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Ke, N. R., Didolkar, A., Mittal, S., Goyal, A., Lajoie, G., Bauer, S., Rezende, D., Bengio, Y., Mozer, M., and Pal, C. (2021). Systematic evaluation of causal discovery in visual model based reinforcement learning.

Kipf, T., Elsayed, G. F., Mahendran, A., Stone, A., Sabour, S., Heigold, G., Jonschkowski, R., Dosovitskiy, A., and Greff, K. (2022a). Conditional object-centric learning from video. In *International Conference on Learning Representations*.

Kipf, T., Elsayed, G. F., Mahendran, A., Stone, A., Sabour, S., Heigold, G., Jonschkowski, R., Dosovitskiy, A., and Greff, K. (2022b). Conditional object-centric learning from video. *ICLR*.

Kipf, T., van der Pol, E., and Welling, M. (2020). Contrastive learning of structured world models. In *International Conference on Learning Representations*.

Li, S., Li, W., Cook, C., Zhu, C., and Gao, Y. (2018). Independently recurrent neural network (indrnn): Building a longer and deeper rnn.

Li, S., Wu, K., Zhang, C., and Zhu, Y. (2022). On the learning mechanisms in physical reasoning. *ArXiv*, abs/2210.02075.

Lin, Z., Wu, Y.-F., Peri, S., Fu, B., Jiang, J., and Ahn, S. (2020). Improving generative imagination in object-centric world models. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org.

Liu, D., Shah, V., Boussif, O., Meo, C., Goyal, A., Shu, T., Mozer, M. C., Heess, N. M. O., and Bengio, Y. (2022). Stateful active facilitator: Coordination and environmental heterogeneity in cooperative multi-agent reinforcement learning. *ArXiv*, abs/2210.03022.

Locatello, F., Weissenborn, D., Unterthiner, T., Mahendran, A., Heigold, G., Uszkoreit, J., Dosovitskiy, A., and Kipf, T. (2020). Object-centric learning with slot attention. NIPS'20.

Madan, K., Ke, N. R., Goyal, A., Schölkopf, B., and Bengio, Y. (2021). Fast and slow learning of recurrent independent mechanisms.

Maddison, C. J., Mnih, A., and Teh, Y. (2016). The concrete distribution: A continuous relaxation of discrete random variables. *International Conference On Learning Representations*.

Mansouri, A., Hartford, J., Ahuja, K., and Bengio, Y. (2022). Object-centric causal representation learning. *NeurIPS Workshop on Symmetry and Geometry*.

Pervez, A., Lippe, P., and Gavves, E. (2022). Differentiable mathematical programming for object-centric representation learning. *arXiv preprint arXiv: Arxiv-2210.02159.*

Qi, H., Wang, X., Pathak, D., Ma, Y., and Malik, J. (2021). Learning long-term visual dynamics with region proposal interaction networks. In *International Conference on Learning Representations.*

Rosenbaum, C., Cases, I., Riemer, M., and Klinger, T. (2019). Routing networks and the challenges of modular and compositional computation. *CoRR*, abs/1904.12774.

Santoro, A., Faulkner, R., Raposo, D., Rae, J., Chrzanowski, M., Weber, T., Wierstra, D., Vinyals, O., Pascanu, R., and Lillicrap, T. (2018). Relational recurrent neural networks.

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q. V., Hinton, G. E., and Dean, J. (2017). Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *CoRR*, abs/1701.06538.

Singh, G., Wu, Y.-F., and Ahn, S. (2022). Simple unsupervised object-centric learning for complex and naturalistic videos. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K., editors, *Advances in Neural Information Processing Systems.*

van Steenkiste, S., Chang, M., Greff, K., and Schmidhuber, J. (2018). Relational neural expectation maximization: Unsupervised discovery of objects and their interactions. *International Conference On Learning Representations.*

Wang, Z., Bovik, A., Sheikh, H., and Simoncelli, E. (2004). Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612.

Wu, Z., Dvornik, N., Greff, K., Kipf, T., and Garg, A. (2023). Slotformer: Unsupervised visual dynamics simulation with object-centric models. In *The Eleventh International Conference on Learning Representations.*

Yi, K., Gan*, C., Li, Y., Kohli, P., Wu, J., Torralba, A., and Tenenbaum, J. B. (2020). Clevrer: Collision events for video representation and reasoning. In *International Conference on Learning Representations.*

Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O. (2018). The unreasonable effectiveness of deep features as a perceptual metric. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 586–595, Los Alamitos, CA, USA. IEEE Computer Society.

Zhang, Y., Zhang, D. W., Lacoste-Julien, S., Burghouts, G., and Snoek, C. G. M. (2023). Unlocking slot attention by changing optimal transport costs. *ARXIV.ORG.*

Zhao, M., Liu, Z., Luan, S., Zhang, S., Precup, D., and Bengio, Y. (2021). A consciousness-inspired planning agent for model-based reinforcement learning. *Advances in neural information processing systems*, 34:1569–1581.

Chapter 5

Conclusion and Future Directions

In this thesis, our journey has been an exploration of utmost importance—addressing the formidable challenge of out-of-distribution generalization through the innovative perspectives from causal structure and representation learning. Our research was rooted in the understanding that effectively learning causal relationships and the construction robust representations could offer the missing link between AI's performance on known data and its adaptability to unforeseen scenarios. The culmination of our efforts resulted in a coherent narrative of innovations building toward RSM, a method that achieves state-of-the-art in generalization under real-world benchmarks.

The thesis commenced with the introduction of Reusable Factor Graphs (RFG) that departed from the conventional Directed Acyclic Graph (DAG) approach to structure learning. RFG modified and harnessed the potential of factor graphs as a representation of causal mechanisms since the modification allowed for effectively incorporating the important inductive biases of reusbaility, modularity, and sparsity. This work enhanced the sample efficiency of structure learning by successfully showing how we can exploit the reusability of causal mechanisms, and thus opens avenues for more effective causal discovery.
While our experiments yielded promising outcomes in the context of structure learning on synthetic datasets, there are numerous avenues for further research. To avoid the computational intricacies associated with computing the partition function, we simplified the problem by focusing on time-directed reusable factor graphs. However, recent advancements in Generative Flow Networks (GFlowNets) offer potential avenues to address this computational challenge. It is noteworthy that our approach thus far has predominantly addressed scenarios where a single factor, at each step of the Markov chain, transforms the entirety of latents z. A natural progression could involve the incorporation of multiple factors competing for transforming *blocks* of latents (akin to the principles underpinning object-centric representations). Although Chapter 4 extensively builds on this idea and shows its effectiveness in

practice, an in-depth exploration of its theoretical aspects could be insightful. Furthermore, extending our perspective from time-directed RFGs, there lies the opportunity to generalize our findings by recovering a broader class of factor graphs. Additionally, the translation of RFGs into Causal DAGs merits exploration, as the directionality of causality holds significance in specific tasks. On the front of inferring the number of ground truth factors, a more refined approach might involve the use of sophisticated adaptive clustering techniques, such as Dirichlet Process Mixtures.

Though we leveraged access to causal representations to learn RFG structures, we later pivoted our focus to learning such representations in an attempt to relax the aforementioned assumption. Recognizing that the world is inherently object-centric, our research departed from traditional assumptions, marking the inception of an algorithm that, for the first time, learns *object-centric* disentangled representations—breaking away from the limitations of simplistic assumptions surrounding monolithic fixed-size vectors.

This work harnessed weak supervision derived from partial and incomplete knowledge of the underlying causal structure inherent within observations to learn causal representations. We also demonstrated how embracing object-centricity of the natural world can lead to significant sample efficiency gains for learning such representations. This approach, while rooted in theoretical insights, shines a light on the practical implications of object-centric representation learning.

We analyzed the performance of our model comprehensively on two synthetic datasets that are relatively limited in capturing the complexities of very real-world scenarios. Yet, we found and demonstrated that such level of complexity was a necessary first step to identify the intricacies involved in making our algorithm work. Therefore, our analysis has been limited in several aspects, which present challenges that we are enthusiastic to address in future research endeavors.

First, while we do consider a wide range of properties to be disentangled, we focused on relatively low numbers of objects. Future research could explore how our model scales to real-world scenes with a multitude of objects—offering a deeper understanding of the complexities that lie ahead. Second, although our experiments include artifacts related to occlusion, depth, and lighting, in all of our experiments we simplify the problem by having the objects situated in a plain white background. Future directions include attempts to extend our results to real-world scenes containing far more complex backgrounds and artifacts. Additionally, depth is a strong signal that is widely available with today's Lidar sensors. Incorporating such signals and potential interventions on them could enrich the capabilities of our method. Lastly, much like our approach of embracing the inherent set nature of environments and uncovering object-centric disentangled representations, a promising avenue for future research lies in the exploration of per-object set representations. This entails encoding individual objects not as fixed vectors, but as sets of properties. This approach would allow for varying-sized sets of disentangled properties, catering to the unique characteristics of different objects. This extension could further enhance our ability to capture the complexity and diversity of real-world scenes, propelling us closer to achieving more comprehensive and robust representation learning.

The narrative of our thesis culminated in the third chapter where the synergy of the previous distinct perspectives naturally converged, igniting the idea of joint learning of the reusable structure and representation—a path that naturally emerged as a resolution to both facets of the challenge. We introduced the Reusable Slot Mechanisms (RSM) that shows great promise and holds potential for real-world applications. With an emphasis on efficient communication among object-centric representations, RSM showcased its prowess in modeling object dynamics. The empirical evidence, showcased through various tasks and evaluation scenarios, reinforced the significance of the Central Contextual Information (CCI).

The RSM architecture, while robust in modeling objects' dynamics across tasks and settings, reveals its limitations in the following areas:

- Sensitivity to Hyperparameters: The process of fine-tuning the number and size of mechanisms within RSM requires careful manual adjustment. Future exploration could focus on enhancing the model's robustness to hyperparameters.
- Threat to Time Complexity: In larger systems with numerous slots, the sequential update of slots can become computationally intensive. While this limitation does not currently impact our work, parallelization techniques could be explored to optimize computational efficiency, simultaneously assigning mechanisms and predicting states for large-scale systems.
- Observable Environments: The environments studied in this work are predominantly observable. To expand the horizons of RSM's applicability, future research could explore a wider range of observable and unobservable environments to gather deeper insights into the architecture's capabilities and limitations.

The culmination of this thesis underscores the transformative potential of merging causal structure and representation learning to address the challenge of out-of-distribution generalization. As our contributions converged, they carved a pathway towards enhancing the generalization capacities of deep learning models. From RFG's innovative approach to structure learning to the innovative concept of object-centric disentangled representations and the practical manifestation of RSM, this journey spans theoretical innovation to real-world application.

As we conclude the thesis, we stand at the intersection of knowledge and possibility. The fusion of theory and practice of structure and representation learning has equipped us with novel tools to tackle challenges, bridge gaps, and spark further advancements toward addressing the challenge of out-of-distribution generalization. We look back with pride at our contributions and forward with anticipation of the prospects that await. Our journey has been a testament to the power of blending diverse perspectives, challenging conventional wisdom, and embarking on uncharted territories. Our contributions in the realm of causal structure and representation learning has not only enriched these fields, but it has also leveraged their power to enhance out-of-distribution generalization. As we conclude this chapter, we stand on the threshold of new beginnings, armed with insights, innovation, and an insatiable thirst for knowledge.

Appendix A

Supplementary Materials for Structure Learning of Reusable Factor Graphs

A.1. Implementation Details

In all experiments $X_0 \sim$ uniform$[-0.3,0.3]^d$, and the coefficient matrices of the ground truth factors are sampled as unitary matrices of dimension similar to the dimension of X_0, i.e. $\dim(X_0) = d$. The noise variance is set to $\sigma^2_{\mathcal{N}} = 0.1$ (or $1/3$ of X_0's range). We train the models for either 100 or 1000 epochs with Adam optimizer with default parameters for βs with an initial learning rate of 0.1 for neural networks and 0.01 for the optimization of the lambda matrix. Learning rate scheduler is ReduceLROnPlateau. The results are averaged over 5 runs. The test set is composed of 20 samples (chains) of Length 50, dimension d same as the training, and they are generated by a new sequence but with the same factors as the training.

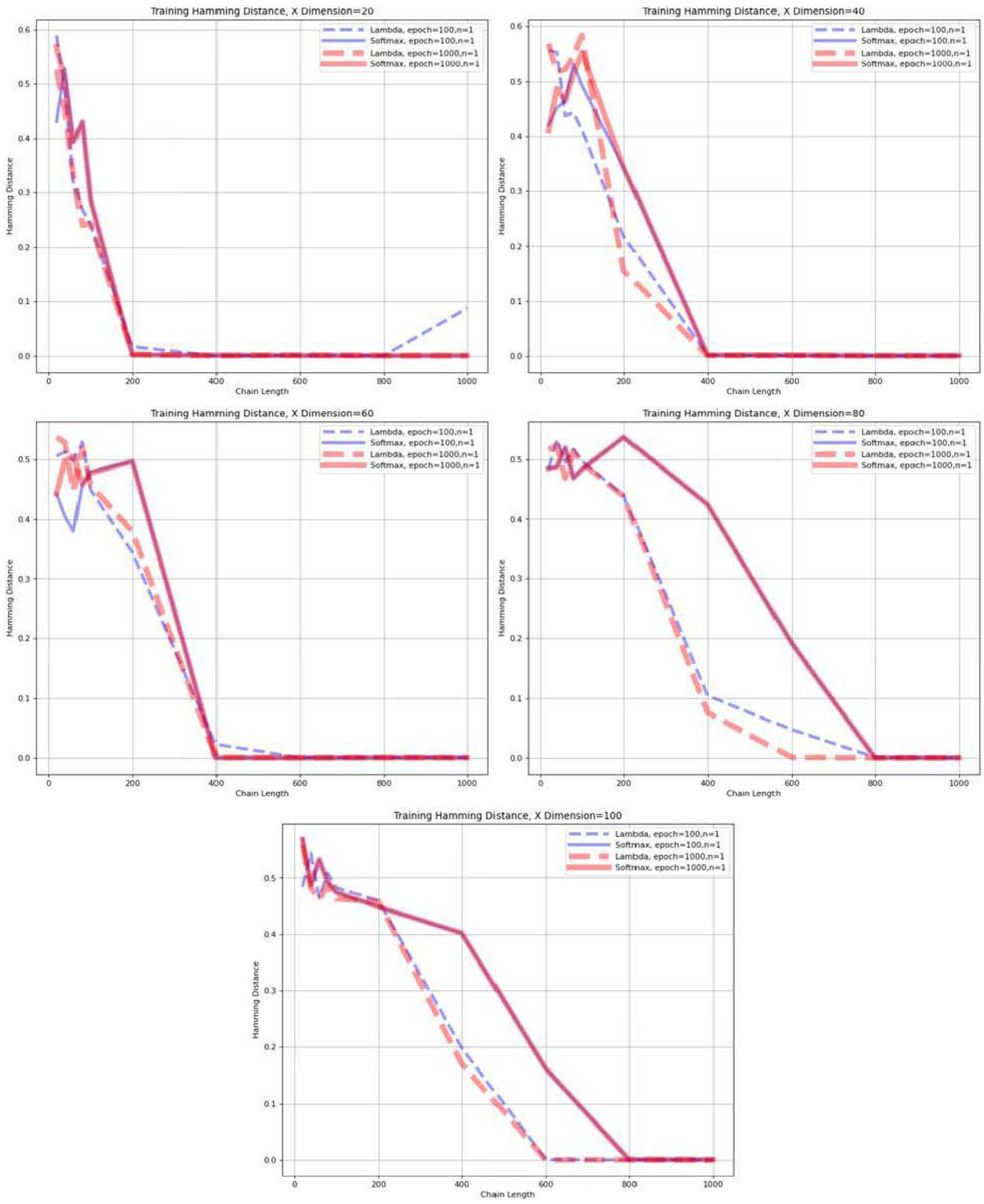


Fig. A.1. Training Hamming distance vs. the chain length. Each plot corresponds to some $d \in [20,40,60,80,100]$.

A.2. Scaling r,k as well as the latent dimension d

Here we consider the case where $r = k$ and observe the impact of larger number of factors as well as higher dimensions in attempt to stress test the model and see where it starts to break. We increased r to 5, 10, 20, 30, and we see that the test Hamming Distance (HD) starts to deviate from zero. However this *break* is rather *graceful*. The settings we explore are as follows: $L = 5000, n = 100$, $d \in [2,5,10,15,20,25,30,35,40,50], r \in [2,5,10,15,20,25,30]$, noise standard deviation σ is 15% of the elements of $\|AX\|$.
Using the lambda method, below is a descriptive summary of our observations (Also see figure A.2):

- $r = 10, d = 10$: HD = 0, all 10 factors are learned after epoch 300.
- $r = 15, d = 10$: HD = 0, all 15 factors are learned after epoch 300.
- $r = 30, d = 10$: HD = 0.1, 26/30 factors are learned after epoch 600.
- $r = 20, d = 20$: HD = 0.04, 18/20 factors are learned after epoch 400.
- $r = 20, d = 30$: HD = 0.05, 18/20 factors are learned after epoch 300.

The fact that Hamming distance for more factors and higher dimensions stays at $0.1 - 0.2$ should not be disappointing, because as we inspected, only several out of 20-30 factors are not learned. So this could be thought of as a good sign that with relatively large number of factors, our algorithm managed to learn the position and the parameters of about %90 of the factors.

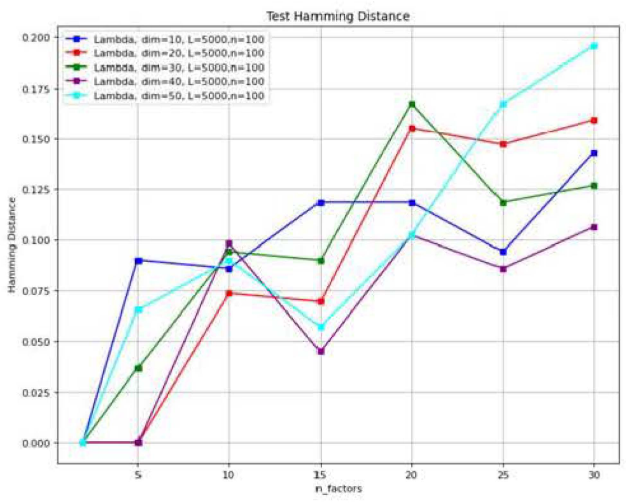


Fig. A.2. Stress testing the proposed method when presented with large latent dimensions in a reusable structure comprising of increasing number of factors. Given the large chain lenght, the metric shows that the algorithm is successfully recovering about %90 of the structure.

A.3. Connection to Expectation-Maximization

Here we show how our Λ approach connects to the EM algorithm. Consider a Markov Chain (MC) of length L. At each time t we have a random vector $\mathbf{X}$ of length p, the concatenation of p random variables. Suppose there are r unique factors generating the observed data, and at each time t a *latent variable* z_t picks a factor $f_i \in F = \{f_1, f_2, \ldots, f_r\}$ that operates on $\mathbf{X}_t$ and generates $\mathbf{X}_{t+1}$. We want to maximize the expected log-likelihood of the observations. Using the Markov property of the chain we have:

$$\mathbb{E}[\text{log-likelihood } (\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_L)] = \mathbb{E}[\log \sum_{z_1} \sum_{z_2} \cdots \sum_{z_L} p_\theta(\mathbf{X}_1, z_1, \mathbf{X}_2, z_2, \ldots, \mathbf{X}_L, z_L)] \tag{A.3.1}$$

$$= \mathbb{E}[\log \sum_{z_1} \sum_{z_2} \cdots \sum_{z_L} p_\theta(\mathbf{X}_1 \mid z_1) p(z_1) p_\theta(\mathbf{X}_2 \mid z_2, \mathbf{X}_1) p(z_2) \ldots p_\theta(\mathbf{X}_L \mid z_L, \mathbf{X}_{L-1}) p(z_L)] \tag{A.3.2}$$

$$= \mathbb{E}[\sum_{t=1}^{L} \log \sum_{z_t} p_\theta(\mathbf{X}_t \mid z_t, \mathbf{X}_{t-1}) p(z_t)] \tag{A.3.3}$$

Denoting $p(z_t)$ by $\pi(z_t)$, our objective would be the following:

$$\mathbb{E}[\text{log-likelihood } (\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_L)] = \mathbb{E}[\sum_{t=1}^{L} \log \sum_{z_t} p_\theta(\mathbf{X}_t \mid z_t, \mathbf{X}_{t-1}) \pi(z_t)] \tag{A.3.4}$$

Now we'll make use of Jensen's inequality. If $f : \mathbb{R} \to \mathbb{R}$ is convex, and $\boldsymbol{X}$ is an integrable R.V.:

$$\mathbb{E}_{\boldsymbol{X}}(f(\boldsymbol{X})) \geq f(\mathbb{E}_{\boldsymbol{X}}(\boldsymbol{X}))$$

If $f : \mathbb{R} \to \mathbb{R}$ is *strictly* convex, we have equality *if and only if* $\boldsymbol{X} = \text{constant}$.
If we introduce a proposal density $q(z)$ (which we will use as an approximator for the posterior $p(z \mid x)$), then our objective will become:

$$\mathbb{E}_{\boldsymbol{X}}[\text{log-likelihood } (\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_L)] = \mathbb{E}_{\boldsymbol{X}}[\sum_{t=1}^{L} \log \sum_{z_t} \frac{p_\theta(\mathbf{X}_t \mid z_t, \mathbf{X}_{t-1}) \pi(z_t)}{q(z)} q(z)] \tag{A.3.5}$$

$$= \sum_{t=1}^{L} \mathbb{E}_{\boldsymbol{X}}[\log \sum_{z_t} \frac{p_\theta(\mathbf{X}_t \mid z_t, \mathbf{X}_{t-1}) \pi(z_t)}{q(z)} q(z)] = \sum_{t=1}^{L} \mathbb{E}_{\boldsymbol{X}}\left[\log \mathbb{E}_{q_t}[\frac{p_\theta(\mathbf{X}_t \mid z_t, \mathbf{X}_{t-1}) \pi(z_t)}{q_t(z_t)}]\right] \tag{A.3.6}$$

Now since log is strictly *concave*, we have the following inequality for the objective:

$$\sum_{t=1}^{L} \mathbb{E}_{\boldsymbol{X}}\big[\log \mathbb{E}_{q_t}[\frac{p_\theta(\mathbf{X}_t \mid z_t,\mathbf{X}_{t-1})\pi(z_t)}{q_t(z_t)}]\big] \geq \sum_{t=1}^{L} \mathbb{E}_{\boldsymbol{X}}\big[\mathbb{E}_{q_t}[\log \frac{p_\theta(\mathbf{X}_t \mid z_t,\mathbf{X}_{t-1})\pi(z_t)}{q_t(z_t)}]\big] = \mathcal{L}(q,\theta) \tag{A.3.7}$$

where $\mathcal{L}(q,\theta)$ is the evidence lower bound (ELBO). In the EM algorithm, we first achieve the equality case of the Jensen's inequality by finding the $\arg\max_q \mathcal{L}(q,\theta) = p_\theta(z \mid x)$, and then we will maximize the ELBO w.r.t. model parameters θ. Note that in EM, we find the argmax of ELBO, and its maximum *exactly*. Now let's revisit the objective of our Λ approach (θ includes the parameters of all factors):

$$\begin{aligned}\mathbb{E}_{\boldsymbol{X}}\big[\log\big[L(\theta, \mathsf{softmax}(\Lambda), \boldsymbol{X})\big]\big] &= \mathbb{E}_{\boldsymbol{X}}\big[\sum_{t=1}^{L-1}\sum_{j=1}^{r} \Lambda^s_{tj} \log\big(\tilde{f}_{\theta_j}(X_t, X_{t+1})\big)\big] \\ &= \sum_{t=1}^{L-1} \mathbb{E}_{\boldsymbol{X}}\big[\sum_{j=1}^{r} \Lambda^s_{tj} \log\big(\tilde{f}_{\theta_j}(X_t, X_{t+1})\big)\big]\end{aligned} \tag{A.3.8}$$

Compare the last equation to ELBO in EM:

$$\sum_{t=1}^{L} \mathbb{E}_{\boldsymbol{X}}\big[\mathbb{E}_{q}[\log \frac{p_\theta(\mathbf{X}_t \mid z_t,\mathbf{X}_{t-1})\pi(z_t)}{q(z)}]\big] = \sum_{t=1}^{L} \mathbb{E}_{\boldsymbol{X}}\big[\sum_{j=1}^{r} q(z_t = j) \log \frac{p_\theta(\mathbf{X}_t \mid z_t,\mathbf{X}_{t-1})\pi(z_t)}{q(z_t = j)}\big] \tag{A.3.9}$$

We should note that $\log\big(\tilde{f}_{\theta_j}(X_t, X_{t+1})\big)$ is a log-likelihood under some factor. Also after E step where $q_t(z_t = j) = p(z_t = j \mid x)$, then $\log \frac{p_\theta(\mathbf{X}_t|z_t,\mathbf{X}_{t-1})\pi(z_t)}{q_t(z_t)} = \log p_{\theta_j}(\mathbf{X}_t \mid \mathbf{X}_{t-1})$ is the log-likelihood under $z_t = j$.

So the connection starts to become clear. The rows of our Λ matrix are the proposal densities q for different time steps. By imposing a softmax over the rows Λ, we constrain the entries in each row to sum up to one. In EM, at each time t, we require that $\sum_{j=1}^{r} q_t(z_t = j) = 1$. In EM, we compute the posterior *exactly*, and replacing that in ELBO, we *exactly* maximize the log-likelihood. However in the Λ approach, we do *not* compute the posterior and log-likelihood exactly. We take gradient *steps* in the proper directions. In implementation of the Λ method, we do both of the E,M steps at the same time, and hence updating with SGD, could be considered as an *approximation* to exact EM. But more interestingly, our ***softmax*** approach which takes the softmax over the *losses* of r different factors at any time t as the weights, amounts to EM where we do the E step *exactly* and only leave the M step to gradient updates. This is because in E step we find the proposal density which satisfies $q_t(z_t = j) = p(z_t = j \mid x)$, and since the posterior $p(z_t = j \mid x)$ is proportional to likelihood

$p_\theta(x,z)$, computing the softmax over negative losses, results in the true posterior under the model parameterized by θ, and hence in this case we have obtained the optimal proposal density q_t at each time. It is noteworthy to mention that our experiments with very long Markov chains demonstrate the effectiveness of the *approximation* of EM with Λ compared to using the softmax approach which does the E step exactly.
Thus while EM gives a more direct intuition on the latent variables of the model, we should keep this observation in mind, so that if required, we can translate them to one another and evaluate the performance of each method.

Appendix B

Supplementary Materials for Object-Centric Causal Representation Learning

B.1. Proof of theorem 1

We want to compare the number of perturbations needed to disentangle shared properties with a standard encoder to those needed by an object-centric encoder. Our strategy will be as follows,

(1) Setup a data generating process with multiple objects where injectivity holds by construction so that we can restate Theorem 1 from Ahuja et al. (2022b) to show they need $k \times d$ perturbations.

(2) Define an object-centric architecture in terms of the object-wise partitions that we defined in Definition 1.

(3) Restate an analog of Theorem 1 from Ahuja et al. based on the object-centric encoder.

(4) Theorem 1 in the main text will follow as a collary of the difference between the number of perturbations used in the two theorems above.

We begin by defining a data generating process such that $g(\mathbf{vec}_\pi(Z))$ is injective by construction. We can achieve this by appending an id, i, to each z_i in Z, such that $Z = \{z_i \oplus [i]\}_{i=1}^k$ where $\oplus$ denotes concatenation, and then choosing $\mathfrak{g}$ such that x depends on i (for example, each i could be rendered in a different color). Like Ahuja et al., we assume we have data that is perturbed by $\Delta := \{\{\delta_{i,j}\}_{j=1}^d\}_{i=1}^k$, a set of 1-sparse perturbations that perturbs each of the d properties from each of the k objects. Taken together, we have the following data generating process (DGP),

$$Z = \{z_i \oplus [i]\}_{i=1}^k \sim \mathbb{P}_Z,\, x := \mathfrak{g}(Z) \quad \tilde{z}_{j,l} := z_j + \delta_{j,l} \,\forall\, \delta_{i,j} \in \Delta, \quad \tilde{x}_{j,l} := \mathfrak{g}(\{z_1, \dots, \tilde{z}_{j,l}, \dots, z_k\}) \tag{B.1.1}$$

where each object has d shared properties, $z_i \in \mathbb{R}^d$, and $z_{i,j}$ and $z_{i',j}$ are of the same type—e.g. position x, hue, etc.— for all j. As before, assume $\mathfrak{g}$ is injective, and define $g^* = g(\mathbf{vec}_{\pi^*}(Z))$ where π^* is the permutation that sorts Z by the index i, so that g^* is injective by construction.

Now, Ahuja et al. show that if the encoder, $\hat{f} : \mathcal{X} \to \mathbb{R}^{kd}$, is chosen to minimize the following loss,

$$\hat{f} \in \arg\min_{f'} E_{x,x',\delta}\left[\left(f'(x) + \delta - f'(x')\right)^2\right] \tag{B.1.2}$$

and the following assumptions hold,

Assumption 2. *The dimension of the span of the perturbations in equation B.1.1 is kd, i.e., $\mathsf{dim}\Big(\mathsf{span}\big(\Delta\big)\Big) = kd$.*

Assumption 3. *$a(z) := f \circ g^*(z)$ is an analytic function. For each component $i \in \{1, \cdots, kd\}$ of $a(z)$ and each component $j \in \{1, \cdots, kd\}$ of z, define the set $\mathcal{S}^{ij} = \{\theta \mid \nabla_j a_i(z+b) = \nabla_j a_i(z) + \nabla_j^2 a_i(\theta) b, z \in \mathbb{R}^{kd}\}$, where b is a fixed vector in $\mathbb{R}^{kd}$. Each set $\mathcal{S}^{ij}$ has a non-zero Lebesgue measure in $\mathbb{R}^{kd}$.*

Then we have,

Theorem 2 ((Ahuja et al., 2022b)). *Assume we have data from the DGP in equation B.1.1 and assumption 2 and 3 hold and the number of perturbations per example equals the latent dimension, $m = kd$, then the encoder that solves equation B.1.1 identifies true latents up to permutation and scaling, i.e. $\hat{z} = \Pi\Lambda z + c$, where $\Lambda \in R^{kd \times kd}$ is an invertible diagonal matrix, $\Pi \in R^{kd \times kd}$ is a permutation matrix and c is an offset.*

Proof. See Ahuja et al. for the proof. □

Now, consider an object-centric architecture encoder of the form $F(x) := \{\mathfrak{f}(x^i)\}_{x^i \in P}$ where P is an object-wise partition and $\mathfrak{f} : \mathcal{X} \to \mathbb{R}^d$. Let $\sigma \in \Sigma$ denote a permutation of the latents from the set of all $k-$permutations. Let:

$$\hat{F}(x) \in \arg\min_{\mathfrak{f}} E_{x,x',\delta}[\min_{\sigma' \in \Sigma} \|((\mathfrak{f}(x')^{\sigma'(i)}) - ((\mathfrak{f}(x)^{(i)}) + \delta)\|^2] \tag{B.1.3}$$

Note that since δ is non-zero for only one pair of patches $x^{(i)}, x^{(i)'}$ and zero otherwise, the minimizer over Σ is almost surely unique. Assumptions 4 and 5 are analogs of 2 and 3 above, but make reference to the dimensionality of the co-domain of $\mathfrak{f}$ rather than f.

Assumption 4. *The dimension of the span of the perturbations in equation B.1.1 is d, i.e., $\mathsf{dim}\Big(\mathsf{span}(\Delta)\Big) = d$.*

Assumption 5. *$a(z) := \mathfrak{f} \circ g^*(z)$ is an analytic function. For each component $i \in \{1, \cdots, d\}$ of $a(z)$ and each component $j \in \{1, \cdots, d\}$ of z, define the set $\mathcal{S}^{ij} = \{\theta \mid \nabla_j a_i(z+b) = \nabla_j a_i(z) + \nabla_j^2 a_i(\theta) b, z \in \mathbb{R}^d\}$, where b is a fixed vector in $\mathbb{R}^d$. Each set $\mathcal{S}^{ij}$ has a non-zero Lebesgue measure in $\mathbb{R}^d$.*

With this setup, the following theorem follows directly from Theorem 2 as a reduction from the multi-object to single-object setting,

Theorem 3. *Assume we have data from the DGP in equation B.1.1 and assumption 4 and 5 hold and the number of perturbations per example equals the latent dimension, $m = d$, then the encoder that solves equation B.1.3 for an object-wise partition P, identifies true latents up to permutation and scaling, i.e. $\hat{z} = \Pi\Lambda z + c$, where $\Lambda \in R^{kd \times kd}$ is an invertible diagonal matrix, $\Pi \in R^{kd \times kd}$ is a permutation matrix and c is an offset.*

Proof. Because P is an object-wise partition, the function that produces each $x^{(i)} \in P$ is injective with respect to some $z_{\sigma(i)}$ (i.e. one of the object's latents). Thus for each $x^{(i)}$, the solution to equation B.1.3 is equivalent to the single object setting with $k = 1$, and thus theorem 2 applies, which implies that $\mathfrak{f}(x^{(i)}) = \hat{z}_i = \Pi\Lambda z_i + c$ for all i. Now let $\hat{z} = \mathbf{vec}_\pi(\{\hat{z}_i\}_{i=1}^k)$. Because each $\hat{z}_i$ is identified up to a permutation, scaling and offset, and for any π, there exists a Π such that $\hat{z} = \Pi\Lambda z + c$ which completes the result. □

Corollary 1. *If the assumptions for Theorem 2 and 3 hold and a data generating process outputs observations containing k objects with shared properties, then an object-centric architecture of the form $F(x) := \{\mathfrak{f}(x^{(i)}\}_{x^{(i)} \in P}$ will disentangle in $1/k$ fewer perturbations than an encoder of the form $f : \mathcal{X} \to \mathbb{R}^{kd}$.*

Proof. This follows directly from comparing the the number of perturbations required in Theorems 2 and 3. □

B.2. Background on slot-attention-based architectures

Slot attention (Locatello et al., 2020b) is a neural network component that, intuitively, summarizes the relevant information in the input set (most commonly, image features with position embeddings) into a smaller set of so-called "slots". Each slot is a feature vector that can be thought of as capturing information about one "object" in the input set, which usually comprises multiple elements of the input set. This is done by repeating cross-attention between the inputs and the slots to compute per-slot updates.

In the traditional set-up, these slots are then used to reconstruct the input with an auto-encoder objective: each slot is decoded into a separate image through a shared image decoder, which is followed by merging these per-slot images into a single reconstructed image. Ideally, slot attention is able to decompose the original image into distinct objects, each of which is modeled by a single slot.

More concretely, slot attention takes as input a matrix $\mathbf{X} \in \mathbb{R}^{n \times c}$ with n as the number of inputs and c the dimensionality of each input. We also randomly initialize the slots $\mathbf{Z}^{(0)} \in \mathbb{R}^{m \times d}$. We start by computing the query, key, and value matrices as part of cross-attention.

$$\mathbf{Q}^{(t)} = \mathbf{Z}^{(t)}\mathbf{W}_Q \quad \mathbf{K} = \mathbf{X}\mathbf{W}_K \quad \mathbf{V} = \mathbf{X}\mathbf{W}_V \tag{B.2.1}$$

This is followed by the normalized cross-attention to determine the attention map $\mathbf{A} \in \mathbb{R}^{m \times n}$, then a GRU to apply this update to the slots.

$$\mathbf{A}^{(t)} = \text{normalize}(\mathbf{Q}^{(t)}\mathbf{K}^\top) \tag{B.2.2}$$

$$\mathbf{Z}^{(t+1)} = \text{GRU}(\mathbf{Z}^{(t)}, \mathbf{A}^{(t)}\mathbf{V}) \tag{B.2.3}$$

The function normalize encourages slots to compete for inputs by applying a softmax over slots and normalizing the weights for each input to sum to one. After T steps, the algorithm outputs $\mathbf{Z}^{(T)}$, a set of m embedding vectors $\{\mathbf{z}_i^{(T)}\}_{i=1}^m$ that can be used as input to a shared image decoder.

SA-MESH.. The specific version of slot attention we use in this paper is SA-MESH (Zhang et al., 2023). It makes regular slot attention more powerful by giving it the ability to break ties between slots more effectively. [1] In practice, this improves the quality of the individual slot representations significantly due to less mixing of unrelated inputs into the same slot.

The key difference with regular slot attention is that it features an entropy minimization procedure to approximate an optimal transport solution, which makes the attention map sparse. The connection to optimal transport is made by the use of the standard Sinkhorn algorithm (Sinkhorn and Knopp, 1967; Cuturi, 2013).

$$\text{MESH}(\mathbf{C}) = \arg\min_{\mathbf{C}} H(\text{sinkhorn}(\mathbf{C})) \tag{B.2.4}$$

$$\mathbf{A}^{(t)} = \text{sinkhorn}(\text{MESH}(\mathbf{Q}^{(t)}\mathbf{K}^\top)) \tag{B.2.5}$$

The optimization problem is solved by unrolling gradient descent, with a noisy initialization to ensure that ties are broken.

B.3. Alternative perturbation mechanisms

Dense vs. Sparse. We can have a number of assumptions on the perturbation mechanisms and the nature of model's knowledge about those mechanisms. In the most general case, suppose $\mathcal{M} = \{m^1(\cdot), m^2(\cdot), \ldots, m^k(\cdot)\}$ denotes the set of all possible perturbation mechanisms. To obtain x', the perturbed variant of x, we then select a subset of $k' \leq k$ objects as targets that undergo perturbations determined by a subset of k' mechanisms $\mathcal{M}' \subset \mathcal{M}$ from the set of all possible perturbation mechanisms. The correspondence between the k' mechanisms and k' perturbed objects is decided by a random permutation $\pi_t^{\mathcal{M}}$, i.e. $i = \pi_t^{\mathcal{M}}[j]$ means that mechanism i governs the transition dynamics of object j to produce z_{t+1}^j (for objects

[1]Concretely, it makes the mapping from the initial slots to the final slots *exclusively multiset-equivariant* (Zhang et al., 2022a) rather than permutation-equivariant/set-equivariant.

that are not supposed to change from $t \to t+1$ a dummy mechanism with index -1 can be assumed which results in no change). A mechanism $m^i(\cdot) : \mathbb{R}^d \to \mathbb{R}^d$ in $\mathcal{M}$ is a vector-valued function that operates on object-wise true latents z_t^j and outputs $z_{t+1}^j = z_t^j + \delta_i$ such that $i = \pi_t^{\mathcal{M}}[j]$. Perturbation vectors δ_i could be sparse or not. The subset $\mathcal{M}'$ can contain $k' = k$ mechanisms to perturb all of the k objects in the environment, and if none of the $k' = k$ resulting perturbations is sparse, we denote the set $\mathcal{M}'$ as *fully dense perturbations*, i.e., all of the properties of all objects will change from $t \to t+1$. $\mathcal{M}'$ can also contain at least one object but not all of them ($1 \leq k' < k$) with sparse or dense perturbations, or it may consist of only a single object ($k' = 1$) that is perturbed by a fully sparse mechanism, one that only alters a single property and leaves the rest unchanged. We denote this scenario as *fully sparse perturbation.*

B.4. Matching

Perturbations alter the properties of objects from $t \to t+1$ and the model has to infer which object's properties were perturbed to update its representations and minimize the latent loss equation 3.2.1. But recall that the model has no direct access to objects. It receives the observations at t,$t+1$ and encodes each of them to a set of slots $\mathcal{S}_t, \mathcal{S}_{t+1}$. These slots do not follow any fixed ordering, and moreover, there is no guarantee that each slot binds to exactly one unique object. Slots can also correspond to the background. Each perturbation δ_t^j changes the properties of some object z_t^i, so the model requires to find a pair of slots (s_t^u, s_{t+1}^v) that are bound to object z^i at t and $t+1$, respectively. Once the model figures out such a matching, then the latent loss that results in disentanglement can be computed via the projections of these slots $\hat{z}$:

$$\mathcal{L}_z = \sum_{i=1}^{m} \|\hat{z}_t^i + \delta_t^{\pi_t^{\mathcal{M}'}[i]} - \hat{z}_{t+1}^i\|^2, \qquad \hat{z}_t^i = \hat{f}_z(s_t^u),\ \hat{z}_{t+1}^i = \hat{f}_z(s_t^v) \tag{B.4.1}$$

The problem of finding a correspondence between slot projections at t,$t+1$ and the perturbations is an instance of the 3-dimensional matching. We can use the following methods to solve this problem.

Hungarian Matching. If the changes from $t \to t+1$ are not dramatic, or the scene is not composed of exactly identical objects (same shape and color), then empirically we observe that more often than not, initializing the slots identically at t,$t+1$ results in sets of slots that preserve the ordering from $t \to t+1$. When this assumption is valid for many samples, our problem reduces to a bipartite matching of perturbations and slots for which the order does not change for two consecutive steps. This bipartite matching can be accomplished via the Hungarian algorithm. Concretely, the cost matrix required by the Hungarian algorithm C_H is an $m \times |\mathcal{S}|$ matrix composed as the following:

$$C_H[i,j] = \|\hat{f}_z(s_t^j) + \delta_t^{\pi_t^{\mathcal{M}'}[i]} - \hat{f}_z(s_{t+1}^j)\|^2 \tag{B.4.2}$$

i.e., each row consists of the squared errors of applying one mechanism (corresponding to that row) to each slot (the columns). Hungarian algorithm finds an optimal assignment π_t^* of slot projections and mechanisms such that $m^i(\cdot)$ is applied to MLP$(s_t^{\pi_t^*[i]})$ $\pi_t^*[i]$ for $i \in 1, \dots, m$. Then the latent loss is computed as follows:

$$\mathcal{L}_z = \sum_{i=1}^{m} \|\hat{f}_z(s_t^{\pi_t^*[i]}) + \delta_t^i - \hat{f}_z(s_{t+1}^{\pi_t^*[i]})\|^2 \tag{B.4.3}$$

Note that in equation B.4 the summation index runs over object indices, but in equation B.4 it runs over the mechanism indices (same range, but slightly different meaning.)

Double Matching via Constrained Linear Programs (CLP). Although Hungarian matching can work in many situations, there exist some cases where the assumption of preserved order of slots from t to $t+1$ does not hold anymore. For instance in situations where there exist a lot of symmetries (i.e., same color for all objects), then slots' binding to object will face a higher degree of randomness, and thus, using Hungarian matching would result in very noisy gradients that hinder convergence. Or when the perturbations are not very local, i.e. two relatively distant objects swap their positions from $t \to t+1$, then we can no longer assume that slots obtained at $t+1$ reflect the same binding to objects as slots that were obtained at t.

In such situations we resort to a more accurate matching scheme that significantly reduces the noise slot-object bindings and speeds up convergence drastically for these corner cases. This method deals with the more difficult problem of 3-dimensional matching, and uses slots at both t,$t+1$ to find the assignments, hence the name *double matching*. Recall that $\pi_t^s, \pi_t^{\mathcal{M}'}$ relate slots in $\mathcal{S}_t$ and mechanisms to the objects in $\mathcal{Z}_t$, respectively. Thus, the model at each step is required to jointly solve for these permutations at t,$t+1$ to minimize the following:

$$(\pi_t^s, \pi_{t+1}^s, \pi_t^{\mathcal{M}'})^* = \underset{\pi_t^s, \pi_{t+1}^s, \pi_t^{\mathcal{M}'}}{\arg\min} \sum_{i=1}^{m} \|\hat{f}_z(s_t^{\pi_t^s[i]}) + \delta_t^{\pi_t^{\mathcal{M}'}[i]} - \hat{f}_z(s_{t+1}^{\pi_{t+1}^s[i]})\|^2 \tag{B.4.4}$$

Notice that we are effectively finding the correspondence between perturbations $\mathcal{M}'$ and *pairs* of slots (s_t^i, s_{t+1}^j) for $i,j \in [\mathbf{1} : |\mathcal{S}|]$, such that the pair of slots correspond to the same object as the one that is perturbed by the assigned mechanism. To find such assignments we could construct an $m \times |S|^2$ cost matrix ($|S|^2$ denotes all the possible pairs of (s_t^i, s_{t+1}^j)) as follows:

$$C_{\mathrm{CLP}}[i,j] = \|\hat{f}_z(s_t^{k_t(j)} + \delta_t^i - \hat{f}_z(s_{t+1}^{k_{t+1}(j)})\|^2, \qquad i \in [\mathbf{1} : m], j \in [1 : |S|^2] \tag{B.4.5}$$

$$k_t(j) = \lfloor j/|S| \rfloor \tag{B.4.6}$$

$$k_{t+1}(j) = \mathrm{mod}(j, |S|) \tag{B.4.7}$$

However, the assignment cannot be recovered by Hungarian matching alone. The reason is that there are constraints that need to be satisfied for a matching in this scenario to be valid. Note that for each row i (mechanism), the matched column index j determines the pair of slots that correspond to the same object at t,$t+1$, which is perturbed by mechanism i. Such assignments have to satisfy the following constraints:

- Selected js for all rows should be such that no slot at time t is selected more than once as the first element of any slot pair, i.e. a slot cannot be the subject of two perturbations at any given t because each object is affected by one and only one mechanism.
- Selected js for all rows should be such that similarly, no slot at time $t+1$ is selected more than once as the second element of any slot pair, i.e. a slot cannot be the outcome of two perturbations at any given $t+1$ because again, each object is affected by one and only one mechanism.

Since any matching has to fulfill these constraints, it can no longer be treated as a simple bipartite matching solvable by the Hungarian algorithm. But we can still find an assignment as the solution of a constrained linear program (LP). We can define a binary $m \times |S|^2$ weight matrix W such that when multiplied element-wise by C_{CLP}, masks all the entries that do not correspond to the matching by zero, and leaves the entries corresponding to the matching unchanged. Thus we can find the assignments simply by looking at the non-zero entries. So to summarize, the matching could be found by solving the following Constrained LP:

$$\text{minimize} \sum_{i=1}^{m} \sum_{j=1}^{|\mathcal{S}|^2} W \odot C_{\text{CLP}} = \sum_{i=1}^{m} \sum_{j=1}^{|\mathcal{S}|^2} C_{\text{masked}} \tag{B.4.8}$$

$$\text{subject to} \quad W[i,j] \in \{0,1\} \quad \forall i,j \tag{B.4.9}$$

$$\sum_{i=1}^{m} \sum_{j \in \{k, k+|S|, k+2\|S|, \ldots\}} C_{\text{masked}}[i,j] = 1 \quad \forall k \in [1 : |S|] \tag{B.4.10}$$

$$\sum_{i=1}^{m} \sum_{j \in \{k|S|, k|S|+1, k|S|+2, \ldots\}} C_{\text{masked}}[i,j] = 1 \quad \forall k \in [1 : |S|] \tag{B.4.11}$$

$$\sum_{j=1}^{|\mathcal{S}|^2} C_{\text{masked}}[i,j] = 1 \quad \forall i \tag{B.4.12}$$

Equations B.4.10, B.4.11 make sure the constraints mentioned above are satisfied, and the last constraint makes sure each mechanism is exactly assigned to one pair only. Although solving this CLP provides an exact solution to our matching problem, we do not opt for a binarized W as it will result in a mixed-integer CLP, which is NP-hard, and in practice would become intractable fairly quickly as the number of slots increases ($|S|^2$ dependence). Hence we will relax the constraint of equation B.4.9 to $0 \leq W[i,j] \leq 1 \, \forall i,j$ to avoid any mixed-integer situation in the program. It is noteworthy to mention that although the relaxed CLP is significantly faster than the binarized version, it is still much slower than the Hungarian matching, despite our efforts to implement the constraints and the objective as parallel-friendly as possible. The bottleneck results from the constraints that we have introduced, but this is the price we need to pay to overcome those particularly hard cases with significant symmetries in the scene that otherwise could not be dealt with. As a matter of fact, even running Hungarian matching in those situations can still reasonably guide the latent representations toward disentanglement but it is for the evaluation of the predicted properties of *all* objects $\hat{z}$ against their true properties z that we absolutely require double matching via CLPs.

Matching Fully Sparse Perturbations. Fully sparse perturbations not only tend to be a more realistic choice than fully dense ones, but also they result in a much easier matching scheme which significantly improves the efficiency of our method. When the model is presented with fully sparse perturbations, then the 3-dimensional matching reduces to finding the minimum element in a $1D$ array of size $|\mathcal{S}|^2$, where $|\mathcal{S}|$ is the number of slots. The reason is that since only one property of a single object z_t^i is being altered (and the model knows perturbations are fully sparse), the model can apply that 1-sparse perturbation δ_t (or the transformed version in the unknown setting) to all of slot projections $\hat{z}_t^j$ for $j \in \{1, \dots, |\mathcal{S}|\}$ at time t, and *only* find one pair of slots from the set of $|\mathcal{S}|^2$ possible pairs at t,$t+1$, which correspond to the perturbed object:

$$(i,j) = \arg\min_{i',j'} \|\hat{z}_{t+1}^{j'} - (\hat{z}_t^{i'} + \delta_t)\|^2 \tag{B.4.13}$$

However, this minimization can be made simpler if we use the slots obtained at t to initialize the slots at $t+1$. This way, in practice, the order of slots at t,$t+1$ is very likely to be preserved, not only since the slots at $t+1$ are initialized with hints from t, but also the sparse perturbation makes it much easier for all slots to bind to the same object as only a small subset of the scene needs to be readjusted among the slots. Therefore the matching would reduce to a simple minimization over $|\mathcal{S}|$ elements:

$$i = \arg\min_{i'} \|\hat{z}_{t+1}^{i'} - (\hat{z}_t^{i'} + \delta_t)\|^2 \tag{B.4.14}$$

Note however that we still would use all slot projections for evaluation, the only difference with this matching scheme is that gradient signals are only propagated from the perturbed object (as they also should, since there is no change in other slots, there is nothing there to be learned that helps disentanglement.)

B.5. Further Experimental Results

B.5.1. 2D Shapes

Tables B.1,B.2 extend our results under unknown fully sparse perturbations on the 2D shapes dataset to more combinations of disentanglement target properties. We can observe that our results stay very close to the upper bound on the achievable performance which uses a *supervised* linear regression from slot projections $\hat{z}$ to *ground truth* latents z. These tables highlight once again how big of a role the injectivity assumption plays in achieving identification with conventional encoders that ignore the object-centricity of the environment (see the performance drop from CNN† to CNN, where the latter drops the unrealistic injectivity assumption).

Table B.1. Linear Disentanglement (LD) scores on 2D shapes test set under *unknown* fully sparse perturbations. All results are averaged over 3 seeds except those requiring to train SA-MESH from scratch that were trained only once. SA-LR achieves a score of 1.0 in all settings.

	p_x,p_y,color			p_x,p_y,shape		
Model	$n = 2$	$n = 3$	$n = 4$	$n = 2$	$n = 3$	$n = 4$
Ours	**1.00** $\pm$0.01	**0.98** $\pm$0.01	**0.99** $\pm$0.00	**1.00** $\pm$0.01	**0.98** $\pm$0.01	**0.99** $\pm$0.01
SA-RP	0.77	0.61	0.60	0.71	0.68	0.70
SA-PC	0.97	0.98	0.99	0.80	0.66	0.87
CNN†	1.00 $\pm$0.00	0.99 $\pm$0.01	0.98 $\pm$0.00	1.00 $\pm$0.00	1.00 $\pm$0.00	0.99 $\pm$0.00
CNN	0.35 $\pm$0.00	0.15 $\pm$0.00	0.07 $\pm$0.01	0.32 $\pm$0.01	0.15 $\pm$0.01	0.11 $\pm$0.01

	p_x,p_y,color,shape		
Model	$n = 2$	$n = 3$	$n = 4$
Ours	**0.99** $\pm$0.00	**0.98** $\pm$0.01	**0.99** $\pm$0.00
SA-RP	0.69	0.73	0.60
SA-PC	0.74	0.75	0.52
CNN†	1.00 $\pm$0.00	0.99 $\pm$0.01	1.00 $\pm$0.00
CNN	0.40 $\pm$0.00	0.21 $\pm$0.00	0.11 $\pm$0.00

B.5.2. 3D Shapes

Quantitative Results. Tables B.3,B.4 extend our results under unknown fully sparse perturbations on the 3D shapes dataset to more combinations of disentanglement target properties. Again, we can observe the applicability of our method to this more complex 3D dataset that contains artifacts related to depth, occlusion, and lighting, to name a few. Again our results stay very close to the upper bound on the achievable performance which uses a *supervised* linear regression from slot projections $\hat{z}$ to *ground truth* latents z.

Qualitative Results. Figures B.1-B.3 illustrate the learned disentangled (object-centric) representations. Each figure shows a sequence of 3D samples evolving over 5 steps (shown on the left), and how the learned representations respond to the perturbations (shown on the right). Perturbations include changing the object's p_x, p_y, colour, ϕ (rotation). The model used here is trained with 3 slots, and the learned representations are the result of a projection layer learned through our weakly-supervised method applied to 64−dimensional

Table B.2. Permutation Disentanglement (MCC) scores on 2D shapes test set under *unknown* fully sparse perturbations. All results are averaged over 3 seeds except those requiring to train SA-MESH from scratch that were trained only once. SA-LR achieves a score of 1.0 in all settings.

	p_x,p_y,color			p_x,p_y,shape		
Model	$n=2$	$n=3$	$n=4$	$n=2$	$n=3$	$n=4$
Ours	**1.00** ±0.01	**0.95** ±0.05	**0.97** ±0.02	**0.99** ±0.01	**0.99** ±0.01	**0.99** ±0.01
SA-RP	0.74	0.60	0.60	0.66	0.63	0.59
SA-PC	0.87	0.89	0.90	0.83	0.81	0.89
CNN†	1.00 ±0.00	0.99 ±0.01	0.99 ±0.01	1.00 ±0.00	1.00 ±0.00	0.99 ±0.00
CNN	0.55 ±0.01	0.35 ±0.01	0.24 ±0.01	0.52 ±0.02	0.33 ±0.02	0.28 ±0.02

	p_x,p_y,color,shape		
Model	$n=2$	$n=3$	$n=4$
Ours	**0.99** ±0.01	**0.98** ±0.01	**0.99** ±0.01
SA-RP	0.54	0.68	0.55
SA-PC	0.64	0.63	0.57
CNN†	1.00 ±0.00	0.99 ±0.01	1.00 ±0.00
CNN	0.61 ±0.00	0.43 ±0.00	0.30 ±0.00

Table B.3. Linear Disentanglement (LD) scores on 3D shapes test set under *unknown* fully sparse perturbations. All results are averaged over 3 seeds except those requiring to train SA-MESH from scratch that were trained only once. SA-LR achieves a score of 1.0 in all settings.

	p_x,p_y,size			p_x,p_y,color,rotation		
Model	$n=2$	$n=3$	$n=4$	$n=2$	$n=3$	$n=4$
Ours	**0.98** ±0.01	**0.98** ±0.01	**0.98** ±0.00	**0.98** ±0.00	**0.97** ±0.01	**0.98** ±0.01
SA-RP	0.61	0.62	0.53	0.59	0.54	0.55
SA-PC	0.78	0.84	0.78	0.70	0.72	0.69

object-centric representations. The projection maps each slot from $\mathbb{R}^{64} \to \mathbb{R}^{4}$, i.e., the disentanglement target space. In figures B.1-B.3, the 4 dimensions of such projections for object slots are presented over 5 steps, i.e., each set of coloured lines shows the evolution

Table B.4. Permutation Disentanglement (MCC) scores on 3D shapes test set under *unknown* fully sparse perturbations. All results are averaged over 3 seeds except those requiring to train SA-MESH from scratch that were trained only once. SA-LR achieves a score of 1.0 in all settings.

	p_x,p_y,size			p_x,p_y,color,rotation		
Model	$n = 2$	$n = 3$	$n = 4$	$n = 2$	$n = 3$	$n = 4$
Ours	**0.96** $\pm$0.02	**0.96** $\pm$0.05	**0.96** $\pm$0.03	**0.98** $\pm$0.01	**0.98** $\pm$0.02	**0.97** $\pm$0.01
SA-RP	0.60	0.57	0.52	0.55	0.51	0.49
SA-PC	0.87	0.90	0.86	0.73	0.76	0.74

of the projection of a slot corresponding to the object with the same colour. Please refer to the figures for details of the perturbations. Lastly, we have kept the number of objects in these scenes to two for clarity of the presentation, however tables 3.3, 3.4, B.3, B.4 show that we achieve similar performances with other sets of properties and number of objects in the scene.

B.5.3. Comparison of Sample Efficiency

Figure B.4 demonstrates the sample efficiency of our object-centric model compared to a ResNet that achieves disentanglement with an injective DGP. Both models are trained with varying number of training samples that contain $n = 4$ objects for which p_x,p_y,colour,shape are the disentanglement target properties. Since we sample 1-sparse perturbations uniformly, the training dataset size could be thought of as a proxy for the number of different perturbations a given configuration of objects would encounter. Although according to theory, the injective ResNet should require *at least* n times more perturbations to identify the latents up to affine transformations, we observe that the advantage of our object-centric model in terms of sample efficiency is much more pronounced in practice. Our method can achieve close to perfect disentanglement with as few as 100 training samples, while an injective ResNet takes 100 times more samples to raise to a comparable performance. This highlights the practical importance of exploiting the inherent set structure of objects in a scene for representation learning.

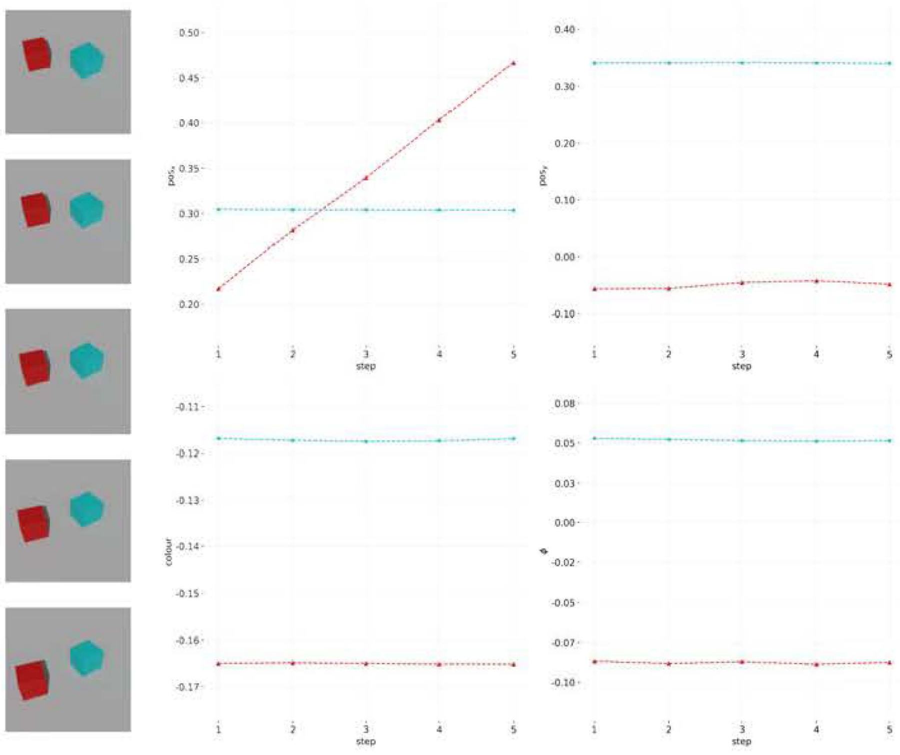


Fig. B.1. *(Left)* From top to bottom, each steps perturbs the p_x coordinate of the red object by 0.2 in the *ground-truth latent space*, *(Right)* which is reflected (through an affine transform) as a linear increase in p_x, while the rest of the properties for both objects remain the same, demonstrating that the learned representations are indeed disentangled. In generating these samples, camera has some non-zero angle w.r.t. the origin, therefore, perturbations in the x direction appear as vertical displacements.

B.6. (Known) Dense Perturbations

We consider the known perturbations when the vectors δ_t^i for all objects are fully dense. Known sparse perturbations are just a simpler instance than unknown sparse perturbations for which we presented successful results in section 3.7. Tables B.5, B.6 show the performance of our model compared to the baselines when the disentanglement target properties are objects' p_x,p_y coordinates. Table B.5 is a particularly challenging case where all properties other than p_x,p_y are kept fixed, even the colour. Therefore, the objects look completely identical and are only placed in different parts of the scene and the model should identify the true positional latents up to irrelevant transformations based on fully dense perturbations. Such perturbations could totally alter the object arrangements from t to $t+1$, so we cannot

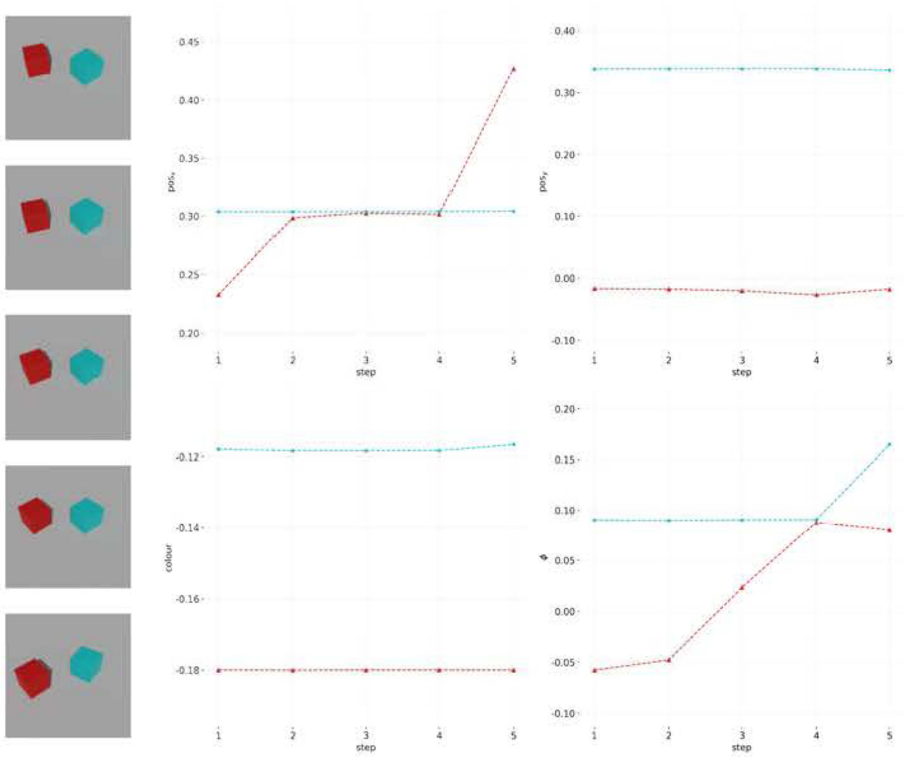


Fig. B.2. *(Left)* From top to bottom, the objects are perturbed as follows; Red: Downward perturbation so it sits at the same x coordinate as the Blue cube (1-2), Rotates counterclockwise along the $z-$axis to be at the same orientation as the Blue cube (2-4), Further downward perturbation (2 times the displacement from step 1 to 2). Blue: Rotates counterclockwise along the $z-$axis (4-5). *(Right)* Note how the learned representations mapping correctly reflects the similar positions and rotations in the ground-truth, i.e., both by having properties of objects coincide at the same value, and by preserving the ratio of perturbations.

rely on Hungarian matching with approximate gradients to recover the true latents as it results in a very unstable training. Therefore, this situation highlights how identical objects are not just a theoretical inconvenience and how a set treatment of these objects results in disentanglement. Using the constrained linear program defined in B.4 and the disentanglement procedure from (Ahuja et al., 2022b) we can identify the positional properties as shown in table B.5.

In table B.6, we show the results for when objects are allowed to only differ in colour. This is a simpler case where if we order objects according to some ordering (not a realistic assumption), then an injective ConvNet could recover the true latents (see the row for CNN†).

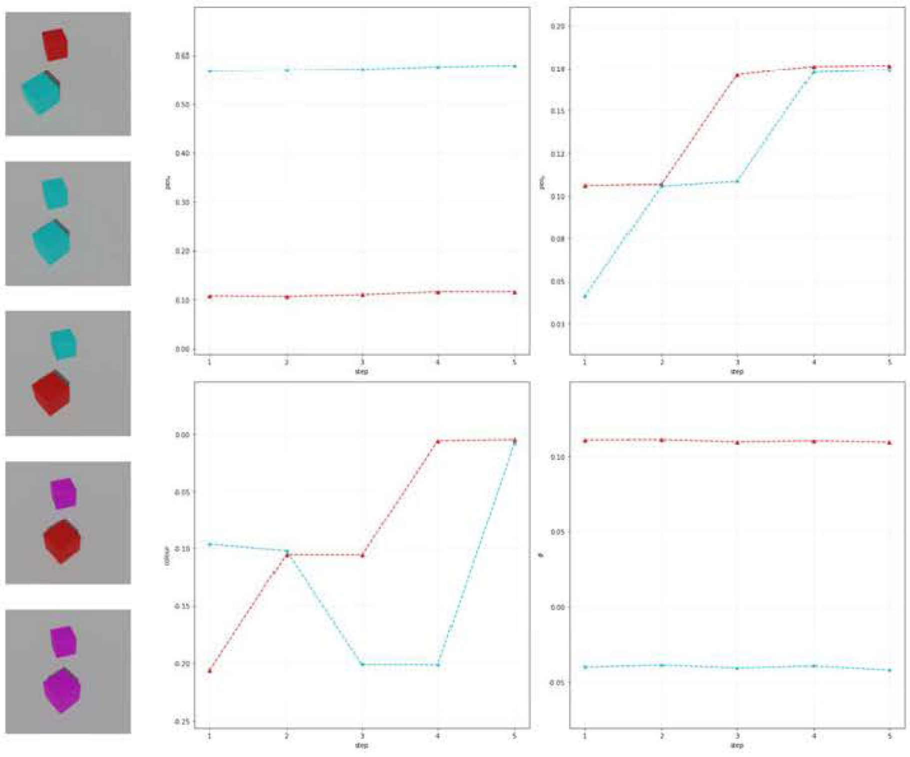


Fig. B.3. *(Left)* From top to bottom, the objects are perturbed as follows; Since the objects change colour, let us call the red cube in the top frame to be object **1**, and the blue one to be object 2. Object 1: The colour is perturbed to be equal to object 2 (1-2), Moves to the right as its p_y is perturbed by 0.2 (2-3), The colour hue is once again perturbed to become purple (3-4). Object 2: Moves toward right (perturbation in p_y by 0.2) to be at the same y coordinate as object 1 (1-2), The colour hue is *decreased* so now the colours of objects 1,2 are swapped (2-3). Moves by 0.2 in the y direction to align with the other object once again (3-4), Change its colour hue twice the previous colour perturbation with the opposite sign to match the colour of the other object (4-5). *(Right)* Red curves correspond to object 1, and the blue curves correspond to object 2. Again, notice the sections where the curves coincide, as well as the ratio of jumps in the properties, showing consistency of the learned representations with the ground-truth causal representation that gives rise to these observations.

Although we show our method works under (known) dense perturbations, there exist a number of sources of non-identifiability which render this scenario less feasible, in addition to the fact that the assumption of having all objects change properties is not so realistic. Below is a

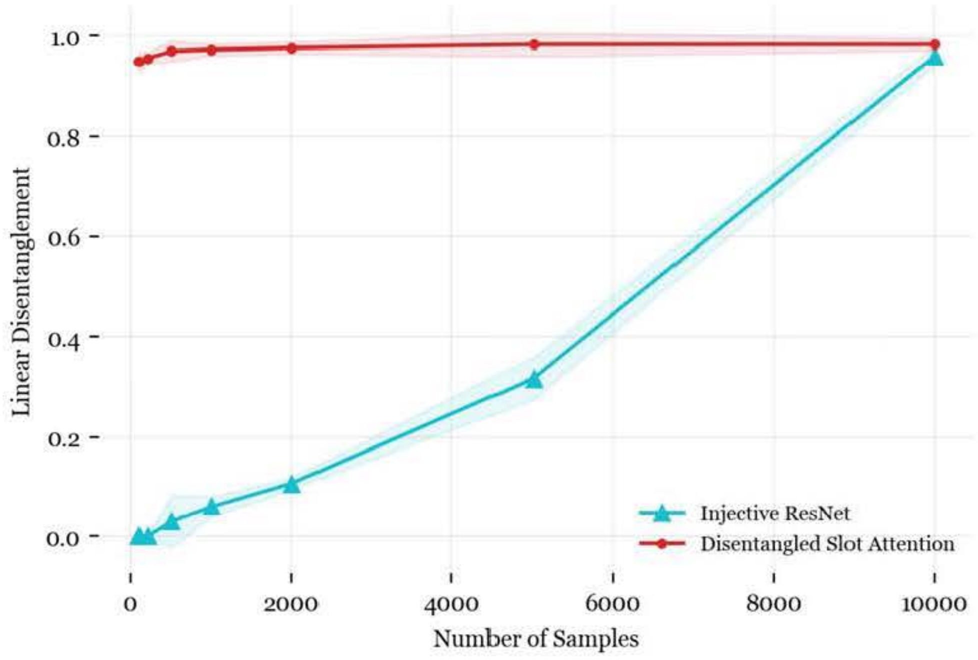


Fig. B.4. Comparing the disentanglement performance of an injective ResNet vs. our object-centric method based on the number of training samples. The dataset contains four 2D objects in which p_x,p_y,colour,shape can vary.

number of scenarios in which the matching has no way of identifying the correct assignment and fails:

- If two or more perturbations are identical or have close norms, then the matching procedure has no way of assigning the correct perturbation to the correct object's slot, which results in obtaining wrong gradients and hinders identification. The sensitivity of the matching procedure depends on the dimensionality of the perturbations, as well as the permissible perturbation steps (i.e., for discrete properties). Therefore, further fine-tuning of the cost matrix is required to prevent numerical issues when two or more norms are close. Identical perturbations which are more likely in lower dimensions and higher number of objects would still be impossible to distinguish.
- Due to the same reason, objects that are not perturbed at all from t to $t+1$ would be confused because they induce the same $\delta = 0$ in the cost matrix. There exist ways to tackle this problem such as identifying the objects that remain the same using slot attention decoder masks, and then removing all such slots from the matching procedure. However, such tricks would only make the method more complex, and

are tangential to the point we wish to make in this study. With sparse perturbations however, we do not care about any zero mechanisms as we are only after a single pair that minimizes the prediction error norm.

- Since the number of slots is arbitrary and always more than the number of objects, it is possible that more than one slot binds to the same object. A duplicate slot can confuse the matching procedure into assigning two different perturbations (corresponding to two different objects) to two duplicate slots that bind to the same object. This is because the cost matrix only operates on the norm of differentials. Such incorrect assignments will then propagate and cause the matching to fail for the rest of the objects as well, which results in incorrect gradients. With sparse perturbations we do not need to deal with such issues, as only finding one slot that minimizes the error norm suffices, whether it is a duplicate slot or not.
- With dense perturbations, it is quite likely that the scene drastically changes from t to $t+1$, therefore, it would be increasingly difficult to obtain the same order of slots at t,$t+1$ given the same initialization, hence, solving the constrained linear programs for finding the optimal matching becomes inevitable, which is not ideal due its imposed computational burden. On the other hand, fully sparse perturbations reduce such heavy computation to a fast and simple $\arg\min$ operation over an array of size $|S|$ on average.

Having a synthetic dataset where all object properties as well as the perturbations can be carefully tuned has been particularly helpful in the early stages of this study when identifying non-identifiability sources.

Table B.5. Disentanglement scores under *known mechanisms* and *fully dense perturbations* when the target properties are p_x,p_y and objects are identical, i.e., all have the same colour, shape, size, and rotation angle.

	LD			MCC		
Model	$n=2$	$n=3$	$n=4$	$n=2$	$n=3$	$n=4$
Ours	**0.99**	**0.94**	**0.92**	**0.99**	**0.97**	**0.96**
SA-RP	0.19	0.07	0.06	0.28	0.15	0.12
SA-PC	0.40	0.20	0.13	0.36	0.25	0.17
CNN	0.31	0.15	0.07	0.44	0.32	0.24
SA-LR	0.43	0.25	0.16	0.50	0.36	0.23

Table B.6. Disentanglement scores under *known mechanisms* and *fully dense perturbations* when the target properties are p_x,p_y and objects are not identical, i.e., have different colors (but have the same shape, size, and rotation angle.).

	LD			MCC		
Model	$n=2$	$n=3$	$n=4$	$n=2$	$n=3$	$n=4$
Ours	**0.96**	**0.90**	**0.87**	**0.98**	**0.94**	**0.93**
SA-RP	0.14	0.07	0.05	0.16	0.15	0.08
SA-PC	0.16	0.18	0.10	0.18	0.30	0.18
CNN†	0.99	0.97	0.98	0.91	0.95	0.97
CNN	0.25	0.12	0.07	0.39	0.27	0.21
SA-LR	0.44	0.24	0.17	0.47	0.31	0.24

B.7. Implementation and Experimental Details

B.7.1. SA-MESH Architecture

For the slot attention architecture, we closely follow Locatello et al. (2020b), in particular, we use the same CNN encoder and decoder as they use for CLEVR, except for the initial resolution of the spatial broadcast decoder with 3D shapes where we use 4×4 since we are dealing with 64×64 images. We use a slot size of 64 and always use $n+1$ number of slots, where n is the number of objects in the scene. We use 3 iterations for the recurrent updates in SA-MESH. For details concerning SA-MESH we follow Zhang et al. (2022b). Additionally, we also truncate the backpropagation through slot updates as suggested by Chang et al. (2022) to improve training stability.

B.7.2. Disentanglement Heads

SA-MESH outputs $n+1$ slots that are of size 64, yet we need to project each of these slots to a $d-$dimensional space so we can leverage the disentanglement method from Ahuja et al. (2022b). We can simply achieve this projection by a single MLP, however, we decided to allocate more parameters for this projection and use d separate projection heads mapping 64-dimensional vectors to d separate scalars. This way identification of different properties will not affect one another due to model capacity constraints. We stack the layers shown in table B.7 to obtain a projection head per each property. The same set of d projections will be shared among all slots.

Table B.7. Layers in a projection head for disentanglement.

Layer	Input Size	Output Size	Bias	Activation
Linear (1)	64	32	True	ReLU
Linear (2)	32	32	True	ReLU
Linear (3)	32	16	False	ReLU
Linear (4)	16	1	False	ReLU

B.7.3. ConvNet Baseline

As a baseline for injective scenarios, we use a ResNet18 with an output width of 128 that is passed through `LeakyReLU` activation, which is then followed by d linear projection heads (for the same reason we use separate disentanglement heads) that map the 128-dimensional output of the CNN encoder to d separate 1-dimensional scalars that should correspond to the target $d-$dimensional space.

B.7.4. Training

For each n and for each set of disentanglement target properties, we first train SA-MESH for 2000 epochs with a batch size of 64 for 2D shapes (as the images are 128×128), and 128 for 3D shapes (since images are 64×64) on a single A100 GPU with 40GB of memory. We used a fixed schedule for the learning rate at 2×10^{-4}, and we used AdamW with a weight decay of 0.01 along with $\epsilon = 10^{-8}, \beta_1 = 0.9, \beta_2 = 0.999$. SA-MESH was firstly solely trained by minimizing for reconstruction error on the training set, then its disentanglement performance was reported on the test set for projection-based baselines (RP, PC, LR). Due to the high number of combinations of target disentanglement properties and n, we just trained SA-MESH for each configuration only once.
Unsupervised disentanglement with our method has an additional stage which takes the aforementioned pre-trained SA-MESH models and *jointly* minimizes the reconstruction and the latent loss. Note that at this stage, the SA-MESH model is *not* frozen, so the gradients flow through its network as well and help adjust the slot representations with the signal from the latent loss. Under *known perturbations*, we use the actual perturbations from the DGP to guide the model by optimizing the total loss, however, under *unknown perturbations* setting, we replace all perturbation by a hyperparameter C (see section 3.5).
CNN baselines were trained similar to SA-MESH but for much shorter, i.e., 200 epochs, and usually converge very fast in less than 50 epochs.

B.7.5. Hyperparameter Optimization

We started around the hyperparameters used by Locatello et al. (2020b) and Zhang et al. (2022b) where applicable, and tuned on small subsets of the 2D shapes training data based on linear and permutation disentanglement metrics. We considered 5 values for the learning rate [2×10^{-3},2×10^{-4},10^{-4},6×10^{-5},2×10^{-5}]. Larger batch sizes were always better and we were only constrained by memory in the case of 128×128 images of the 2D shapes dataset. We considered 2 values [0.1,0.5] for $|C|$, the fix value representing all unknown perturbations, and found $|C| = 0.1$ to perform better. We also considered slot sizes $[64, 128]$ on a small subset of the 2D shapes training dataset. Lastly we considered 9 combinations for the relative importance of latent loss and reconstruction loss when training the disentanglement heads, i.e., we considered all combinations of $w_{\text{latent}} \in \{1, 10, 100\}$, $w_{\text{recons}} \in \{1, 10, 100\}$, and found the combination of $w_{\text{recons}} = 100, w_{\text{latent}} = 10$ to strike the optimal balance between maintaining good reconstructions and allowing the slot representations to give rise to disentangled projections.

B.7.6. Datasets

2D Shapes. We use `pygame` engine (Shinners, 2011) for generating multi-object 2D scenes. Object properties in both datasets include p_x,p_y, colour, shape, size, and rotation angle. In the 2D dataset:

- p_x,p_y are generated randomly and uniformly in the [0,1] range, i.e., the boundaries of the scene, such that no two objects overlap and no object falls even partially outside the boundaries. Positional coordinates can be perturbed by ± 0.2.
- For colour, we use HSV colour representations and fix saturation (S) and value (V) at 0.6 and choose hue (H) from a set of values predefined before training (for instance $[0.0, 0.25, 0.5, 0.75]$). We adopted this 1-d representation to be consistent and have each property be represented by a scalar. Additionally we wanted to test the model's capacity when dealing with mixed discrete (colour, shape) and continuous (p_x,p_y, size, rotation angle) properties because the theory does not prevent us from doing so. Also, training Slot Attention or SA-MESH with discrete colours is computationally advantageous since the model will not have to deal with reconstructing all colours. However it should be noted that HSV is a cylindrical geometry with colour hues being the angular dimension which results in values that have a distance of 1.0 being exactly the same colour (given a fixed saturation and value). That is why a list of

colour hues such as $[0.0, 0.33, 0.66, 1.00]$ would not work since 0.0 and 1.0 are the same colour, yet our model interprets the difference as a perturbation with the amount of 1.0, which is clearly wrong. A change of colour from colour i to j where i,j index the list of colour hues H would be provided to the model as a perturbation in the amount of $(H[j] - H[i])/|H|$, where $|H|$ denotes the number of colours in H.

- Shape is also clearly discrete and is selected at random uniformly from the following set of shapes $S = \{\text{circle}, \text{square}, \text{triangle}, \text{heart}, \text{diamond}\}$. Note however, that the effects of perturbations need to be visible in the pixel space, and we should be wary of the disentanglement target properties, and for instance if the rotation angle ϕ_t is a property we aim to disentangle with perturbations, then we should exclude *circle* from the set of possible shapes as it does not reflect in the pixel space the angle perturbations. A shape transformation from shape i to j where i,j index $\mathcal{S}$, would be provided to the model as a perturbation with the amount of $(j - i)/|\mathcal{S}|$.
- Size is a continuous property in the range $[0.12, 0.24]$ of the height or width of the image which is 1. It can be perturbed by ± 0.02.
- Rotation angle is also another continuous property in $[0, \pi/4]$. Similar to colour hues, since this property is also angular, we have limited the range not to encounter situations that appear the same in the pixel space but have very different rotation angles (a square that is rotated $\pi/2$ clockwise seems unaltered, or $\pi/4$ and $3\pi/4$ rotations both look the same for a square.). Angular perturbations are ± 0.2.

We generate samples in pairs corresponding to t,$t+1$. For fully dense perturbations, we generate n vectors of dimension d, where n is the number of objects. We repeat the generation until the conditions of non-overlapping objects and non-identifiability are met, i.e., no two objects at either t or $t+1$ should overlap (before and after the perturbations), no object should fall in whole or partially out of the scene, and no two objects should be perturbed by d-dimensional offsets that are closer than some ϵ. The last condition is necessary for fully dense perturbations as otherwise the matching has no way of distinguishing which perturbation to assign to which object since the matching solely relies on the *difference* between t,$t+1$. For fully sparse perturbations, we are not constrained by the latter, and we only need to choose perturbations that do not push the chosen object out of boundaries, or make it overlap with another object. For any experiment we can have a subset of $\{p_x, p_y, \text{colour}, \text{shape}, \text{size}, \text{rotation angle}\}$ as the properties we wish to disentangle by observing perturbations in the pixel space, and we call them *disentanglement target properties*. In the generation process, any non-target property will be fixed for all objects in the whole dataset to avoid introducing unwanted variance to the disentanglement of target properties, i.e., if we choose $\{p_x, p_y, \text{colour}, \text{shape}, \text{rotation angle}\}$ as target properties, then all the objects in all samples would have the same fixed size. Lastly, we can choose to make a DGP injective or not. If we choose to make a DGP injective, we would index the objects and choose a property to be set for objects according to the indices, i.e., we can choose to make the DGP injective by colour; Suppose $n = 4$ and the list of our colour hues is $[0.0, 0.25, 0.5, 0.75]$. We would colour the objects, which are now ordered according to some index set $\mathcal{I}$, according to $\mathcal{I}$. The rest is as before, non-target properties (excluding the injectivity imposing property) will be kept fixed for the whole dataset, and target properties are generated according to fully dense or fully sparse perturbation schemes. The perturbations to all properties are *signed*, and this is especially crucial for discrete properties such as shape. The reason is that disentanglement is achieved through observing *relative* distances in the pixel space, and having only positive or only negative perturbations deprives the model of having a reference for each property.

For training we generate 1000 pair per target property such that the model on average sees at least 500 samples for either positive or negative perturbations to each property, i.e., if we choose $\{p_x, p_y, \text{colour}\}$ as target properties, we will generate 3000 samples for training. The validation and test sets always have 1000 samples. For the 2D dataset, we generate 128×128 images for better visual quality that is not distorted due to artifacts caused by perturbations. We then normalize and clip the image features (RGB values) to be in $[-1,1]$ range.

3D Shapes. For generating the 3D datasets we leverage `kubric` library (Greff et al., 2022) to obtain realistic scenes which we can highly customize. Objects sit on a floor, a perspective camera is situated at $(2.5, 0, 3.0)$ and looks at $(0.0, 0.0, 0.0)$. Directional light illuminates the scene from $(1.0, 0.0, 1.0)$ towards the center. The set of possible target properties are similar to 2D shapes, and the range of properties in which each object is spawned is as follows:

- p_x, p_y are generated randomly and uniformly in the $[-1.5,1.5]$ and $[-1.0,1.0]$ ranges respectively, such that no two objects overlap and no object falls even partially outside the boundaries. Note however, by overlap we mean that objects are spawned such that they mutually fill a volume in the 3D space, and we only prevent such occurrences, but we *do* allow *occlusions* from the perspective of the camera, which adds to the complexity of this synthetic dataset. pos_z is never a disentanglement target property and is always set such that objects sit on the floor (except when rotated). The reason for fixing the z coordinate is that any possible perturbation to the 3D coordinates is always going to be interpreted on a 2D scene that is observed by a camera that is placed somewhere above the floor. Therefore, introducing a third coordinate in the DGP and target properties has no point. Positional coordinates can be perturbed by ± 0.3.

- Colour is similarly parameterized by a scalar in HSV format as in 2D.
- Shape can be any of $\{\text{sphere}, \text{cube}, \text{cylinder}, \text{cone}\}$. Again, since the effects of perturbations need to be visible in the pixel space, and we will not use spheres if the rotation angle ϕ_t is a property we aim to disentangle with perturbations, as sphere rotations do not reflect in the pixel space. For rotation in the 3D space we choose the z-axis as the axis of rotation so that angular perturbations are maximally visible (w.r.t. the perspective camera's location).
- Size is a continuous property in the range $[0.3, 0.7]$ and can be perturbed by ± 0.15.
- Rotation angle follows the convention of 2D shapes DGP, except that the rotations are around the z-axis for better visual quality.

Since images are already generated by high fidelity using the `kubric` library, we use 64×64 images to lower the computational burden of SA-MESH autoencoder. The number of samples is always fixed at 20,000 regardless of the target properties since the 3D dataset is more complex. We use a similar transformation as in 2D, and normalize and clip the image features (RGB values) to be in $[-1,1]$ range.

Appendix C

Supplementary Materials for Reusable Slotwise Mechanisms

	Gold	RSM	RSM$_{!2}$	RSM$_{!3}$	RSM$_{\bar{k}}$	RSM$_p$
LPIPS$_{\times 100}\downarrow$	-	**7.88**	15.32	9.36	13.72	8.43

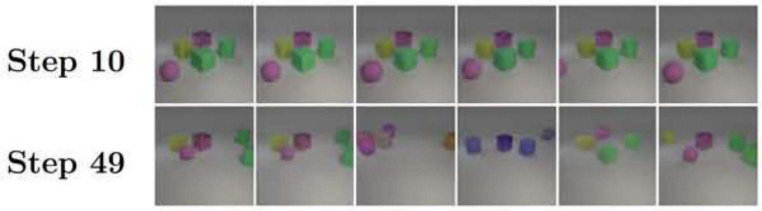


Fig. C.1. Ablation studies in OBJ3D. The original design of RSM always demonstrates the dominant performance and the accurately predicted future frames compared to the modified versions, including breaking the usages of the CCI in step 2 ($\mathbf{RSM_{!2}}$) and step 3 ($\mathbf{RSM_{!3}}$), randomly selecting mechanisms ($\mathbf{RSM_{\bar{k}}}$), and parallelly slots updating ($\mathbf{RSM_p}$). Best view in video format.

C.1. Reproducibility

Each experiment is trained on 4 A100-GPUs with 12 CPUs, using a distributed data-parallel training strategy. The number of parameters and the average training time over 5 runs are summarized in Tab. C.1. In addition, Tab. C.2 provides the necessary configurations to reproduce our work, including the setting related to datasets and the training process that follows the prior work Wu et al. (2023), and RSM's design that achieves the best tuning results.

Regarding the rollout frames K and the video length V in Tab. C.2, we predict and consider K future frames from the last burn-in steps for training, whereas, we produce the total of V frames in the inference time, including T burn-in and $V - T$ rollout steps. In other words, V equals T plus the actual rollout frames in the inference time. Regarding the training process, we employ the Adam optimizer with an initial learning rate of 2×10^{-4} and employ the decay cosine schedule to 0. We further discuss the RSM design in Appendix C.3.

C.2. Dataset Collection

We follow the collection and pre-processing of the dataset, including video length and the dataset splits, as done by Wu et al. (2023). In addition, we provide the general datasets visualization in Fig. C.2 and the visualization of *templates* in PHYRE in Fig. C.3.

OBJ3D We collect the OBJ3D dataset from the official GitHub repository[1].

[1]https://github.com/zhixuan-lin/G-SWM##datasets

Algorithm 1 Reusable Slotwise Mechanisms

N: number of slots

M: number of mechanisms

T: number of burn-in frames

K: number of rollout steps

d_s: slot dimension

d_{cci}: central contextual information dimension

Input: $s^{1:N}_{\tau^*:t} = \{s^1_{t-\tau+1}, s^2_{t-\tau+1}, \dots, s^N_{t-\tau+1}, \dots, s^1_t, s^2_t, \dots, s^N_t\} \in \mathbb{R}^{(\tau \times N) \times d_s}$ with $\tau^* = t - \tau + 1$: unrolled N slots of the previous τ steps to time t

Output: $s^{1:N}_{T+1:T+K}$: predicted N slots in the next K steps from time $T+1$ to $T+K$

Variables in RSM

- $\mathsf{cci} \in \mathbb{R}^{d_{cci}}$: the Central Contextual Information (CCI).
- $s^n_t \in \mathbb{R}^{d_s}$: the slot of interest, which is slot n^{th} at time t.
- $p \in \mathbb{R}^M$: the Gumbel distribution over M choices of selecting mechanisms.
- $\Delta s^n_t \in \mathbb{R}^{d_s}$: changes of the slot n^{th} from time t to $t+1$.

Components in RSM:

- $\mathbf{W_q}$, $\mathbf{W_k}$, $\mathbf{W_v} : \mathbb{R}^{d_s} \to \mathbb{R}^{d_s}$ denote query, key, and value projection layers transforming the unrolled $s^{1:N}_{\tau^*:t}$ in the attention mechanism.
- $\mathbf{MultiheadAttention}(\cdot) : \mathbb{R}^{((\tau+1) \times N) \times d_s} \to \mathbb{R}^{d_s}$: apply self-attention on the $s^{1:N}_{\tau^*:t}$
- $\phi(\cdot) : \mathbb{R}^{d_s} \to \mathbb{R}^{d_{\mathsf{cci}}}$ computes the central contextual information by passing the outputs of the $\mathbf{MultiheadAttention}(\cdot)$ through a nonlinear transformation (MLP).
- $\psi(\cdot) : \mathbb{R}^{d_{\mathsf{cci}}+d_s} \to \mathbb{R}^M$ computes the unnormalized probability of selecting a mechanism from M possible choices by taking the CCI and slot of interest as input and feeding that to an MLP.
- Set of M mechanisms $g_j(\cdot) : \mathbb{R}^{d_{cci}+d_s} \to \mathbb{R}^{d_s}, j \in \{1 \dots M\}$: predict the changes of each slot based on the CCI and current state of the slot. These are also realized with MLPs.

```
for each $t$ in $[T \dots t+K)$ do
    Step 0: Prepare slots buffer
    $s^{1:N}_{\tau^*:t} = \text{concat}(s^{1:N}_{\tau^*:t}, s^{1:N}_t) \in \mathbb{R}^{((\tau+1) \times N) \times d_s}$

    for each $s^n_t$ in $s^{1:N}_t$ with $n \in 1 \dots N$ do
        Step 1: Compute the central context
        $\mathsf{cci} = \phi(\mathbf{MultiheadAttention}(\mathbf{W_q}(s^{1:N}_{\tau^*:t+1}), \mathbf{W_k}(s^{1:N}_{\tau^*:t+1}), \mathbf{W_v}(s^{1:N}_{\tau^*:t+1})))$

        Step 2: Select a mechanism for slot $s^n_t$
        $p = \mathsf{Gumbel\text{-}max}(\psi(\text{concat}(\mathsf{cci}, s^n_t))$

        Step 3: Apply the selected mechanism to slot $s^n_t$. Note that $p$ is one-hot-like distribution.
        $\Delta s^{n,j}_t = g_j(\text{concat}(\mathsf{cci}, s^n_t)) * p^j \quad \forall j \in \{1, \dots, M\}$
        $\Delta s^n_t = \sum_{j=1}^{M} \Delta s^{n,j}_t$

        Step 4: Update the slots buffer with the new value of $s^n_{t+1}$
        $s^n_{t+1} = s^n_t + \Delta s^n_t$
    end for
end for
return $s^{1:N}_{T+1:T+K}$
```

Table C.1. Summary of the number of parameters and training duration. "M" stands for *millions*. "h" stands for GPU hours.

	RSM	NPS	SwitchFormer	SlotFormer
OBJ3D				
Num. Params	0.76M	0.99M	0.82M	0.82M
Training Duration	21h	25h	20h	21h
CLEVRER				
Num. Params	3.1M	4.06M	3.22M	3.22M
Training Duration	82h	94h	72h	86h
PHYRE				
Num. Params	5.13M	5.98M	6.38M	6.38M
Training Duration	29h	33h	28h	30h
Physion				
Num. Params	5.61M	6.7M	6.41M	6.41M
Training Duration	32h	41h	30h	34h

CLEVRER In this work, we directly download the CLEVRER dataset from the official website[2].

PHYRE In this work, we explore the PHYRE-1B version that defines the amount of the red ball as 1. The PHYRE dataset is generated by the instructions provided from the official GitHub page[3].

Physion We directly download the Physion dataset from the official GitHub page[4]

C.3. Implementation Details

C.3.1. Loss Functions

The following is the training objective that follows the prior work (Wu et al., 2023).

[2]clevrer.csail.mit.edu/

[3]github.com/facebookresearch/phyre

[4]github.com/cogtoolslab/physics-benchmarking-neurips2021#downloading-the-physion-dataset

Table C.2. Summary of experiments' configuration, including the configuration of datasets, training process, and RSM.

	OBJ3D	**CLEVRER**	**Physion**	**PHYRE**
Frame Size	64×64	64×64	128×128	128×128
Num. Slots N	6	7	6	8
Slot Size d_s	128	128	192	128
Burn-in Frames T	6	15	15	1
Temporal Window τ	6	15	15	6
Rollout Steps K	10	10	10	10
Video Length V	6+44	15+42	15+35	1+14
Batch Size	128	128	128	64
Num. Epochs	200	80	25	50
Object-centric Model	SAVi	SAVi	STEVE	SAVi
Loss Weight λ	1.0	1.0	0.0	0.0
Num. Mechanisms M	7	7	5	5
Num. Layers of $\psi(\cdot)$	1	2	2	2
Num. Layers of Mechanism	1	3	3	3

We employ slot reconstruction loss to train the rollout future frames prediction, as described in Eq. C.3.1 with n is slot index, s^n_{T+k} is the predicted rollout slot, and s^{*n}_{T+k} is the pre-trained slot (that is used as the target slot).

$$\mathcal{L}_S = \frac{1}{K \cdot N} \sum_{k=1}^{K} \sum_{n=1}^{N} \| s^n_{T+k} - s^{*n}_{T+k} \|^2 \tag{C.3.1}$$

Experiments using SAVi as the object-centric model also use the image reconstruction loss, as described in Eq. C.3.2 with f_{dec} as frozen decoder and x_{T+k} is the ground truth image. Experiments using STEVE as the object-centric model can still employ the image reconstruction loss; however, we do not conduct such experiments with image reconstruction loss due to the dramatically extended training time. In PHYRE, we do not utilize the image reconstruction loss $\mathcal{L}_I$ due to the large image size that could affect the training time and the

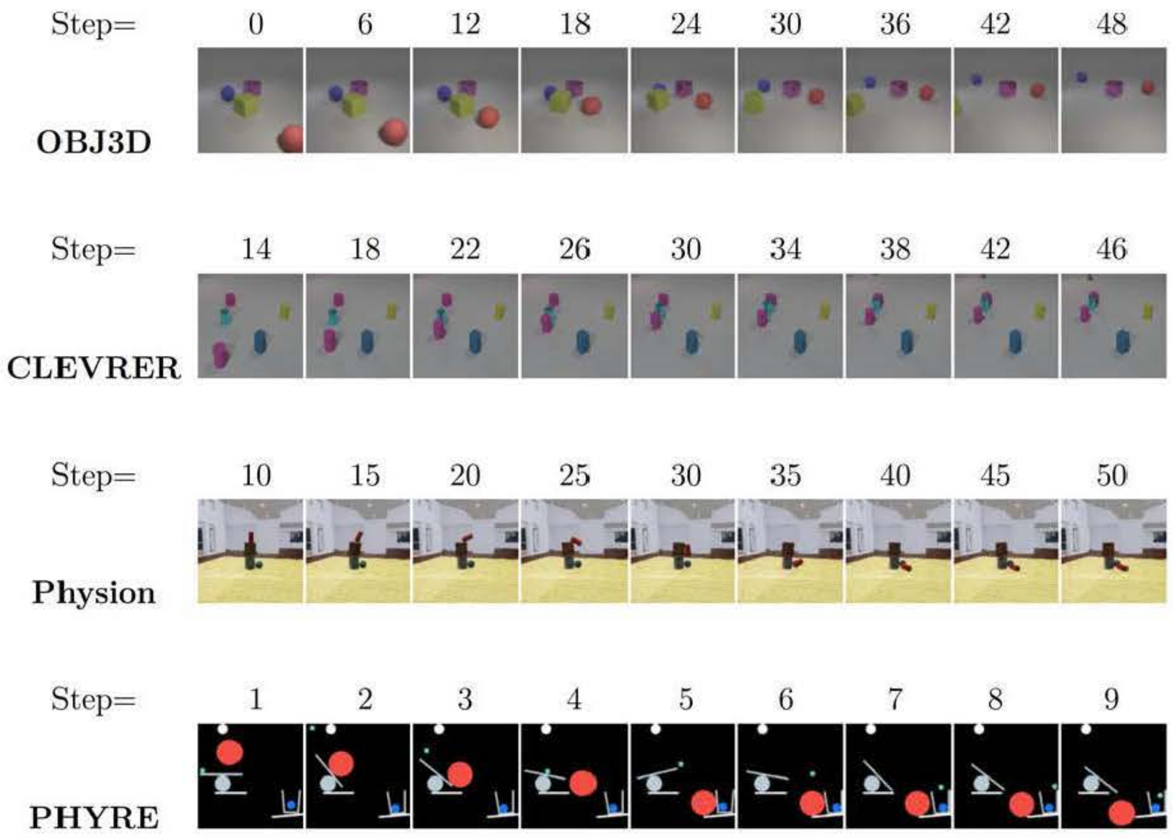

Step= 0 6 12 18 24 30 36 42 48
OBJ3D
Step= 14 18 22 26 30 34 38 42 46
CLEVRER
Step= 10 15 20 25 30 35 40 45 50
Physion
Step= 1 2 3 4 5 6 7 8 9
PHYRE

Fig. C.2. Dataset visualization over steps.

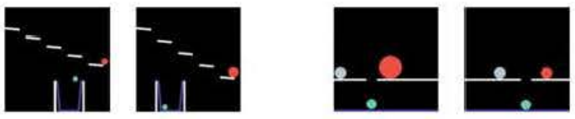


Fig. C.3. Pairs of PHYRE scenes in the same template with similar objects in the background and differences in objects' positions.

simplicity of PHYRE's object compared to other environments in this work.

$$\mathcal{L}_I = \frac{1}{K}\sum_{k=1}^{K} \|f_{dec}(s_{T+k}) - x_{T+k}\|^2 \tag{C.3.2}$$

The overall objective function is the weighted sum of the above losses, as presented in Eq. C.3.3.

$$\mathcal{L} = \mathcal{L}_S + \lambda\mathcal{L}_I \tag{C.3.3}$$

Note that when using slot and image reconstruction losses as presented in Eq. C.3.1 and Eq. C.3.2, the model will fail in the end-to-end training since the overall objective function is constrained to pre-trained slots ($s*^{n}_{T+k}$) and well-trained decoder ($f_{dec}(\cdot)$). Motivated by this observation, in Appendix C.5, we provide the end-to-end training objective to compare RSM and other approaches' abilities in objects' dynamics modeling from scratch.

C.3.2. Model Architecture

The distinction among methods is in the process of predicting the next state based on the provided input of past states. In this section, we provide a detailed description of how both the baselines and the proposed RSM approach the task of the next state prediction. Specifically, we examine two key aspects: (1) how slots communicate with each other and (2) the precise value to be predicted, as revealed through the design specifics of each method.

C.3.2.1. Baselines. SlotFormer (Wu et al., 2023) SlotFormer consists of 3 main parts: (1) the Multi-Layer Perceptron (MLP) input projection layer, (2) a Transformer architecture layer, and (3) the MLP output projection layer. First, the unrolled past states (the sequence of $\tau \times N$ slots) are passed through the input projection layer before being processed by the Transformer model. Afterward, the output projection produces the next state from the Transformer's output. Through this process, slots densely communicate with each other in all 3 parts of SlotFormer. In addition, the entire next state of slots is directly generated by the models. In this work, we utilize the public implementation[5] of SlotFormer.

SwitchFormer from Switch Transformer (Fedus et al., 2021) The Switch Transformer is a variant of the Transformer architecture designed to improve efficiency and scalability. It achieves this through the usage of dynamic routing and adaptive computation. This architecture employs a "switch" module that intelligently routes tokens to different layers based on their content.
In this work, we create SwithFormer that integrates the Switch Transformer implementation[6] into the SlotFormer codebase and replace the vanilla Transformer by Switch Transformer. In this way, SwitchFormer follows the same strategy as SlotFormer, which conducts the dense communication among slots and directly predicts the entire next state of slots.

NPS (Goyal et al., 2021a) NPS is a framework that combines neural networks and production systems for object modeling that integrates neural networks into the production system. In traditional production systems, rules are used to represent knowledge and guide the system's behavior. In the case of NPS, they conduct a set of rules to handle the pair-wise

[5]github.com/pairlab/SlotFormer

[6]nn.labml.ai/transformers/switch/index.html

interaction of slots. The two slots involved in an interaction, which are named the *primary* and *contextual* slots, are selected through attention mechanisms. In addition, the official design of NPS for object dynamics' modeling, inspired by Kipf et al. (2020), predicts the changes of the *primary* slot within a time step instead of the entire slots. Afterward, the predicted next state of slots is the sum of the current state and the predicted slots changes. In this work, we integrate the official NPS[7] to the SlotFormer's codebase for consistent in the training pipeline and sharing the pre-trained object-centric model. NPS consists of an MLP slots encoder that requires a fixed input size; therefore, at the beginning of the rollout prediction process of PHYRE, which environment has the *actual* temporal window size increases from 1 to 6, we duplicate the burn-in frame to have a fixed 6 steps window size along the rollout process.

C.3.2.2. RSM. We design RSM as a framework for dynamics modeling with a relaxed inductive bias in the communication density of slots that enables a subset of slots involved in communication, based on a particular context through the CCI. RSM consists of 3 main elements as described in Sec. 4.2.1: (1) the multi-head self-attention that computes the CCI, (2) the $\psi(\cdot)$ that estimates the suitable mechanism, and (3) a list of mechanisms. In terms of the multi-head self-attention, We employ a 4-head architecture for multi-head self-attention, where the hidden size of the Feed-forward Networks is set to $2 \times d_s$. We design $\psi(\cdot)$ as MLP with the number of hidden layers being tuned (See Appendix C.4.3). Similarly, each individual mechanism is designed as MLP layers with a tuned number of hidden layers. All mechanisms share the same architecture but have separate weights. In addition, the total parameters from all mechanisms are constrained to be similar across different amounts of mechanisms, meaning that as the number of allocated mechanisms increases, the size of each mechanism decreases (further investigated in Appendix C.4.3). Like NPS, the mechanism predicts the changes in a slot within 2 consecutive steps instead of the entire slot. Last but not least, we (and NPS) omit the input and output projection layers of SlotFormer and SwitchFormer.

C.3.2.3. Downstream tasks. CLEVRER VQA model Inherit from baseline (Wu et al., 2023), we employ Aloe as the VQA model that concatenates the predicted rollout slots and the processed question (represented as language tokens) before passing through a stack of Aloe Transformer encoder to predict the answer.s

[7]github.com/anirudh9119/neural_production_systems

Physion VQA model In the VQA task of Physion, the objective is to determine whether the red object will come into contact with the yellow object once the dynamics of all the objects have been completed. Since the task does not involve any language processing, we construct an MLP model that takes the rollout slots as input. The MLP processes these slots and produces a binary prediction, indicating whether the red object and the yellow object or not "touched" and "did not touch" each other.

PHYRE Readout In the PHYRE action planning task, an action involves determining the size and position of the red ball. In our approach, we utilize the set of 10,000 predefined actions introduced by Bakhtin et al. (2019) and train a readout model to determine if a given action can solve the task. To construct the readout model, we draw inspiration from Wu et al. (2023) that designs a 2-layer MLP model on top of the encoded states. The readout model takes the predicted rollout states as input. To process these states, we employ an encoder, which differs depending on the specific model variant used. In SlotFormer, a Transformer is used, while in SwitchFormer, a Switch Transformer is employed. An MLP is used as the encoder in the NPS model, and a Multi-head Self-Attention mechanism is utilized in the RSM model. Once the states have been processed by the encoder, the classifier generates a binary output that indicates whether the task has been solved or not. This output serves as an inference for the task's solvability.

C.4. Further Discussion on Experiment Results

C.4.1. Future Roll-out Prediction

In Fig. 4.3, we provide the upgraded version of Fig. 4.2 with the exact the same prediction results but with significantly higher dpi (dots per inch) in plotting the predicted frames. Despite leveraging pre-trained slots and object-centric models, RSM effectively maintains visual quality by predicting the *changes* of slots instead of predicting the next state of a slot instead of the entire slot. This approach allows RSM to handle action-free scenarios well and significantly reduce error accumulation by facilitating null transitions (predict zero changes of the slot) that preserve slot integrity.

Fig. 4.2 depicts the rollout frames generated by RSM alongside the baselines. Notably, RSM excels in producing robust future frames that accurately capture the dynamics of objects while maintaining visual fidelity. Nevertheless, we have encountered a challenge in generating objects with sharpness within the CLEVRER dataset. SlotFormer marks a partially incorrect object's dynamics in this case.

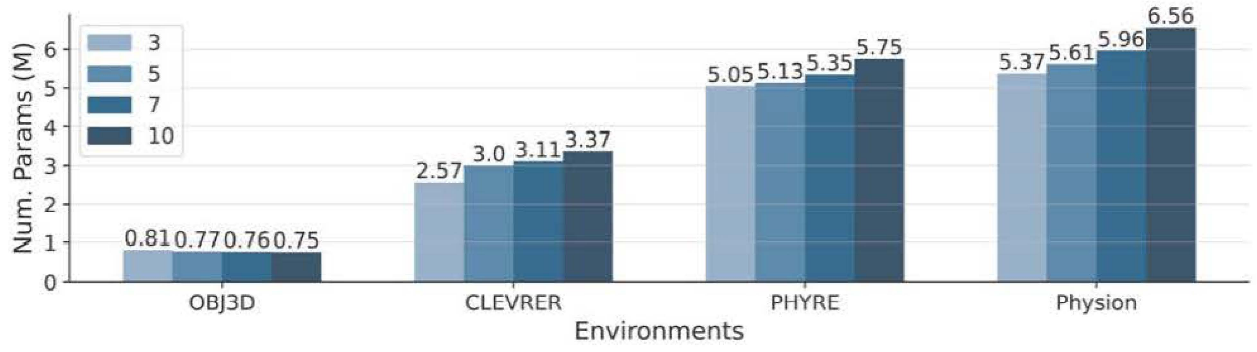


Fig. C.4. The scaling of RSM's parameters with different amounts of mechanisms.

C.4.2. Discussion on the Action Planning task in PHYRE

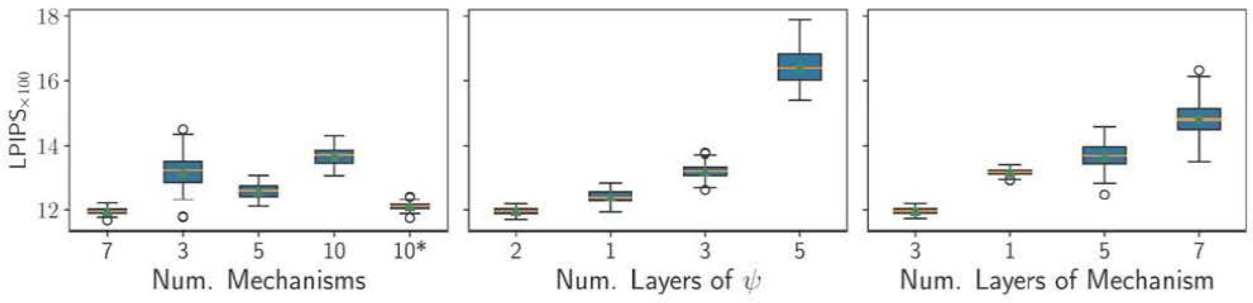
We encountered difficulties in reproducing the action planning results of SlotFormer, even when using their provided checkpoints. In Table 4.3, we report the iid result of 76.4 for SlotFormer, which is 5.6 points lower than the officially reported value. To investigate this issue, we explored potential reasons and identified the following possible factors: (1) Instability of results: The official GitHub page of SlotFormer[8] acknowledges that the results in this specific task may exhibit instability. This suggests that achieving consistent and reproducible results with SlotFormer can be challenging, and (2) Data discrepancies: The PHYRE dataset, being regenerated rather than downloaded from a common source, introduces variations in the computing configuration. These data collection and processing differences may contribute to the disparities observed between our results and the official reports.

C.4.3. Finetuning Results in RSM

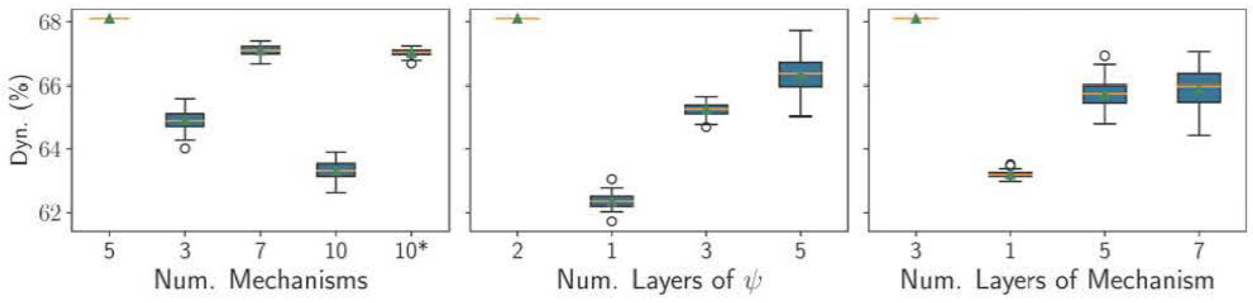
In Figure C.4.3, we present the results of hyperparameter fine-tuning on CLEVRER and Physion datasets. This section explores the impact of the number of mechanisms, the expansion of $\psi(\cdot)$ parameters, and the structure of the mechanism models. The term *number of layers* in this analysis refers to the hidden layers within the MLP structure that maps the input dimension to the output dimension in the respective models.

In terms of the number of mechanisms, we have observed that having either a large or a small value for M significantly worsens the results and leads to high variance. However, we have also discovered that the tasks can be solved with relatively few mechanisms, even when dealing with diverse object movements. To validate the design principle stated in

[8]github.com/pairlab/SlotFormer



(a) CLEVRER. The lower score indicates better performance.



(b) Physion. The higher score indicates better performance.

Fig. C.5. The finetuning results in CLEVRER and Physion. The first box of each subplot is the configuration that achieves the best performance, whereas other boxes are performance that has a different number of mechanisms, layers of the ψ model, or layers of each mechanism from the best setting. The plotted values are the mean and standard deviation over 5 different runs corresponding to each configuration. See text for more details.

Appendix C.3.2.2, which proposes an inverse relationship between the number of the mechanism's parameters and the number of mechanisms, we conducted experiments with named *10**, which replicates mechanisms to achieve a total of 10 mechanisms without reducing the number of parameters, as compared to the best configuration (the first boxes). Our findings indicate that the replicated configuration (10*) achieves a slightly lower score than the best configuration, and the additional mechanisms are not selected by the $\psi(\cdot)$ models.

In terms of the number of layers in $\psi(\cdot)$, the challenge is to map a $2 \times d_s$ vector to a compact vector of size M. Our findings suggest that using 1 or 2 hidden layers yields favorable results in terms of achieving a high score. However, we find a complication in identifying consistent patterns for fine-tuning $\psi(\cdot)$ across different datasets.

Lastly, when considering the mechanism structure, we have noticed that increasing the number of parameters allows a single mechanism to achieve a moderately decent score, although not a high score. Consequently, the $\psi(\cdot)$ model has a tendency to choose only one mechanism for all cases, resulting in a lower score when assigning 5 or 7 layers to the mechanism.

C.4.4. Ablation Studies

Fig. C.1 provides the visualization that supports the ablation studies in Sec. 4.4.5 in the main text.

C.5. Experiments on End-to-end Training Pipeline

We propose an additional experiment to verify the methods' ability to model objects' dynamics from scratch, additionally, in action-conditioned environments. We establish an end-to-end training pipeline that comprehensively assesses the models' effectiveness in the entire process of extracting slots from frames, handling the objects' action-conditioned dynamics, and finally decoding slots back into frames.

RSM generally demonstrates the dominant ability to model objects' dynamics and produce meaningful slots compared to the baselines.

C.5.1. Experiments Setup

Environment: This dataset consists of objects arranged in a 5×5 grid. At each time step, a single frame and an action are provided. The action specifies one object and one manipulation from the set of UP, RIGHT, DOWN, and LEFT. The challenge of this task is to determine the feasibility of the given action and predict the resulting frame. For example, an action of moving an object to the LEFT is executable if no objects are obstructing the left side of the target object. Otherwise, the frame remains unchanged.

Encoder and Decoder architecture In this experiment, we follow the encoder proposed by Kipf et al. (2020) that contains a simple CNN-base *Object Extractor* to extract input frame to N feature maps and MLP-base *Object Encoder* to encode feature maps to vector space, *i.e.* the object slots. Afterward, we propose a Decoder architecture for slot-based visual prediction, consisting of N Slot Decoders with separate weights. Each slot is decoded into an RGB reconstruction, and the final frame reconstruction is obtained by summing the reconstructions of all slots.

Training Objective This experiment follows the Contrastive loss setup as Kipf et al. (2020) that uses the prediction result of the transition model to form the positive hypothesis, whereas, sampling random input states in the same batch to form the opposing hypothesis. The target slots, $s^{*1:N}_{t+1}$, are obtained by passing the target next frame through the Encoder, whereas, $s^{1:N}_{t+1}$ represent the predicted slots, and $\tilde{s}^{1:N}_{t}$ indicates the random slots in the training batch.

$$\begin{gathered} H = \mathsf{MSE}(s^{1:N}_{t+1}, s^{*1:N}_{t+1}), \quad \tilde{H} = \mathsf{MSE}(\tilde{s}^{1:N}_{t}, s^{*1:N}_{t+1}) \\ \textit{Contrastive Loss}: \mathcal{L} = H + \max(0, 1 - \tilde{H}) \end{gathered} \tag{C.5.1}$$

We consider the sum of two BCE loss terms in training Decoder, $\mathcal{L}_1$ and $\mathcal{L}_2$. $\mathcal{L}_1$ is applied on $x^{'}_{t}$, obtained by passing $s^{1:N}_{t}$ through Decoder, which is expected to close to the input frame x_t. $\mathcal{L}_2$ is applied on $x^{'}_{t+1}$, obtained by passing $s^{1:N}_{t+1}$ through Decoder to achieve the prediction of next frames reconstruction, which is desired to close to the target next frame x_{t+1}.

$$\begin{gathered} x'_t = \mathsf{Decoder}(s^{1:N}_{t}), \quad x'^{1:N}_{t+1} = \mathsf{Decoder}(s'^{1:N}_{t+1}) \\ \mathcal{L}_1 = \mathsf{BCE}(x'_t, x_t), \quad \mathcal{L}_2 = \mathsf{BCE}(x'_{t+1}, x_{t+1}) \\ \textit{BCE Loss}: \mathcal{L}_{Decoder} = \mathcal{L}_1 + \mathcal{L}_2 \end{gathered} \tag{C.5.2}$$

C.5.2. Experimental and Analytical Results

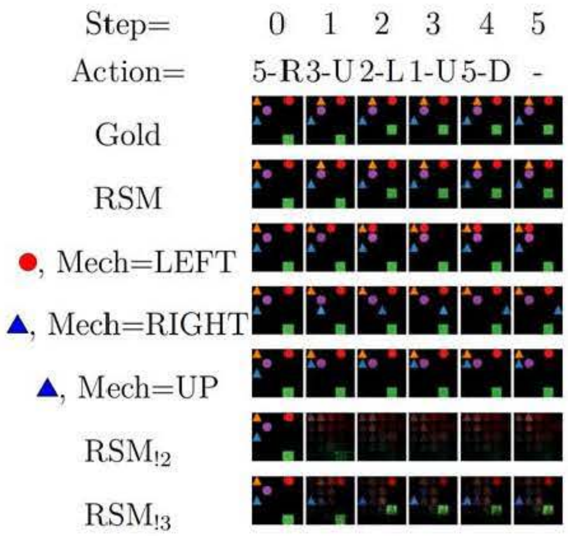


Fig. C.6. Observation of mechanisms assignment and performance in 5 rollout steps in 2D Shapes. In the last three rows, we present the reconstruction by only changing 1 slot with 1 mechanism applied to that slot over 5 steps, while all other slots are untouched.

Disentangling objects' transition to mechanisms In Figure C.6, we study the role of each mechanism in RSM. The analysis shows that RSM produces a reasonable reconstruction compared to the ground-truth frame and encourages the mechanisms to distinguish

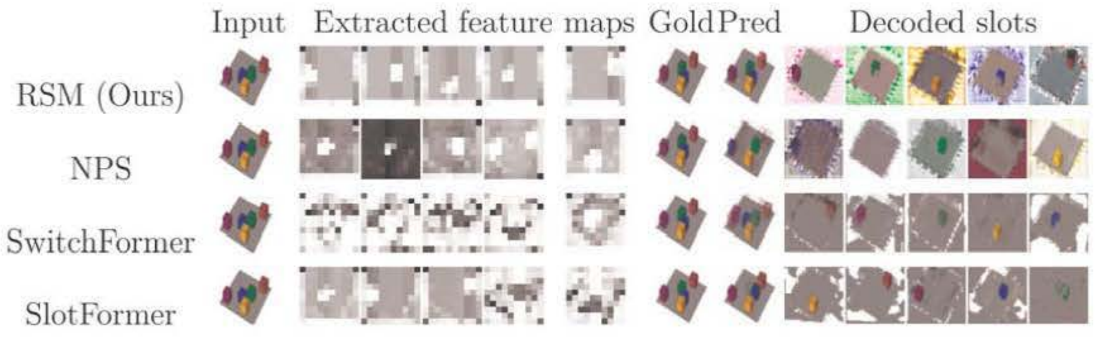


Fig. C.7. Comparison of extracted feature maps from a scene and reconstructions in 3D Cubes. RSM deals with object slots better than baseline in both slots extracting and slots decoding phases.

themselves in their roles. In this sample, we can infer the 5 slots corresponding to: **1**: red round, **2**: blue triangle, **3**: green square, **4**: purple round, and **5**: yellow triangle. Likewise, we observe from the mechanism assignment in each step and list the role of each mechanism (5 mechanisms in this experiment) specialize as in the following actions: Mechanism **1**: RIGHT, Mechanism **2**: UP, Mechanism **3**: LEFT, Mechanism **4**: *DO NOT MOVE*, and Mechanism **5**: DOWN.

Looking deeper into the reconstruction result, RSM takes advantage of the CCI to assign a suitable mechanism for each slot in all scenarios, considers the particular situation, and reacts differently to the same action. For instance, we observe that with the same action UP given in step 1 on the green rectangle and step 3 on the red round, RSM recognizes the situations that the object is allowed to move and blocked by the upper wall, respectively, then applies the movement at step 1 while not modifying objects at step 3 and generating the correct reconstruction in both cases. We can see a similar example in the row of mechanism 2 at step $3 \rightarrow 4$ when the green object does not move UP and remains at the same position since the red object blocks it.

The ability to decompose frame into slots We analyze RSM's slot-centric representation ability. In Figure C.7, we illustrate a comparison of the extracted feature maps with a size of 10×10, which are constructed by the Encoder model and the reconstructed slots and frame with a size of $3 \times 50 \times 50$, acquired by the SlotDecoder models that receive the input as the predicted next state. We find that RSM decomposes in the input frame into separated slots and keeps each object in the same slot until the decoding phase. In contrast, the baselines do not capture all objects but produce noised feature maps (SlotFormer and

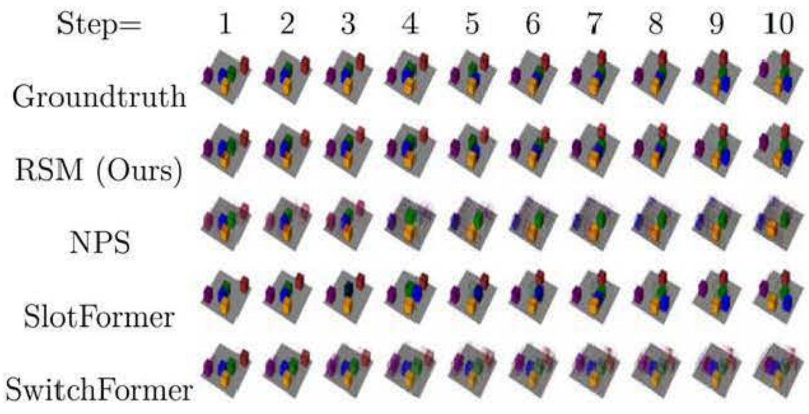


Fig. C.8. Reconstruction comparison on 3D Cubes dataset

SwitchFormer), or put the same object in two slots and identify two objects in another slot (NPS).

We observe that combining the CCI with the sequential updates that encourage slots to observe the modification of each other benefits from recognizing the overlap of objects. More specifically, RSM only generates a part of the object in case that object is covered by another one (e.g. the green and blue cubes in slot 2 and slot 4 are partially covered by the yellow cube). On the other hand, the baselines overlook the condition of executability of action and generate overlapped objects in some cases (e.g. the green, blue, and yellow cubes in SlotFotmer overlapped with each other). Lastly, SwitchFormer produces blurry objects and incorrectly predicts objects' dynamics.

Reconstruction in 3D Cubes One of the challenges to generating reconstructions in 3D Cubes is to recognize the visibility order of objects. Figure C.8 exposes RSM's strength in communication among slots to obtain the order information, as well as generate the proper movement of slots and achieve an accurate reconstruction compared to the ground truth. In contrast, SlotFotmer misses that kind of information from the beginning steps and renders the blue and green objects inside each other. Besides, other methods lose the information about some objects and produce a not completed reconstruction at the end, witnessing a huge gap from the following steps to step 10.

C.6. Limitations and Future Works

While RSM has demonstrated its robustness for modeling objects' dynamics in various tasks in both iid and OOD settings, there is room to expand this work due to the following limitations:

(1) Sensitivity to Hyperparameters: RSM requires tuning the number and size of mechanisms. Future research could explore automated methods for determining optimal values, and enhancing RSM's adaptability across tasks and scenarios.
(2) Environments in this study are all observable. Future work should explore a larger range of observable and unobservable environments for more insights.