Exploring Discrete Representations in Stochastic Computation Graphs: Challenges, Benefits, and Novel Strategies

Inauguraldissertation
zur Erlangung des akademischen Grades
eines Doktors der Naturwissenschaften
der Universität Mannheim

vorgelegt von

David Friede

Mannheim, 2023

Dekan:	Professor Dr. Claus Hertling, Universität Mannheim
Referent:	Professor Dr. Heiner Stuckenschmidt, Universität Mannheim
Korreferent:	Professor Dr. Mathias Niepert, Universität Stuttgart

Tag der mündlichen Prüfung: 23.01.2024

Acknowledgments

I want to thank my supervisor, Heiner Stuckenschmidt. He has given me the trust and freedom to evolve and make my own decisions from day one. His support helped me to see the big picture and guided me when I became lost in detail. Special thanks go to my second supervisor and mentor, Mathias Niepert. More than anyone else, Mathias got to know my potential and my weaknesses alike. He made me realize there is always a way out and never stopped pushing me to my limits. Third, I want to thank Margret Keuper for introducing me to the realm of proper research. If there is anything I have learned in the last four years, it is that I know nothing. I want to thank my parents, Iris and Peter Friede. They have given me the strength and confidence that guide me through all my life decisions. Special thanks go to my sister, Jana Friede. We have always been a strong team, feeling invincible together. I want to thank Ye Jing for all the good times and for her patience. Pursuing a Ph.D. can be a lonely endeavor; Jing and my friends, especially Yannick Mogge and James Gollmart, ensured that I always kept in touch with reality, which I am most thankful for. Additional thanks go to my friends, Dieter Langer, Christian Reimers, and Malin Lachmann. They have guided and supported me for as long as I can remember and have played an important role in shaping who I am today. I want to thank Lea Cohausz, who made our office hallway slightly less empty. Lastly, I want to thank Bernhard Schäfer and Bhushan Kotnis for all the fruitful discussions. These interactions have been the most enjoyable part of my Ph.D.

Abstract

The evolution of deep learning has led to a need for models with enhanced interpretability and generalization behaviors. As part of this, discrete representations play a significant role since they tend to be more interpretable. This thesis explores discrete representations in Stochastic Computation Graphs (SCGs), focusing on challenges, benefits, and novel strategies for their structure and parameter learning. Recent successes in model-based reinforcement learning and text-to-image generation have demonstrated the empirical advantages of discrete latent representations. However, the reasons behind their benefits remain unclear. Furthermore, training deep learning models with discrete representations presents unique problems, primarily associated with differentiating through probability distributions. In response, we establish a background as a solid foundation for our research, focusing on SCGs. We then analyze the challenges associated with training models with discrete representations and their benefits. In addition, we propose novel strategies to address these challenges, which we evaluate experimentally across various domains. On the one hand, we propose learning the structure of computation graphs for efficient Neural Architecture Search. On the other hand, we propose altering the scale parameter of Gumbel noise perturbations and implementing dropout residual connections for efficient parameter learning of discrete SCGs. Furthermore, we present a new approach of employing a categorical Variational Autoencoder to enhance disentanglement. Our extensive experimental evaluations across diverse domains demonstrate the effectiveness of the proposed methods. We find that the challenges associated with training discrete representations can be significantly mitigated, and our strategies help to improve the models' interpretability and generalization behavior. Our findings also reveal the inherent grid structure of categorical distributions as an efficient inductive prior for disentangled representations. This study provides critical insights into discrete representations in deep learning, extending our understanding and proposing novel methods that show promising results in experimental evaluations. Our work highlights promising future work for further refinement of discrete representations and their diverse applications.

Zusammenfassung

Die Weiterentwicklung im Bereich des Deep Learning hat zu einem Bedarf an Modellen mit verbesserter Interpretierbarkeit und Generalisierbarkeit geführt. Diskrete Repräsentationen spielen dabei aufgrund ihrer Interpretierbarkeit eine bedeutende Rolle. Diese Dissertation untersucht diskrete Repräsentationen in Stochastic Computation Graphs (SCGs) und konzentriert sich dabei auf Herausforderungen, Vorteile und ihre strukturellen und parametrischen Lernaspekte. Jüngste Erfolge im modellbasierten Reinforcement Learning und der Text-zu-Bild-Generierung haben die empirischen Vorteile diskreter latenter Repräsentationen aufgezeigt. Die Gründe für ihre Vorteile bleiben jedoch unklar. Zudem ergeben sich spezifische Probleme beim Training von Deep-Learning-Modellen mit diskreten Darstellungen, die hauptsächlich mit der Differenzierung durch Wahrscheinlichkeitsverteilungen verbunden sind. Als Antwort darauf etablieren wir eine theoretische Grundlage für unsere Forschung, wobei wir uns auf SCGs konzentrieren. Wir analysieren die Herausforderungen, die mit dem Training von Modellen mit diskreten Repräsentationen einhergehen, sowie deren Vorteile. Darüber hinaus schlagen wir neue Strategien zur Bewältigung dieser Herausforderungen vor, die wir in verschiedenen Domänen experimentell evaluieren. Einerseits schlagen wir vor, die Struktur von Computation Graphs für eine effiziente Neural Architecture Search zu lernen. Andererseits schlagen wir vor, den Skalenparameter von Gumbel-Verteilungen anzupassen und Dropout-Residualverbindungen für ein effizientes Parameterlernen von diskreten SCGs zu implementieren. Zusätzlich präsentieren wir einen neuen Ansatz zur Verwendung eines kategorischen Variational Autoencoders zur Verbesserung der Entflechtung von Repräsentationen. Unsere umfangreichen experimentellen Auswertungen in verschiedenen Domänen demonstrieren die Effektivität der vorgeschlagenen Methoden. Wir stellen fest, dass die mit dem Training diskreter Repräsentationen verbundenen Herausforderungen erheblich reduziert werden können und dass unsere Strategien zur Verbesserung der Interpretierbarkeit und der Generalisierungsleistung der Modelle beitragen. Zudem zeigen unsere Ergebnisse, dass die Gitterstruktur von kategorischen Verteilungen als effizienter induktiver Prior für entflochtene Repräsentationen dient. Diese Studie liefert wichtige Erkenntnisse über diskrete Repräsentationen im Bereich des Deep Learning, erweitert unser Verständnis und stellt neue Methoden vor, die in experimentellen Auswertungen vielversprechende Ergebnisse zeigen. Unsere Arbeit legt den Grundstein für vielversprechende zukünftige Forschung zur weiteren Verbesserung diskreter Darstellungen und ihrer vielfältigen Anwendungen.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Deep learning has transformed the landscape of artificial intelligence by enabling machines to learn representations directly from raw data. Representation learning transforms complex, high-dimensional data into low-dimensional and manageable forms [7]. When optimizing these representations for machine learning tasks, the decision to use continuous or discrete representations is fundamental. While the former has seen wide adoption due to their differentiability, the latter has gained interest due to their distinct benefits, such as interpretability and better generalization [12], motivating the profound exploration of discrete representations.

Discrete representations are challenging to optimize due to their non-differentiability [30, 109, 29]. However, they hold great potential for increased interpretability and effective data structuring [12]. Unlike continuous representations, which often require a close examination of their context for interpretation, discrete representations such as symbols or graphs can enhance the transparency and trustworthiness of the models. Moreover, discrete representations can capture data structures more effectively, which is beneficial in tasks like natural language processing or program synthesis [76, 15].

Additionally, there have been an increasing number of empirical successes of discrete representations across a variety of machine learning tasks [35, 75, 81, 91, 36]. Methods based on discrete latent spaces in image generation [91] and model-based reinforcement learning [36] outperform their continuous counterparts. However, a comprehensive understanding of the reasons behind these empirical successes and under which conditions they occur remains unknown [35]. This creates a knowledge gap in the current literature and highlights the need for a deeper exploration into the benefits of discrete representation learning.

Given the potential advantages of discrete representations and their demonstrated successes, it is evident that the obstacles preventing their wider adoption must be addressed. Discrete representations are often depicted as random variables that are stochastic nodes in the computation graph [41, 64]. Stochastic Computation Graphs (SCGs) were first introduced by Schulman et al. [95] as a general framework for modeling and optimizing differentiable computation graphs with such stochastic nodes. Estimators that aim to circumvent the non-differentiability of discrete random variables suffer from high variances and biasedness, complicating the training process [30, 109, 29]. This motivates the search for novel techniques and methods that can leverage the benefits of discrete representations while mitigating the challenges associated with their optimization.

The overall motivation for this thesis lies in exploring the unknown properties of discrete representations within discrete SCGs. Understanding their challenges and benefits and proposing novel strategies for their effective utilization form the core of this exploration. By bridging the existing gaps in this field, this work aims to advance our understanding of discrete representation learning and contribute to the development of more robust and interpretable models.

1.1 Discrete Representation Learning

Representation learning focuses on automatically discovering representations of raw data inputs needed for machine learning tasks [7]. The goal is to replace manual feature engineering with efficient algorithms that find important data features in an unsupervised manner. This capability is particularly critical for complex tasks such as natural language processing or computer vision, where raw data (like text or pixels) need to be transformed into higher-level features for more effective processing [19].

A key consideration in representation learning is whether the learned representations should be continuous or discrete [110]. Continuous representations, often used in deep learning, encode input data as real-valued vectors. These representations have been effective for tasks like image classification, where inputs (such as pixels) have natural continuity [37]. On the other hand, discrete representations express input data as discrete variables. A simple example is one-hot encoding, where each category in the data is represented as a binary vector. In this thesis, we concentrate on discrete representations depicted as random variables that are stochastic nodes in the computation graph [41, 64]. These computation graphs represent a discrete version of Stochastic Computation Graphs [95].

Discrete representations have gained attention in deep learning research because they offer several advantages. They are often more interpretable than their continuous counterparts. While a continuous vector is challenging to interpret beyond its immediate neighbors, discrete representations can capture the structure of the data more effectively. In natural language processing, for instance, a graph representation of a sentence can capture its syntactic structure, which a vector might not [15]. Similarly, in program synthesis, a discrete program representation naturally captures the program's control flow structure [76]. However, it is important to note that the benefits of discrete representations extend beyond tasks that inherently involve discrete structures. Empirical evidence shows that discrete representations can also benefit various machine learning tasks, including non-discrete datasets. For instance, the application of discrete representations in image generation and model-based reinforcement learning has led to substantial advances in these areas. One prominent example is the use of discrete variational autoencoders in text-to-image generation models like Dall-E [81] and Stable Diffusion [91], who leveraged discrete latent variables to achieve state-of-the-art results. Similarly, other work applied discrete representations for model-based reinforcement learning, demonstrating improved generalization behavior [35, 75, 36]. Discrete latent spaces significantly enhanced performance in these areas, indicating a potential advantage not restricted to discrete or symbolic tasks.

Nonetheless, the underlying reasons for these empirical successes are not fully understood, and the question of why discrete representations sometimes outperform their continuous counterparts is still open [35]. Despite this lack of a concrete theoretical understanding, the empirical successes of discrete representations in diverse areas underline their relevance and significance in the current machine learning landscape. This emphasizes the need for further research to deepen our understanding of the *challenges* and *benefits* of discrete representation learning, which is the primary focus of this thesis.

One of the challenges in learning discrete representations is the non-differentiability of discrete variables. This makes it hard to apply popular learning algorithms like stochastic gradient descent and backpropagation, which rely on smoothly (differentiably) changing the model's parameters. Additionally, optimizing over discrete structures can often be computationally challenging leading to intractable problems [21]. This is because the space of discrete structures (like graphs) is usually much larger and less smoothly structured than the space of continuous vectors, making optimization more difficult. Thus, there is a significant research interest in developing *novel methods* for learning discrete representations to utilize their potential advantages. Such methods form the secondary focus of this thesis.

1.2 Problem Statement

Discrete representations have shown great promise across various machine learning tasks. Their potential benefits range from improved interpretability of models to better generalization performance, particularly in tasks that require complex reasoning [35, 75, 91]. However, despite these advancements, significant gaps and challenges still need to be addressed in understanding and effectively training models based on discrete Stochastic Computation Graphs (SCGs) that employ such representations.

This section delves into these challenges, exploring two significant problems that the thesis aims to address. The first subsection discusses the difficulties inherent in training models based on discrete SCGs, focusing on the non-differentiability of discrete variables, the associated challenges related to training instability, and the potential facilitation of local minima. We then move on to the benefits of discrete representations, emphasizing the need for a thorough understanding of their advantages, particularly in the context of disentangled representations. By addressing these problems, we aim to provide a better understanding of the mechanics of discrete representations and enhance their practical effectiveness in machine learning tasks.

1.2.1 Challenges in Training Discrete SCGs

Training models that based on discrete SCGs poses several distinct challenges. An inherent difficulty arises from the non-differentiable nature of discrete variables. Unlike continuous variables supporting differentiability, discrete variables cannot be trained by common optimization strategies, such as stochastic gradient descent, that rely on differentiable functions and their gradients to update model parameters [89].

This non-differentiability issue has motivated the development of methods to circumvent it, such as the score function estimator [117] and the Gumbel-Softmax technique [41, 64]. The score function estimator, also known as REINFORCE, introduces a way to estimate the gradients of the expectation with respect to the parameters of the distribution over which the expectation is taken. Meanwhile, the Gumbel-Softmax technique presents a differentiable approximation, making it possible to backpropagate through categorical variables.

While these approaches address the non-differentiability of discrete variables, they open new challenges. For instance, the score function estimator is known for its high variance [67], which can lead to instability during training and slow convergence. On the other hand, the Gumbel-Softmax technique can introduce bias into the learning process because it only approximates the discrete function and does not exactly match it [41, 64]. These challenges of training models with discrete representations affect the model's performance and efficiency, underlining the need for more research into robust and effective techniques for training discrete representation models.

Prior research has focused on incorporating control variates to reduce the variance to improve the stability of the gradient estimators [109, 29]. However, the dynamics of these estimators, when applied to more complex problems such as SCGs with multiple sequential discrete distributions, are not well-understood. This thesis will show that the training instability can often be attributed to local minima and saturation, leading to insufficient gradient signals. This issue of poor gradient signals manifests as instability in training, with the model failing to converge or exhibiting erratic behavior over training iterations. Such instability not only hinders convergence but can also lead to sub-optimal solutions, thereby drastically affecting the performance of the models. We will delve into the analytical and empirical evidence of these challenges, underscoring the need to develop novel methods to mitigate them effectively.

In conclusion, the challenges inherent in training models based on discrete SCGs – from the non-differentiability of discrete variables to the instability of gradient estimators to the local minima and saturation problems – create a difficult landscape for optimization. Addressing these challenges requires approaches that ensure stable training behavior and convergence. As part of this thesis, we propose and evaluate methods designed to mitigate these issues, potentially contributing to more robust and efficient algorithms for learning discrete representations.

1.2.2 Understanding the Benefits of Discrete Representations

The rise of discrete representations in machine learning has led to significant improvements across various tasks. For instance, discrete variational autoencoders, built upon categorical distributions or vector quantization, have proven successful in large-scale image generation and model-based reinforcement learning [110, 83, 35]. Such models have also demonstrated remarkable performance in text-to-image generation tasks [81, 91]. Previous works have argued that discrete representations are a natural choice in tasks involving complex reasoning or planning, and empirical evidence indicates better generalization behavior with discrete latent spaces [41, 81, 35]. However, understanding why discrete representations are beneficial remains an open question, presenting a crucial challenge in the field.

One significant advantage of discrete representations lies in their interpretability. As discussed in the previous Section, continuous representations, such as vectors in a high-dimensional space, can often be challenging to interpret, as their meaning may heavily depend on the context of neighboring points [26]. Discrete representations, in contrast, tend to be more comprehensible [6]. For example, in natural language processing, graph representations of sentences can depict the syntactic structure of the language, which a continuous vector representation might fail to capture effectively [15]. Similarly, in program synthesis, discrete program representations can accurately reflect the control flow structure of the program, facilitating better understanding and manipulation of the represented information [76]. Thus, discrete representations can potentially enhance interpretability, making models more transparent and their predictions more trustworthy. Nonetheless, the degree to which discrete representations enhance interpretability and how this can be maximized in practice needs further exploration.

One potential framework to study the structure of latent spaces is the concept of disentanglement. Disentangled representations aim to uncover the low-dimensional and independent factors of variation in high-dimensional observations [7]. These representations are considered advantageous due to their potential for greater interpretability of learned features, fairness in predictions, and improved sample complexity for learning [39, 59]. The idea is that if a model can learn to represent data in a way that aligns with the intuitive, independent factors of variation, it can better generalize, manipulate, and explain its learned knowledge [7]. In this sense, disentanglement provides a valuable tool for comprehending the structure of latent spaces, a critical aspect of understanding the benefits of different types of representations. However, the disentanglement literature almost exclusively assumes continuous representations. The disentangling properties of discrete representations are mostly unexplored.

Despite the empirical successes of discrete representations, a comprehensive understanding of their benefits and implications, especially within the context of disentangled representations, is lacking. It remains unclear to what extent and under which conditions discrete latent spaces facilitate the learning of disentangled representations and how the specific structure of discrete spaces impacts this process. This lack of understanding represents a significant gap in the current state of the art and poses an important challenge that we address in the scope of this thesis.

1.3 Research Questions

In the previous sections, we discussed the challenges and potential benefits of discrete representations. While continuous representations have been extensively used and studied, discrete representations offer unique challenges and advantages that require deeper exploration. This exploration forms the basis of our research, where we aim to obtain a better understanding of the features of discrete representations.

The research questions presented in this section serve as a guide for this exploration. Each question is designed to uncover a specific aspect of discrete representations within discrete SCGs. These aspects are understanding their inherent challenges of training discrete representations, the benefits they provide over continuous representations, developing methods for effective training, and the integration of these representations into existing deep learning methods. In the following, we will depict and explain the research questions in more detail.

RQ1.1: *Why is the training of models based on discrete SCGs inherently challenging?*

This question seeks to identify the challenges associated with training models using discrete representations.

RQ1.2: *Why do discrete representations possess structural advantages over their continuous counterparts?*

This question explores the benefits of discrete representations over continuous ones.

RQ2.1: *How can we effectively learn the structure and the parameters of discrete SCGs?*

This question looks at potential strategies for learning the structure and the parameters of discrete SCGs improving the efficiency and stability of training.

RQ2.2: *How can discrete representations be integrated effectively into existing deep learning methods?*

This question targets the development of methods to incorporate discrete representations into existing deep learning techniques.

RQ2.3: *How can we further enhance the performance and efficiency of common discrete representations?*

This final question aims to identify innovative strategies for refining and optimizing discrete representations in deep learning applications.

RQ1.1 and **RQ1.2** seek to improve our understanding of discrete SCGs. We are especially interested in the challenges associated with training models using discrete representations and their structural advantages over their continuous counterparts.

Building on this foundation, **RQ2.1** takes a problem-solving approach. It addresses the challenge of effectively learning the structure and the parameters of discrete SCGs, leveraging the insights gained from **RQ1.1**. We aim to develop novel methods and techniques to overcome the identified challenges, targeting more efficient and stable training processes.

RQ2.2 expands on these techniques to consider their integration into existing deep learning methods. This is a crucial step towards practical application, as we seek to combine the advantages of discrete representations with established functionalities of current deep learning techniques.

Finally, **RQ2.3** aims to enhance the performance of standard discrete representations. This question seeks ways to improve discrete representations by building upon the insights and techniques developed in response to the previous questions.

These research questions are not isolated queries but form an interrelated system. Each question gradually progresses from understanding the problem and advantage of discrete representations to developing novel methods based on these insights.

1.4 Contributions

In this thesis, we explore discrete representations, uncovering the inherent challenges in their training, exploring their structural advantages, and proposing novel ways to mitigate these challenges and enhance their performance. The main contributions of this thesis are as follows:

Challenges in Training Discrete SCGs (Section 4.1): We analyze possible explanations for the inherent difficulties in training discrete SCGs based on Gumbel-Softmax distributions. These difficulties can be attributed to small gradient values leading to suboptimal training outcomes. Theorem 1 clarifies these challenges, forming the basis of our understanding in this context.

Understanding the Benefits of Discrete Representations (Section 4.2): We explore the structural benefits of discrete representations and highlight their potential impact on disentanglement in deep learning applications. Theorem 2 is a primary result of this section, spotlighting the rotational equivariance of Gaussian latent spaces and the subsequent challenges it presents to disentanglement.

Efficient Structure Learning of SCGs (Section 5.1): We propose a novel strategy of using supervised learning to address the difficulties of learning the structure of SCGs in the context of Neural Architecture Search. This strategy aims to capture the underlying distributions of the neural architecture representations, thereby effectively predicting the most promising paths in the SCG.

Efficient Parameter Learning of Discrete SCGs (Section 5.2): We propose two unique strategies to mitigate the challenges associated with training discrete SCGs, especially those involving small gradients and local minima. These strategies involve tweaking the scale parameter of the Gumbel noise perturbations and implementing dropout residual connections for discrete-continuous computation graphs.

Learning Disentangled Discrete Representations (Section 5.3): We discuss the integration of discrete representations within deep learning models and explore their disentangling properties. We propose a modified categorical VAE that uses a one-dimensional representation for each category, thereby addressing the rotational invariance issue associated with Gaussian distributions.

Improving Discrete Representations (Section 5.4): We enhance discrete representations further by leveraging findings from the disentanglement literature for Gaussian VAEs. This enhancement includes a version of the total correlation regularizer, semi-supervised training, and addressing the Straight-Through Gap.

Besides these theoretical contributions, we present our empirical contributions supporting these findings in Chapter 6.

1.5 Publications

The research undertaken throughout this doctoral thesis, centered on the exploring discrete representation in deep learning, has culminated in a series of published works. These publications, presented in a chronological order below, reflect our research progression:

Friede, D., Lukasik, J., Stuckenschmidt, H., & Keuper, M. (2019). *A variational-sequential graph autoencoder for neural architecture performance prediction*. arXiv preprint arXiv:1912.05317.

Lukasik, J., Friede, D., Stuckenschmidt, H., & Keuper, M. (2020). *Neural architecture performance prediction using graph neural networks*. In DAGM GCPR (pp. 188-201). Springer, Cham.

Friede, D., & Niepert, M. (2021). *Efficient Learning of Discrete-Continuous Computation Graphs*. Advances in Neural Information Processing Systems, 34, 6720-6732.

Friede, D., Reimers, C., Stuckenschmidt, H., & Niepert, M. (2023). *Learning Disentangled Discrete Representations*. Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD

1.6 Outline

The remainder of this thesis is organized as follows:

Chapter 2: Background This chapter lays the groundwork for understanding Stochastic Computation Graphs (SCGs), their differentiation, reparameterizations, and their application in Variational Autoencoders (VAEs). It introduces the Gumbel-Max and Gumbel-Softmax tricks and explores the concept of disentanglement in the context of VAEs.

Chapter 3: Related Work In this chapter, we examine prior work relevant to our research, focusing on some key concepts and techniques. We start with prior work on discrete representations in reinforcement learning and generative models and discuss gradient estimators for discrete distributions. We further introduce disentanglement and Neural Architecture Search.

Chapter 4: Theoretical Analysis This chapter presents our theoretical analysis of discrete representations within SCGs. We delve into the challenges of training discrete representations and explore the potential benefits of these representations, primarily in enhancing disentanglement.

Chapter 5: Methodology In this chapter, we introduce our methods for efficient structure and parameter learning of models based on discrete SCGs. We further introduce strategies for improving the integration of discrete representations within deep learning models and enhancing them using previous findings from the disentanglement literature.

Chapter 6: Experimental Evaluation In this chapter, we validate our methods for training discrete models and evaluate the benefits of discrete VAE models for unsupervised disentanglement. We demonstrate the generalization capabilities and improved interpretability of models based on discrete representations.

Chapter 7: Conclusion In the final chapter, we conclude this thesis's key findings and contributions, focussing on the insights regarding the challenges, advantages, and potential improvements of discrete representations in deep learning.

Chapter 2

Background

This chapter introduces the background of this research, focusing primarily on Stochastic Computation Graphs (SCGs) [95] and their role in unsupervised learning with Variational Autoencoders (VAEs) [45]. The theoretical foundation laid here will serve as a base to engage with the exploration and discussions that will follow in the upcoming chapters of this thesis.

Section 2.1 provides a detailed introduction to SCGs, computation graphs that combine deterministic and stochastic functions (nodes). These probabilistic computational models function as key tools in the remainder of this thesis. We discuss the wide-ranging usefulness of SCGs, underlining their flexibility in handling different computational tasks. We also introduce the application of SCGs in the Neural Architecture Search (NAS) field, explaining how SCGs can represent potential architectures.

Proceeding to Section 2.2, our attention shifts to the inner workings of SCGs. This section analyzes the process of differentiating stochastic nodes, a complex operation due to the inherent randomness within these elements.

Section 2.3 introduces an approach to managing categorical variables within SCGs. We explore using Gumbel variables for reparameterization, explaining the Gumbel-Max Trick and the Gumbel-Softmax Trick in detail [41, 64]. This procedure enables gradient-based optimization for SCGs with discrete variables.

Next, we introduce the Gumbel-Softmax distribution in Section 2.4. This distribution extends the behavior of the categorical distribution, providing a differentiable approximation [41, 64]. The section discusses this distribution's formal structure and key properties, focusing on the gradients of the sampling operation.

We conclude this chapter with Section 2.5, where we introduce a well-known application of SCGs, the Variational Autoencoder (VAE) [45]. VAEs provide a robust method for learning complex data distributions without supervision. We further discuss the concept of disentanglement, which refers to isolating the independent factors of variation within the data.

2.1 Introduction to Stochastic Computation Graphs

The loss function in many machine learning problems, originating from supervised, unsupervised, and reinforcement learning, is defined by an *expectation* over a collection of random variables. These random variables might be part of a probabilistic model or the external world and exhibit different behavior than the deterministic nodes in standard computation graphs. Stochastic Computation Graphs (SCGs) were first introduced by Schulman et al. [95] as a general framework for modeling and optimizing differentiable computation graphs with such stochastic nodes. SCGs provide a flexible way to model the complex relationships between different types of variables by treating stochastic nodes in the computation graph separately. They enable end-to-end training with gradient-based optimization methods, which is beneficial when learning challenging tasks involving discrete decisions.

Discrete representations are often represented stochastically using the score function estimator [117] or the Gumbel-softmax trick [41, 64]. The score function estimator, also known as REINFORCE, is a method for estimating gradients of expectations with respect to parameters of a model. For instance, in reinforcement learning, it estimates gradients of the expected reward concerning the parameters of a policy. The Gumbel-softmax trick is a continuous relaxation of discrete probability distributions that allows for gradient-based optimization. Thus, SCGs are a straightforward choice for modeling models with discrete representations.

In practice, SCGs are directed acyclic graphs (DAGs) consisting of deterministic functions and conditional probability distributions. The deterministic functions define the relationship between the deterministic nodes in the graph, while the conditional probability distributions describe the stochastic nodes. Figure 2.1 illustrates an example. The SCG combines information from the input nodes, representing the MNIST images of handwritten digits, through a series of deterministic and stochastic nodes. The deterministic nodes, denoted by diamond shapes, perform transformations or computations on the data, often capturing the structural dependencies in the problem. The stochastic nodes, represented by blue circles, capture the inherent uncertainty in the model, and their conditional probability distributions can be learned or inferred from the data. The edges in the SCG indicate the flow

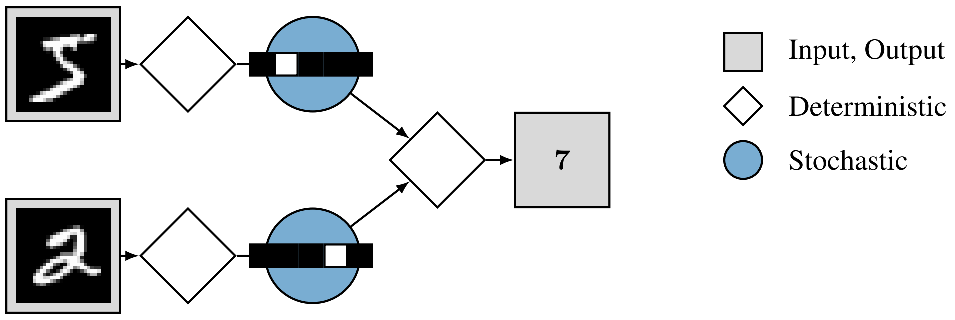


Figure 2.1: A stochastic computation graph (SCG) representation of a model that predicts the sum of two MNIST images. The input nodes contain two MNIST images, one representing the digit 5 and the other representing the digit 2. The model processes these inputs through deterministic (diamond-shaped) and stochastic (circle-shaped) nodes, with a black-and-white pattern indicating the probability distributions representing the values of the digits seen on the individual MNIST images. The output node contains the predicted sum, 7, in this case.

of information between the nodes, with the direction of the arrows representing the dependencies between the nodes. In this example, the SCG models the addition of two handwritten digits, with the stochastic nodes representing the discrete classes of the 10 MNIST digits. The output node contains the predicted sum based on the learned discrete classes.

SCGs are a powerful modeling framework, as they can be adapted to various applications, from simple arithmetic operations to more complex tasks, such as learning variational autoencoders (VAEs), neural architecture search (NAS), and decision-making under uncertainty. By leveraging both deterministic and stochastic components, SCGs can effectively capture and represent the underlying structure and uncertainty in various real-world problems [95].

2.1.1 Levels of Difficulty in Stochastic Computation Graphs

In this section, we will explore various levels of complexity in SCGs, ranging from deterministic computation graphs to hierarchical SCGs with internal graph execution. The goal is to provide a comprehensive understanding of the different challenges and tasks that will be tackled with SCGs throughout this thesis. Figure 2.2 illustrates four levels of SCGs with increasing difficulty. As we progress from left to right, the complexity of the SCGs increases. We take inspiration from Jang et al. [41] to visualize the SCGs in the following figures.

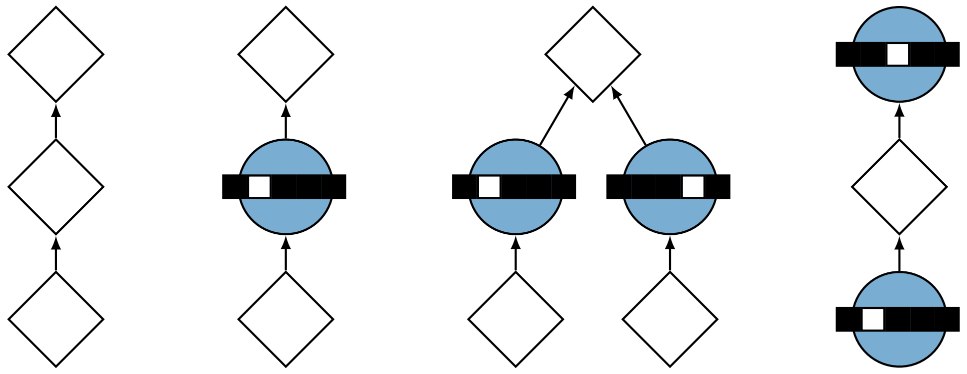


Figure 2.2: Illustration of four SCG levels with increasing difficulty. From left to right: **(1)** Deterministic Computation Graph, featuring a path of deterministic nodes; **(2)** Simple SCG, composed of a path including one stochastic node (blue circle); **(3)** SCG with two separate paths with stochastic nodes converging into a single deterministic node; and **(4)** SCG with two nodes on the same path, which demonstrates a single path containing two stochastic nodes on the same path.

The deterministic computation graph (1) is the simplest form, consisting only of deterministic nodes. In this case, the computation is completely predictable, and the graph can be traversed without any uncertainty.

The simple SCG (2) introduces a stochastic node (blue circle) into the computation graph. This node represents a point of uncertainty, where multiple outcomes are possible with varying probabilities. The presence of stochastic nodes increases the difficulty in determining the optimal path or solution to a given problem.

In the third complexity level, the SCG consists of two separate paths with stochastic nodes that eventually converge (3). This structure poses a challenge in determining the optimal combination of the two paths, as the outcomes of both stochastic nodes influence the final result. The sum of two MNIST images, as illustrated in Figure 2.1, serves as an example of such an SCG.

The fourth complexity level features an SCG with two stochastic nodes on the same computation path (4), further complicating the problem-solving process. The outcome of one stochastic node can influence the outcome of the subsequent stochastic node, resulting in a complex interplay of probabilities. An example of such an SCG is the 3-MNIST addition problem, as illustrated in Figure 2.3.

The fifth complexity level features a hierarchical SCG with an internal SCG as a stochastic node (5), as illustrated in Figure 2.4. In this scenario, the external SCG

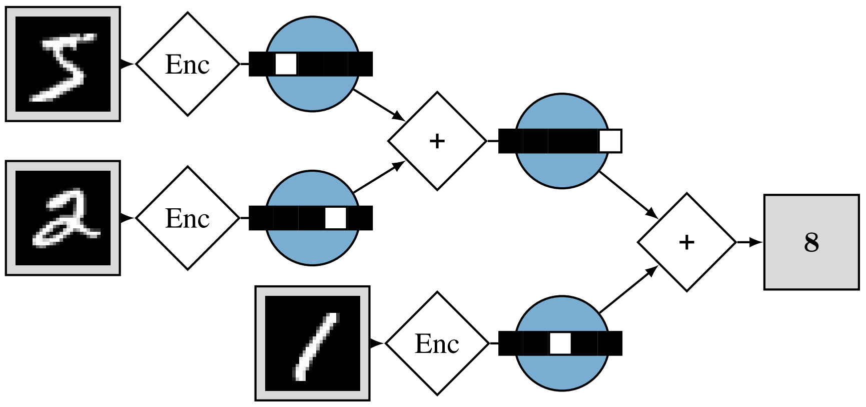


Figure 2.3: Illustration of an SCG for the 3-MNIST addition problem. Three input MNIST images are processed by deterministic encoder nodes (Enc), followed by stochastic nodes. The first two input images converge and are channeled through a second stochastic node on the same path, representing the sum of the first two digits. The third input image is processed in a separate path. The outputs of these paths converge into a second deterministic addition node (+), showcasing the compositionality of the model.

processes the input – [max 2 9] – through a deterministic node and a stochastic node containing the internal SCG. The internal graph represents the parsing of the tokens [max], [2], and [9] as nodes. This structure poses a challenge in learning the correct graph structure and executing it within the external SCG to produce the desired output. The graph traversal continues through the inner SCG, and the connection between the outer and inner SCG is indicated by a dashed arrow. Ultimately, the output 9 is produced after traversing both the external and internal SCGs. In this level of complexity, not only does the model need to account for stochastic nodes and their interplay of probabilities, but it also has to learn and execute an internal graph structure as part of the overall computation in the external SCG. This represents a more complex task, requiring a deeper understanding of the underlying problems and a more sophisticated problem-solving approach.

2.1.2 Neural Architecture Search in the Context of Stochastic Computation Graphs

One application of Stochastic Computation Graphs (SCGs) is their use in Neural Architecture Search (NAS), an area of machine learning focusing on automating the design of neural network architectures. NAS leverages stochastic search strategies to explore the vast space of potential architectures, making SCGs a natural fit

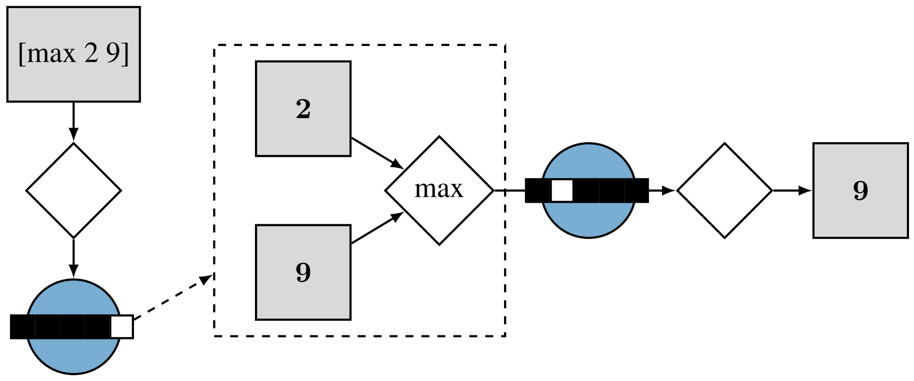


Figure 2.4: Illustration of a hierarchical SCG (5) with an internal SCG as a stochastic node. The external SCG processes the input – [max 2 9] – through a deterministic node followed by a stochastic node. Inside the stochastic node, an internal graph is learned, representing the parsing of the tokens [max], [2], and [9] as nodes. The dashed arrow indicates the connection between the outer and inner SCG. The graph traversal continues through the inner SCG, ultimately producing the output 9.

for representing and guiding this search process.

In the context of NAS, the SCG represents the possible neural network architectures in an internal graph and the selection of a specific architecture in the external graph. The stochastic nodes represent the randomness of selecting specific architecture choices. These often discrete distributions range from categorical for the operation choice, e.g., a convolutional layer or max pooling, to binary for the edge indication. These operations form the internal nodes of the SCG, whereas the internal edges between these nodes represent the flow of data through the neural network. The challenge in NAS lies in learning the optimal architecture, which essentially boils down to learning the optimal internal computation graph that, when followed, would form the best-performing architecture for the given task.

This use of SCGs in NAS is an example of the fifth level of complexity mentioned in Section 2.1.1, involving the learning of an internal graph structure (the neural architecture) as part of the overall computation in the external SCG. Hence, the associated complexities and challenges in optimizing such SCGs in NAS include managing uncertainty related to the stochastic nodes when learning the graph structure representing the optimal neural architecture.

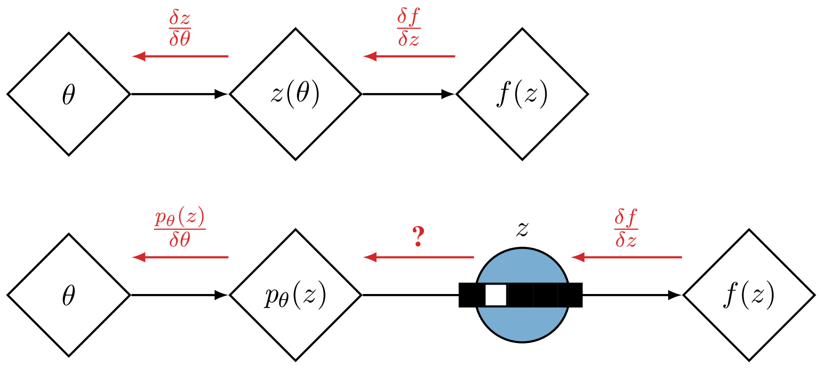

$\frac{\delta z}{\delta \theta}$
$\frac{\delta f}{\delta z}$
θ
$z(\theta)$
$f(z)$
$\frac{p_\theta(z)}{\delta \theta}$
?
z
$\frac{\delta f}{\delta z}$
θ
$p_\theta(z)$
$f(z)$

Figure 2.5: Illustration of the gradients of a deterministic computation graph (top) and a stochastic computation graph (bottom). **Top:** In the deterministic graph, gradients flow smoothly from the function f back to the parameter θ (red arrows). **Bottom:** In the stochastic computation graph, a stochastic node z introduces complexity, as it draws from a distribution $p_\theta(z)$ determined by θ. This introduces a problem when trying to compute the gradient from f through z to θ.

2.2 Differentiating through Stochastic Nodes

Stochastic nodes introduce a layer of complexity into computation graphs that does not exist in deterministic computation graphs. A stochastic node in a computation graph represents a random variable or a distribution over potential outcomes. When we attempt to compute the gradient of the expectation of such a node during backpropagation, we encounter issues such as non-existent closed forms or non-differentiability [68].

To mathematically express the issue of calculating gradients with respect to the parameters of a distribution, let us formulate some initial equations. Assume f is a smooth function (such as the loss function in a learning problem). The gradient of f at the stochastic node can be expressed as:

$$\nabla_\theta \mathbb{E}_{p_\theta(z)}\left[f(z)\right] = \nabla_\theta \int_z f(z) p_\theta(z) dz. \tag{2.1}$$

Here, we rewrite the expectation as an integral over z. The bottom half of Figure 2.5 illustrates the computation graph for this operation, highlighting the flow of gradients through the nodes of the SCG. The stochastic node, represented by z, introduces randomness, which complicates the computation of gradients. Under certain conditions, we can apply Leibniz's rule for differentiation under the integral sign to switch the gradient and the integral in Equation (2.1), yielding:

$$\nabla_\theta \mathbb{E}_{p_\theta(z)}\left[f(z)\right] = \int_z f(z)\nabla_\theta p_\theta(z)dz. \tag{2.2}$$

Unfortunately, the right-hand side of Equation (2.2) typically does not possess a closed-form expectation. This is because, in general, the gradient of a density function is not itself a density function. Thus, a straightforward Monte Carlo estimation is not applicable.

Instead, we need to seek alternative methods to deal with this challenge, like the score function estimator [117] or the reparameterization trick [45]. These methods introduce a way to bypass or handle the problem of calculating gradients with respect to the parameters of a distribution caused by the stochastic nodes, allowing us to perform backpropagation effectively. We will explore these methods in more detail and highlight their advantages and limitations.

2.2.1 The Score Function Estimator

The Score Function Estimator (SFE), also referred to as the REINFORCE algorithm in the context of reinforcement learning [117], is a method for estimating gradients through stochastic nodes. SFE utilizes the log-derivative trick:

$$\nabla_\theta p_\theta(z) = p_\theta(z)\nabla_\theta \log p_\theta(z). \tag{2.3}$$

This trick allows the gradient to be reformulated such that it can be computed using samples from the distribution:

$$\nabla_\theta \mathbb{E}_{p_\theta(z)}\left[f(z)\right] = \mathbb{E}_{p_\theta(z)}\left[f(z)\nabla_\theta \log p_\theta(z)\right]. \tag{2.4}$$

The gradient flow in this estimator, as depicted in Figure 2.6, bypasses the stochastic node by backpropagating along the surrogate loss $f\nabla \log p_\theta(z)$. The expectation on the right-hand side of Equation (2.4) can be estimated via Monte Carlo methods, as it forms a closed-form expectation. Crucially, this expectation relies solely on the score function, $\nabla_\theta \log p_\theta(z)$, rather than the derivative of the function $f(z)$ itself. This feature renders the SFE highly flexible and broadly applicable to various types of stochastic nodes, such as discrete distributions. However, despite its broad applicability, SFE is often prone to high variance, which can hinder learning or even render it unstable [30].

Prior research has focused on modifying the SFE to reduce its variance. This is accomplished by incorporating control variates, denoted as $b(z)$, into the SFE. The modified SFE then becomes:

$$\nabla_\theta \mathbb{E}_{p_\theta(z)}\left[f(z)\right] = \mathbb{E}_{p_\theta(z)}\left[\left(f(z) - b(z)\right)\nabla_\theta \log p_\theta(z)\right] + \mu_b, \tag{2.5}$$

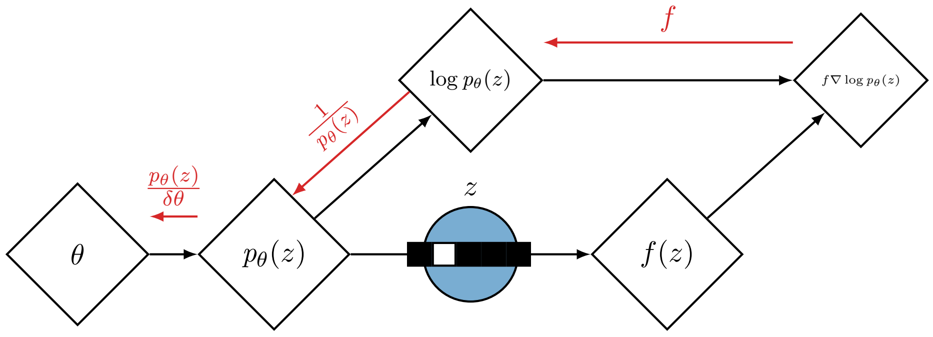


Figure 2.6: Visual explanation of the Score Function Estimator and its gradient flow. The gradient bypasses the stochastic node by differentiating the surrogate loss $f\nabla \log p_\theta(z)$. The exact gradients can be derived by the log-derivative trick.

where $\mu_b = \mathbb{E}_{p_\theta(z)}\left[b(z)\nabla_\theta \log p_\theta(z)\right]$ represents the analytical expectation of the control variate. This inclusion ensures that the estimator remains unbiased. Techniques to reduce variance in score function estimators are diverse, spanning from straightforward baselines [82, 67] to the Rao-Blackwellization method [84, 104], and gradient-based control variates [103].

A notable issue with the variance of SFE is its linear scaling with the number of dimensions in the sample vector [87]. This characteristic poses distinct challenges when the estimator is applied to categorical distributions [41]. While the reparameterization trick [45] may be less flexible than SFE since its functionality depends on certain conditions being met, it often provides powerful estimators with advantageous attributes, such as a reduced variance [95].

2.2.2 The Reparameterization Trick

The Reparameterization Trick [45] offers a more efficient approach for certain types of stochastic nodes. Instead of directly differentiating the stochastic node, the Reparameterization Trick introduces an auxiliary noise variable, allowing the gradient to pass through deterministic nodes. For a class of distributions known as location-scale families (which includes the Gaussian distribution), the Reparameterization Trick provides a lower-variance gradient estimator compared to the Score Function Estimator [45, 95]. However, the Reparameterization Trick's major limitation is that it is not universally applicable. It can only be applied to distributions that can be reparameterized in the following way. The idea is to find a function g and distribution ρ such that one can replace $z \sim p_\theta(z)$ by $z = g(\epsilon, \theta)$

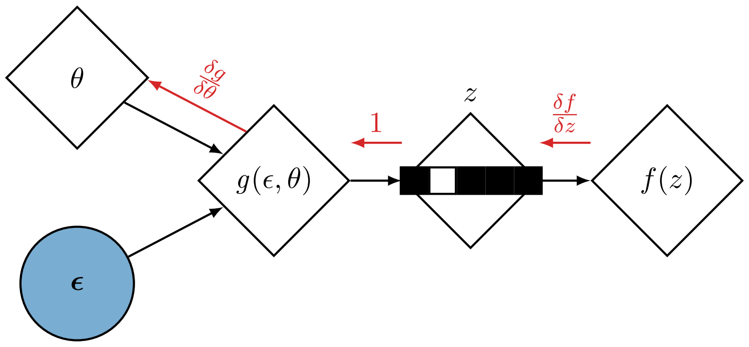


Figure 2.7: Visualization of the Reparameterization Trick applied in a Stochastic Computation Graph. The parameters θ and the noise term ϵ are transformed by a deterministic function $g(\epsilon, \theta)$ to generate a reparameterized random variable z. The gradients can flow back from the function $f(z)$ to the parameters θ, bypassing the stochastic node, as indicated by the red arrows.

with $\epsilon \sim \rho(\epsilon)$. If this is possible, then one can write

$$\nabla_\theta \mathbb{E}_{z \sim p_\theta(z)} [f(z)] = \mathbb{E}_{\epsilon \sim \rho(\epsilon)} [\nabla_\theta f (g(\epsilon, \theta))] . \tag{2.6}$$

Figure 2.7 visually depicts the Reparameterization Trick. The introduced auxiliary noise variable, ϵ, is sampled from a base distribution, $\rho(\epsilon)$. The deterministic node $g(\epsilon, \theta)$ computes the value of z from the noise variable and the parameters θ. The gradient flows directly through the deterministic nodes, effectively bypassing the stochastic node.

To fully take advantage of the Reparameterization Trick, two key conditions must be met. Firstly, the function $f(z)$ must be differentiable with respect to its input z. This is a crucial difference from the Score Function Estimator and restricts the types of functions that can be used in this context. Secondly, a function $g(\epsilon, \theta)$ must exist and be differentiable with respect to θ. This function maps the noise variable ϵ and the model parameters θ to a new variable $z = g(\epsilon, \theta)$.

The Gaussian distribution serves as an example where a reparameterization can be obtained. Consider a sample z drawn from a univariate Gaussian distribution $p_\theta(z) = \mathcal{N}(\mu, \sigma^2)$. Given that the family of Gaussian distributions is closed under linear transformations, we can reparameterize $z = \mu + \sigma\epsilon$ with $\epsilon \sim \mathcal{N}(0, 1)$. Therefore, we can reformulate the expectation under the original distribution in terms of the expectation under the reparameterized distribution:

$$\mathbb{E}_{\mathcal{N}(z;\mu,\sigma^2)} [f(z)] = \mathbb{E}_{\mathcal{N}(\epsilon;0,1)} [f(\mu + \sigma\epsilon)] \tag{2.7}$$

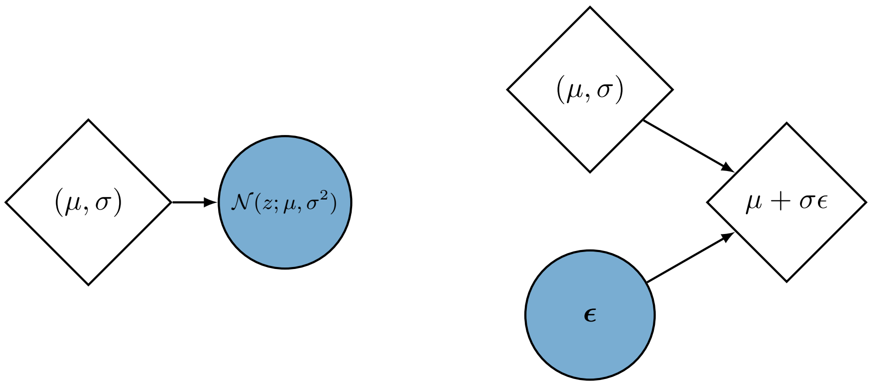


Figure 2.8: Reparameterization of a Gaussian distribution using the Reparameterization Trick. **Left:** The traditional Gaussian distribution is shown, where the parameters (μ, σ) generate a stochastic node z. **Right:** The Gaussian distribution is reparameterized as $z = \mu + \sigma\epsilon$ using the auxiliary variable $\epsilon \sim \mathcal{N}(0, 1)$.

Figure 2.8 illustrates this process of reparameterization for the Gaussian distribution. On the left, the original stochastic node z is drawn from a Gaussian distribution parameterized by (μ, σ). On the right, we see the reparameterization of z as a deterministic function of the noise variable ϵ and the parameters (μ, σ). This transformation enables the gradient of the expectation $\mathbb{E}\left[f(z)\right]$ to be computed more efficiently since the dependency on the parameters is now entirely within deterministic nodes.

The existence and differentiability of this function $g(\epsilon, \theta)$ is not always guaranteed, especially for discrete categorical variables. Indeed, the challenge of applying the Reparameterization Trick to discrete variables has motivated the development of techniques such as the Gumbel-Softmax Trick [41, 64] that is based on the Gumbel-Max Trick [33].

The Gumbel-Max Trick [33] offers a way to construct the function $g(\epsilon, \theta)$ for discrete categorical variables. It involves adding Gumbel-distributed noise to the logits of a categorical distribution and then taking the argmax operation to draw a sample. However, because of the discrete nature of the argmax operation, the resulting function $g(\epsilon, \theta)$ is not differentiable with respect to θ, which makes it challenging to use in gradient-based optimization methods. Recognizing this limitation, the Gumbel-Softmax Trick [41, 64] was developed as a relaxation of the Gumbel-Max Trick. It replaces the non-differentiable argmax operation with a differentiable softmax operation, which allows the gradients to be computed and backpropagated through. The resulting function $g(\epsilon, \theta)$ is differentiable with respect to θ, fulfilling the requirements of the Reparameterization Trick. However, the Gumbel-Softmax Trick introduces a bias because the softmax operation returns a probability distribution over all categories rather than a single categorical outcome, as in the Gumbel-Max Trick. This can lead to a discrepancy between the true distribution and the approximated one, which could potentially affect the learning process and the quality of the results.

We will delve deeper into these two techniques and their applications in the following section. Despite their limitations, these tricks broaden the range of distributions to which reparameterization can be applied.

2.3 Reparameterizations using Gumbel Variables

The Reparameterization Trick and its application to continuous variables have numerous benefits, including providing a lower variance gradient estimator. However, in real-world situations, we often work with discrete random variables. These pose a significant challenge due to their non-differentiable nature. In the previous section, we briefly introduced the Gumbel-Softmax Trick, a method developed to tackle this issue and extend the benefits of reparameterization to discrete categorical variables.

This section will delve deeper into the Gumbel-Softmax Trick and its predecessor, the Gumbel-Max Trick. We will explain in detail how they work and successfully handle the problem of differentiability with discrete variables, increasing the scope of reparameterization in deep generative modeling.

2.3.1 The Gumbel-Max Trick

The concept of the Gumbel-Max Trick extends the Reparameterization Trick to handle discrete variables [33]. Named after Emil Julius Gumbel, who first introduced the Gumbel distribution in extreme value theory, this trick presents an innovative approach for sampling from a discrete categorical distribution [32]. However, it does come with a notable drawback, the non-differentiability issue, which limits its direct use in gradient-based learning methods.

Let $z \sim \mathrm{Cat}(\boldsymbol{\theta})$ be a discrete variable distributed according to a categorical distribution with unnormalized category probabilities $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_m)$ with $\theta_k > 0$ for all $k = 1, \ldots, m$, where m is the number of categories. This means that

$$\mathbb{P}(z = j \mid \boldsymbol{\theta}) = \frac{\theta_j}{\sum_k^m \theta_k}. \tag{2.8}$$

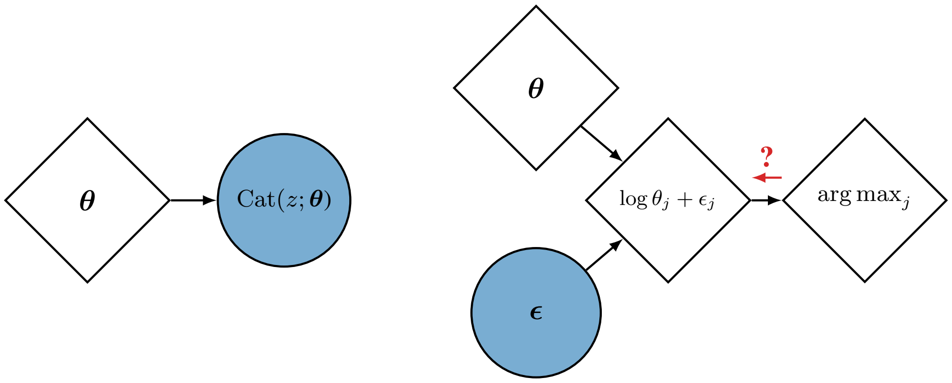


Figure 2.9: Schematic representation of the standard approach and the Gumbel-Max Trick for sampling from a categorical distribution. **Left:** The standard approach directly uses the unnormalized category probabilities $\boldsymbol{\theta}$ to sample from the categorical distribution $\mathrm{Cat}(z;\boldsymbol{\theta})$. **Right:** The Gumbel-Max Trick introduces Gumbel-distributed noise variables $\boldsymbol{\epsilon}$. These variables combine with the $\log\theta_j$ values to pass through an argmax operation, which selects the index of the maximum value, thereby creating a reparameterization of the categorical variable z. The red question mark signifies the non-differentiability issue of the argmax operation.

The Gumbel-Max Trick uses the Gumbel distribution $\epsilon \sim \mathrm{Gumbel}(0,1)$, which is defined by

$$\epsilon = -\log\left(-\log(u)\right), \tag{2.9}$$

where $u \sim \mathrm{Uniform}(0,1)$, to provide a reparameterization of the categorical distribution. Gumbel [33] showed that we can reparameterize $z \sim \mathrm{Cat}(\boldsymbol{\theta})$ by

$$z = \arg\max(\log\theta_j + \epsilon_j), \tag{2.10}$$

where $\epsilon_j \sim \mathrm{Gumbel}(0,1)$ for all $j = 1,\ldots,m$. This reparameterization approach, the Gumbel-Max Trick, is visualized on the right side of Figure 2.9. The graph begins with the unnormalized category probabilities $\boldsymbol{\theta}$ for a categorical distribution. Each noise variable ϵ_j corresponds to a specific category j in the categorical distribution. The index of the maximum value in this array corresponds to the sample z drawn from the categorical distribution, enabling a reparameterization of the categorical distribution.

However, the presence of the argmax operation introduces a significant difficulty in the context of gradient-based learning. The argmax operation is non-differentiable, which means it does not allow the flow of gradient information. This limitation is problematic because it prevents the direct application of the Reparameterization

Trick, which requires the differentiability of the function $g(\epsilon, \theta)$. The following subsection will show how the Gumbel-Softmax Trick addresses this problem.

2.3.2 The Gumbel-Softmax Trick

Recognizing the limitation posed by the non-differentiability of the argmax operation in the Gumbel-Max Trick, a more flexible approach was introduced, known as the Gumbel-Softmax Trick [41, 64]. This technique is essentially a relaxation of the Gumbel-Max Trick that makes it suitable for gradient-based optimization methods. In particular, the Gumbel-Softmax Trick replaces the non-differentiable argmax operation with the differentiable *softmax operation*. This change allows for the computation of gradients, enabling backpropagation through the sampling procedure.

The softmax function is defined as

$$\mathrm{softmax}(\boldsymbol{x})_j = \frac{\exp(x_j)}{\sum_{k=1}^{m} \exp(x_k)} \tag{2.11}$$

for a given vector $\boldsymbol{x} = (x_1, \ldots, x_m)$.

The reparameterization of a categorical variable z using the Gumbel-Softmax Trick is then given by

$$\boldsymbol{z} = \mathrm{softmax}((\log \boldsymbol{\theta} + \boldsymbol{\epsilon})/\tau), \tag{2.12}$$

where $\tau > 0$ is a temperature parameter that controls the sharpness of the distribution and $\epsilon_j \sim \mathrm{Gumbel}(0, 1)$ for all $j = 1, \ldots, m$. Note that $\boldsymbol{z}$ in Equation (2.12) is a vector of size m, the number of categories, and normalized. For $\tau \to 0$, the softmax function tends towards the argmax function, making the Gumbel-Softmax Trick reduce to the Gumbel-Max Trick [41, 64], if we represent the categories as one-hot vectors. We will discuss this in more detail in Section 2.4. Conversely, for $\tau \to \infty$, the output of the softmax function approaches a uniform distribution [41]. This process is visualized in Figure 2.10, which depicts the steps in the Gumbel-Softmax Trick. The softmax operation provides a differentiable transformation, turning the categorical variable z into an approximated representation. This allows for backpropagation through the sampling process, which makes it possible to optimize the parameters $\boldsymbol{\theta}$ with respect to the loss function using gradient-based methods.

Figure 2.11 illustrates the effect of different temperature values on the Gumbel-Softmax distribution. Each subfigure presents a ternary plot representing the distribution of a categorical variable reparameterized using the Gumbel-Softmax Trick with a specific τ value. As τ increases, the distribution becomes smoother and

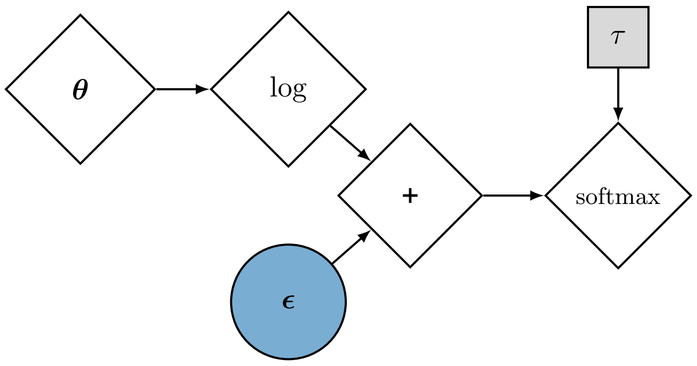


Figure 2.10: Schematic representation of the Gumbel-Softmax Trick. Here, $\boldsymbol{\theta}$ is the unnormalized category probabilities of a categorical distribution, and $\boldsymbol{\epsilon}$ are Gumbel-distributed noise variables. The $\log \boldsymbol{\theta}$ values and $\boldsymbol{\epsilon}$ are summed and passed through a softmax function, yielding a differentiable approximation of the categorical distribution. The τ symbol denotes the temperature parameter, which modulates the sharpness of the resulting distribution.

approaches a uniform distribution. Conversely, as τ decreases, the distribution becomes sharper and tends towards the original discrete distribution determined by the unnormalized category probabilities $\boldsymbol{\theta}$.

While the Gumbel-Softmax Trick provides a differentiable approximation of the Gumbel-Max Trick, it introduces a bias in the sampling process [64]. The softmax operation returns a probability distribution rather than a single categorical outcome, resulting in a continuous approximation to the original discrete distribution. This may cause a discrepancy between the true and the approximated distribution.

To reduce the variance of the gradient estimator used in the Gumbel-Softmax Trick, prior research [109, 29] has focused on incorporating parameterized control variates, as introduced in Equation (2.5). These control variates can theoretically be generalized to chains of stochastic nodes [109, 29]. However, the learning behavior of these estimators for the more complex SCGs that we have discussed in Section 2.1 is not extensively discussed in the literature.

In this thesis, we focus on efficiently training models with gradient estimates for categorical distributions and understanding their advantages. Section 4.1 discusses the challenges posed by the Gumbel-Softmax Trick, particularly in the context of complex discrete-continuous computation graphs with multiple sequential discrete distributions. In Section 5.2, we aim to improve the training efficiency and performance of these models by investigating novel methods and techniques.

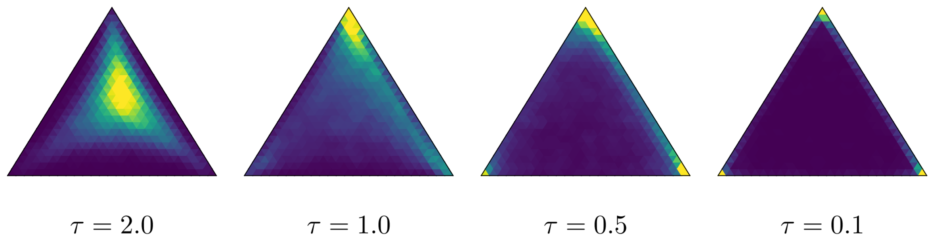


Figure 2.11: Visualization of the Gumbel-Softmax Trick for the unnormalized probabilities $\boldsymbol{\theta} = (2.0, 0.5, 1.0)$ with varying temperature parameter τ. Each subfigure represents a ternary plot showing the distribution of a categorical variable reparameterized using the Gumbel-Softmax Trick. Yellow depicts high, and blue depicts low values. As τ increases, the distribution becomes smoother and approaches a uniform distribution. Conversely, as τ decreases, the distribution becomes sharper and tends towards the categorical distribution.

2.4 The Gumbel-Softmax Distribution

The Gumbel-Softmax Trick yields a unique distribution introduced for modeling discrete data. This section presents a detailed description of the main features of this distribution, mainly following Maddison et al. [64].

The random vector $\boldsymbol{z}$ of the Gumbel-Softmax distribution follows a simplex constraint such that $\boldsymbol{z} \in \Delta^{m-1} = \{\boldsymbol{z} \in \mathbb{R}^m \mid z_j \in [0,1], \sum_{j=1}^m z_j = 1\}$, which reflects the fact that the vector $\boldsymbol{z}$ represents probabilities.

Definition 1 (Gumbel-Softmax Distribution)**.** *Given unnormalized category probabilities $\boldsymbol{\theta} \in (0,\infty)^m$ and temperature $\tau > 0$, a random variable $\boldsymbol{z} \in \Delta^{m-1}$ follows a Gumbel-Softmax distribution* $\mathrm{GS}(\boldsymbol{\theta}, \tau)$ *if its probability density function is given by*

$$p_{\boldsymbol{\theta},\tau}(\boldsymbol{z}) = (m-1)!\tau^{m-1} \prod_{k=1}^{m} \frac{\theta_k z_k^{-\tau-1}}{\sum_{j=1}^m \theta_j z_j^{-\tau}}. \tag{2.13}$$

The density function is determined by the unnormalized category probabilities $\boldsymbol{\theta}$, the temperature τ, and the particular values of the random vector $\boldsymbol{z}$.

2.4.1 Properties of the Gumbel-Softmax Distribution

The following proposition features some key features of the Gumbel-Softmax distribution as introduced in Maddison et al. [64].

Proposition 1 (Properties of the Gumbel-Softmax Distribution). *Let $\boldsymbol{z}$ be a random variable following the Gumbel-Softmax distribution, i.e., $\boldsymbol{z} \sim \mathrm{GS}(\boldsymbol{\theta}, \tau)$ with unnormalized category probabilities $\boldsymbol{\theta} \in (0, \infty)^m$ and temperature $\tau > 0$. Then, it holds that*

1. *(Reparameterization) If* $\epsilon_k \sim \mathrm{Gumbel}(0, 1)$*, then* $\boldsymbol{z} = \frac{\exp(\log \theta_j + \epsilon_j)}{\sum_{k=1}^{m} \exp(\log \theta_k + \epsilon_k)}$,
2. *(Zero temperature)* $\mathbb{P}(\lim_{\tau \to 0} z_k = 1) = \frac{\theta_k}{\sum_{j=1}^{m} \theta_j}$,
3. *(Jacobian)* $\frac{\partial z_i}{\partial \theta_j} = \begin{cases} \frac{1}{\tau}\, z_i(1 - z_i) & \textit{if } i = j \\ -\frac{1}{\tau}\, z_i z_j & \textit{if } i \neq j. \end{cases}$

The first property shows that one can sample from the Gumbel-Softmax distribution using the previous section's reparameterization, making it possible to propagate gradients through the sampling operation.

The second property relates to the behavior of the Gumbel-Softmax distribution as the temperature parameter τ approaches zero. Specifically, as τ decreases, the probability of the random variable z_k being one tends to the proportion of the unnormalized category probability θ_k among all category probabilities. This means that the Gumbel-Softmax distribution converges to a categorical distribution in the zero-temperature limit, showcasing the link between these two distributions.

The third property, the Jacobian, relates to the derivatives of the components of the random vector $\boldsymbol{z}$ with respect to the parameters θ_j. This property plays a key role in understanding the challenges in training discrete representations in Section 4.1.

2.5 Variational Autoencoders

In Section 2.1.1, we explored various levels of complexity within Stochastic Computation Graphs (SCGs). One common feature of the models discussed was that the training process was guided by the relationship between the input and corresponding labels. The model, therefore, needed to capture the internal structure of the data (e.g., the appropriate categories for the stochastic nodes) and the external structure (e.g., the correct parsing of the partial computation graphs) in order to connect the input to the corresponding labels.

However, a greater challenge lies in utilizing SCGs to manage data inputs in an entirely unsupervised manner – without the guidance of any labels. This leads us to Variational Autoencoders (VAEs), which strive to learn a low-dimensional latent representation of the input data, capturing the underlying structure of the data distribution [45].

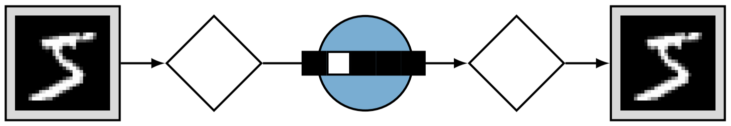


Figure 2.12: This diagram presents a typical Variational Autoencoder (VAE). Given an input image from the MNIST dataset, the image is processed through a deterministic encoder, represented by the diamond shape. This encoder translates the high-dimensional data into a lower-dimensional latent space. The blue circle represents the stochastic node, which models the inherent randomness of the data using a learned distribution. The image is then reconstructed through a deterministic decoder (also a diamond shape), resulting in an output image that is ideally close to the original input image.

VAEs, introduced by Kingma and Welling [45], provide a probabilistic method for describing an observation in latent space. Essentially, they are an extension of Autoencoders, neural networks trained to reconstruct their input data [4], but with an added constraint to the encoding part. Specifically, VAEs enforce the latent variables to follow a specified distribution. This constraint encourages the model to learn a more robust representation of the data and enhances its generative capabilities [45]. Figure 2.12 illustrates an example of a VAE.

The primary objective of a VAE is to learn the posterior distribution of the latent variables given the input data. This objective is achieved by maximizing the *evidence lower bound* (ELBO), which entails two components: the reconstruction loss and the Kullback-Leibler (KL) divergence loss. The reconstruction loss measures the discrepancy between the original input and the reconstructed output, while the KL-divergence loss quantifies the difference between the learned latent distribution and the prior. Training a VAE is typically addressed with the reparameterization trick as introduced in Section 2.2.2.

2.5.1 The Evidence Lower Bound

The representation learning literature is usually premised on the assumption that a high-dimensional observation $\boldsymbol{x}$ from the data space $\mathcal{X}$ is generated from a low-dimensional latent variable $\boldsymbol{z}$ whose entries correspond to the dataset's ground-truth factors of variation such as position, color, or shape [7, 108]. First, the ground-truth factors are sampled from some distribution $\boldsymbol{z} \sim p(\boldsymbol{z}) = \prod p(z_i)$. The observation is then a sample from the conditional probability $\boldsymbol{x} \sim p(\boldsymbol{x}|\boldsymbol{z})$.

The formalism of variational autoencoders [45] enables an estimation of these distributions. Assuming a known prior $p(\boldsymbol{z})$, we can depict the conditional probability $p_\psi(\boldsymbol{x}|\boldsymbol{z})$ as a parameterized probabilistic decoder. In general, the posterior $p_\psi(\boldsymbol{z}|\boldsymbol{x})$ is intractable. Thus, we turn to variational inference and approximate the posterior by a parameterized probabilistic encoder $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ and minimize the Kullback-Leibler (KL) divergence $D_{\mathrm{KL}}\big(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \parallel p_\psi(\boldsymbol{z}|\boldsymbol{x})\big)$. This term, too, is intractable but can be minimized by maximizing the evidence lower bound (ELBO)

$$\mathcal{L}_{\psi,\phi}(\boldsymbol{x}) = \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_\psi(\boldsymbol{x}|\boldsymbol{z})\right] - D_{\mathrm{KL}}\big(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \parallel p(\boldsymbol{z})\big). \tag{2.14}$$

At its core, the ELBO aims to make the modeling of $\boldsymbol{z}$ more accurate while making the encoder $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ close to the true but intractable posterior $p_\psi(\boldsymbol{z}|\boldsymbol{x})$. The first term of Equation (2.14) is the expected log likelihood $\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_\psi(\boldsymbol{x}|\boldsymbol{z})\right]$. It captures how well the decoder can reconstruct the input $\boldsymbol{x}$ given a latent code $\boldsymbol{z}$ sampled from the encoder's distribution. A higher value for this term means that the decoder is better at reconstruction, so we want to maximize this term. The second term is the KL divergence $D_{\mathrm{KL}}\big(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \parallel p(\boldsymbol{z})\big)$. This term measures the difference between the distribution of the encoder and the prior. We aim to minimize this term, encouraging the encoder to produce latent codes $\boldsymbol{z}$ that closely match the prior distribution $p(\boldsymbol{z})$.

Maximizing the ELBO leads to a model that can reconstruct its input data well while ensuring that the learned representations stick to a structure chosen in advance (the prior). Higgins et al. [38] discussed the latent space structure in detail for the diagonal Gaussian prior within the concept of *disentanglement*. In this thesis, we build upon the idea of [38] by questioning the default use of a Gaussian prior and discussing the advantages of a discrete, categorical prior.

2.5.2 Disentangled Representations

The disentanglement literature [7, 39, 59] provides a common approach to analyzing the structure of latent spaces. Disentangled representations [7] recover the low-dimensional and independent ground-truth factors of variation of high-dimensional observations. Such representations promise interpretability [39, 1], fairness [58, 17, 106], and better sample complexity for learning [93, 7, 79, 111]. State-of-the-art unsupervised disentanglement methods enrich *Gaussian* variational autoencoders [45] with various regularizers encouraging disentangling properties [38, 52, 11, 44, 14]. Locatello et al. [59] showed that unsupervised disentanglement without inductive priors is theoretically impossible. This is why a recent line of work has shifted to weakly-supervised disentanglement [61, 97, 60, 47].

Recall from the last section that we have ground-truth factors that are sampled from some distribution $\boldsymbol{z} \sim p(\boldsymbol{z}) = \prod p(z_i)$. The observation is then a sample from the conditional probability $\boldsymbol{x} \sim p(\boldsymbol{x}|\boldsymbol{z})$. The goal of disentanglement learning is to find a representation $r(\boldsymbol{x})$ such that each ground-truth factor z_i is recovered in one and only one dimension of the representation. The idea of mapping a single dimension of the representation to each ground-truth factor and the idea that one ground-truth factor should influence only one dimension of the representation have been discussed thoroughly by Eastwood and Williams [23].

State-of-the-art unsupervised disentanglement methods assume a Normal prior distribution $p(\boldsymbol{z}) = \mathcal{N}\big(\boldsymbol{0}, \boldsymbol{I}\big)$ as well as an amortized diagonal Gaussian for the approximated posterior distribution $q_\phi(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}\big(\boldsymbol{z} \mid \boldsymbol{\mu}_\phi(\boldsymbol{x}), \boldsymbol{\sigma}_\phi(\boldsymbol{x})\boldsymbol{I}\big)$. They enrich the ELBO with regularizers encouraging disentangling [38, 52, 11, 44, 14] and choose the representation as the mean of the approximated posterior $r(\boldsymbol{x}) = \boldsymbol{\mu}_\phi(\boldsymbol{x})$ [59].

The β-VAE model [38] introduces a hyperparameter to control the trade-off between the reconstruction loss and the KL-divergence term, promoting disentangled latent representations. The annealedVAE [11] adapts to the β-VAE by annealing the β hyperparameter during training. FactorVAE [44] and β-TCVAE [14] promote independence among latent variables by controlling the total correlation between them. DIP-VAE-I and DIP-VAE-II [52] are two variants that enforce disentangled latent factors by matching the covariance of the aggregated posterior to that of the prior.

While regularization and supervision have been discussed extensively in the disentanglement literature, the variational autoencoder is a component that has mainly remained constant. At the same time, multivariate Gaussian distributions suffer from rotational invariance, which can harm disentangling properties [116]. We will show in Section 4.2 that switching to a categorical distribution, with its inherent grid structure, mitigates this problem and acts as an efficient inductive prior for disentangled representations.

2.5.3 Measuring Disentanglement

Many metrics have been proposed to quantify the degree of disentanglement. However, designing such a measure is not straightforward due to the complexity of the problem [59]. In this section, we focus on a commonly used approach to measuring disentanglement, the *Mutual Information Gap* (MIG) [14]. Our exploration of this metric closely follows the work of Locatello et al. [59], who comprehensively explored the challenges in evaluating disentangled representations.

Mutual Information Gap

Chen et al. [14] argued that the previously used metrics for evaluating the disentanglement of representations are neither general nor unbiased due to their reliance on specific hyperparameters. In contrast, they proposed a different metric known as the Mutual Information Gap (MIG), which relies on the concept of mutual information, a measure from information theory that quantifies the information shared between two variables.

The MIG is calculated as follows: First, the mutual information $I(r_j, z_i)$ between each ground truth factor of variation z_i and each dimension of the computed representation $r(x)$ is computed. For each ground truth factor z_i, the dimensions in $r(x)$ with the highest and second highest mutual information are identified. The MIG is then defined as the average difference between these two quantities, normalized by the entropy of the ground truth factor $H(z_i)$.

This metric, therefore, aims to reward representations where each dimension is highly sensitive to exactly one ground truth factor of variation. Specifically, the MIG rewards situations, in which the highest mutual information is significantly higher than the second highest, indicating that each dimension in the latent space corresponds to a distinct factor of variation in the data.

We closely follow the implementation of Locatello et al. [59] to estimate the discrete mutual information. Each dimension of the representations obtained from 10 000 points is binned into 20 bins. The score is computed according to the following formula:

$$\frac{1}{n}\sum_{i=1}^{n}\frac{1}{H(z_i)}\left(I(r_{j_i}, z_i) - \max_{j \neq j_i} I(r_j, z_i)\right), \tag{2.15}$$

where z_i is a factor of variation, r_j is a dimension of the latent representation, and $j_i = \arg\max_j I(r_j, z_i)$. Applying this formula ensures that the disentanglement measure is consistent across different datasets and does not depend on particular hyperparameters.

Chapter 3

Related Work

This chapter provides an overview of the work that has informed this thesis. It covers discrete representations in reinforcement learning and generative models, the development and improvement of gradient estimators for discrete latent variables, the evolution of unsupervised and semi-supervised disentanglement approaches, and a brief insight into Neural Architecture Search (NAS).

In Section 3.1, we discuss the importance of discrete representations in machine learning, highlighting how they have been utilized and enhanced in both generative models and reinforcement learning. Section 3.2 examines gradient estimators for discrete latent variables, presenting various algorithms and methods that allow backpropagation through stochastic nodes. Next, we address the learning of disentangled representations, especially in the context of unsupervised and semi-supervised learning in Section 3.3. We discuss prior works that have attempted to separate underlying factors of variation within data. Lastly, Section 3.4 introduces Neural Architecture Search, providing an overview of the paradigms and benchmarks in this area.

While each section in this chapter provides a robust foundation for the specific area, it is not an exhaustive review of all the related work. Rather, it gives a concise yet comprehensive overview of the key concepts, methods, and contributions relevant to this thesis' scope.

3.1 Discrete Representations in Reinforcement Learning and Generative Models

The importance of discrete representations in the field of machine learning is evident in several recent developments. Notably, Vector Quantised-Variational AutoEncoder (VQ-VAE) introduced by Van Den Oord et al. [110] is a generative model that yields discrete representations and overcomes typical issues observed in VAE framework like posterior collapse [62]. These discrete codes utilize an autoregressive prior for generating high-quality multimedia outputs. Razavi et al. [83] extended this concept by using the VQ-VAE-2 model for large-scale image generation. They showed that using simple feed-forward encoder and decoder networks along with VQ-VAE can generate synthetic samples of high coherence and fidelity, rivaling state-of-the-art Generative Adversarial Networks (GANs) [31].

Moving to reinforcement learning, Hafner et al. [35] utilized discrete world models for better generalization and increased sample efficiency. They introduced DreamerV2, an agent that learns from predictions in the compact latent space of a world model using discrete representations. This model achieved human-level performance on the Atari benchmark, outperforming top single-GPU agents. Ozair et al. [75] used discrete autoencoders to handle stochastic and partially-observable environments in model-based RL. They leveraged discrete latent variables for planning in uncertain conditions, outperforming an offline version of MuZero [94] on a stochastic interpretation of chess. In a later work, Hafner et al. [36] showcased the ability of world models in diverse domains. With their DreamerV3 model, they tackled tasks across continuous and discrete actions, visual and low-dimensional inputs, and 2D and 3D worlds, demonstrating their method's flexibility.

Ramesh et al. [81] used transformers that autoregressively model text and discrete image tokens, showcasing the potential of transformers in generating images from textual descriptions in a zero-shot fashion. Furthermore, Rombach et al. [91] used latent diffusion models (LDMs) for high-resolution image synthesis. By training diffusion models (DMs) [40] in the latent space of pre-trained autoencoders, they maintained the quality of DMs while reducing computational requirements.

Prior work has argued that discrete representations are a natural fit for complex reasoning or planning [41, 81, 75] and has shown empirically that a discrete latent space yields better generalization behavior [35, 91]. Hafner et al. [35] hypothesize that the sparsity enforced by a vector of discrete latent variables could encourage generalization behavior. However, they admit that ”we do not know the reason why the categorical variables are beneficial.” We focus on an extensive study of the structural impact of discrete representations to answer this question.

3.2 Gradient Estimators for Discrete Latent Variables

Various gradient estimators have been proposed for the problem of backpropagating through stochastic nodes. The REINFORCE algorithm [117] utilizes the log-derivative trick. The Straight-Through estimator [8] back-propagates through hard samples by replacing the threshold function with the identity in the backward pass. More recent approaches are based on reparameterization tricks [45] that enable the gradient computation by removing the dependence of the density on the input parameters. Maddison et al. [64] and Jang et al. [41] propose the Gumbel-Softmax trick, a continuous (but biased) reparameterization trick for categorical distributions. Tucker et al. [109] and Grathwohl et al. [29] introduce parameterized control variates to lower the variance for these gradient estimators. They show that these estimators can, in theory, also be generalized to chains of stochastic nodes but do not discuss the learning behavior of such models. Shayer et al. [96] modify the local reparameterization trick [46] to improve the training of discrete gradient estimators. Paulus et al. [77] proposed a framework that generalizes the Gumbel-Softmax trick for various discrete probability distributions.

Pervez et al. [78] showed how to utilize harmonic analysis for boolean functions to control the bias and variance for gradient estimates for boolean latent variable models. SparseMAP [74] is an approach to structured prediction and latent variables, replacing the exponential distribution (specifically, the softmax) with a sparser distribution. LP-SparseMAP [73] is a continuation of SparseMAP using a relaxation of the optimization problem rather than a MAP oracle. In addition, the idea to enforce more sparsity can also be exploited for efficient marginal inference in latent variable models [16], which can then be used to compute stochastic gradients.

We aim not to improve gradient estimators for a single stochastic node but to analyze the behavior of discrete-continuous computation graphs with multiple sequential discrete components. During this exploration, we show both analytically and empirically that it is challenging to optimize the parameters of these models, mainly due to insufficient gradient signals often caused by local minima and saturation. We then propose two new methods for mitigating the causes of poor optimization behavior.

3.3 Unsupervised and Semi-supervised Disentanglement

State-of-the-art unsupervised disentanglement methods enhance Gaussian VAEs with various regularizers that encourage disentangling properties. The β-VAE model [38] introduces a hyperparameter to control the trade-off between the reconstruction loss and the KL-divergence term, promoting disentangled latent representations. The AnnealedVAE [11] adapts the β-VAE by annealing the β hyperparameter during training. FactorVAE [44] and β-TCVAE [14] promote independence among latent variables by controlling the total correlation between them. DIP-VAE-I and DIP-VAE-II [52] are two variants that enforce disentangled latent factors by matching the covariance of the aggregated posterior to that of the prior.

In a critical review of these methods, Locatello et al. [59] argue that the unsupervised learning of disentangled representations is fundamentally impossible without inductive biases on both the models and the data. While different methods can enforce disentangling properties encouraged by their corresponding losse, several studies have started to approach disentanglement learning from a semi-supervised perspective. The Deep Convolutional Inverse Graphics Network (DC-IGN) [51] learns an interpretable representation of images that is disentangled with respect to three-dimensional scene structure and viewing transformations. The model utilizes the Stochastic Gradient Variational Bayes (SGVB) algorithm to train its multilayered convolutional and deconvolutional architecture. Siddharth et al. [98] propose to learn disentangled representations using model architectures that generalize from standard VAEs, employing a general graphical model structure in the encoder and decoder. This approach allows for the training of models that make strong assumptions about a subset of interpretable variables and rely on the flexibility of neural networks to learn representations for the remaining variables. Furthermore, two recent studies explored weakly-supervised disentanglement learning [61, 60]. They demonstrated that it is possible to reliably learn disentangled representations with little and potentially imprecise supervision.

In the context of disentangling with discrete factors, Makhzani et al. [65] propose adversarial autoencoders (AAE) for variational inference by matching the aggregated posterior of the hidden code vector of the autoencoder with an arbitrary prior distribution. This setup ensures meaningful samples across the entire prior space. Dupont [22] introduce a framework for learning disentangled and interpretable jointly continuous and discrete representations in an unsupervised manner, enabling the discovery of continuous and categorical factors of variation. Finally, Jeong and Song [42] propose an alternating minimization problem for unsupervised disentanglement of discrete and continuous explanatory factors of data.

Instead of employing Gaussian VAEs, we propose using a categorical one. In our approach, we treat every ground-truth factor not as a continuous variable but as a discrete representation. We show that the grid structure of the categorical VAE depicts an inductive prior that encourages disentanglement.

3.4 Core Concepts of Neural Architecture Search and its Benchmarks

Neural Architecture Search (NAS) [24] is the process of designing neural network architectures in an automatic way. The currently most successful approaches follow different paradigms: Reinforcement learning (RL) [127, 128, 80] considers the neural architecture generation as the agent's action with its reward given in terms of validation accuracy. Evolutionary Algorithm (EA) [85, 56] approaches optimizing the neural architectures themselves by guiding the mutation of architectures and evaluating their fitness given in terms of validation accuracy. Bayesian optimization (BO) [43] derives kernels for architecture similarity measurements to extrapolate the search space. Gradient-based methods [57, 63] use continuous relaxations of neural architectures to allow for gradient-based optimization.

NAS-Bench-101 [123] is a public dataset of $\sim 423\,000$ neural architectures and provides tabular benchmark results for a restricted cell structured architecture search space [127] with exhaustive evaluation on the CIFAR-10 image classification dataset [50]. As shown in Zela et al. [125], only subspaces of the architectures in NAS-Bench-101 can be used to evaluate one-shot NAS methods [57, 80], motivating their proposed variant NAS-Bench-1shot1 [125]. Similarly to NAS-Bench-101, NAS-Bench-201 [20] uses a restricted, cell-structured search space, while the employed graph representation allows to evaluate discrete and one-shot NAS algorithms. The search space is more restricted than NAS-Bench-101, providing only $\sim 6\,000$ unique evaluated architectures. We conduct our experiments on NAS-Bench-101, the largest available tabular neural architecture benchmark for computer vision problems.

Chapter 4

Theoretical Analysis

This chapter presents our theoretical analysis of discrete representations within Stochastic Computation Graphs (SCGs). We explore both the challenges and potential benefits of using discrete representations.

In Section 4.1, we tackle the difficulties of training discrete representations in SCGs focusing on problems posed by the Gumbel-softmax trick. We discuss how the distribution categories are not fully utilized during training. A key finding of this section is the discussion of a constraint on the gradient of a parameter of the Gumbel-softmax distribution.

Section 4.2 focuses on the benefits of using discrete representations. We analyze the impact of these representations on the latent space and how they can enhance disentanglement. We discuss the issue of rotational equivariance of Gaussian Variational Autoencoders (VAEs) and suggest a variant of the discrete variational autoencoder (D-VAE) as a potential solution.

Hence, this chapter provides a theoretical understanding of using discrete representations in SCGs. It sets the groundwork for developing improved methods discussed in Chapter 5.

4.1 Challenges in Training Discrete Stochastic Computation Graphs

This section introduces the challenges in the training process when dealing with discrete representations. It highlights the potential problems associated with using discrete representations in the context of Stochastic computation graphs (SCGs).

This discussion addresses our research question **RQ1.1:** *Why is the training of models based on discrete SCGs inherently challenging?*

Stochastic computation graphs, which extend neural networks by incorporating stochastic operations, complicate the training process by introducing the need to compute gradients with respect to the parameters of a distribution [95]. This challenge becomes even more evident when training SCGs that involve stochastic nodes based on categorical (discrete) distributions.

Here, our analysis focuses on the behavior of the Gumbel-softmax trick in complex SCGs, involving sequential discrete components on a single execution path. We discuss the tendency of such models to not fully utilize all existing categories during training, and to map different input representations to the same category.

A significant result in our understanding of this issue is presented in Theorem 1. The theorem outlines a constraint on the gradient of a parameter of the Gumbel-softmax distribution by the normalized probability of a category. This observation helps explain the issues related to gradient optimization, where small gradient values can lead to suboptimal training outcomes.

The content of this section primarily draws upon the following publication:

> Friede, D., & Niepert, M. (2021). *Efficient Learning of Discrete-Continuous Computation Graphs*. Advances in Neural Information Processing Systems, 34, 6720-6732.

4.1.1 Background

Neuro-symbolic learning systems aim to combine discrete and continuous operations. The majority of recent research has focused on integrating neural network components into probabilistic logics [90, 66], that is, making logic-based reasoning approaches more amenable to noisy and high-dimensional input data. On the other end of the spectrum are the data-driven approaches, which aim to learn a modular and discrete program structure in an end-to-end neural network based system. The broader vision followed by proponents of these approaches are systems capable of assembling the required modular operations to solve a variety of tasks with heterogeneous input data. Reinforcement learning [102], neuro-symbolic program synthesis [76], and neural module networks [2] are instances of such data-integrated discrete-continuous learning systems.

We focus on learning systems comprised of both symbolic and continuous operations where the symbolic operations are modeled as discrete probability distributions. The resulting systems can be described by their stochastic computation graphs (SCGs), as introduced in Section 2.1, a formalism recently introduced to unify several related approaches [95]. Figure 2.1, Figure 2.3, and Figure 2.4 illustrate three instances of such stochastic computation graphs. For a given high-dimensional input such as a set of images or a list of symbols, a computation graph, consisting of stochastic (discrete) and continuous (neural) nodes, is created to solve a particular task. Both, the mechanism to assemble the graphs (if not already provided) and the various operations are learned end-to-end. Discrete nodes in the computation graph are Gibbs distributions modeled at a particular temperature which can also be used to model the $\arg\max$ operation at zero temperature.

The majority of prior work has focused on graphs with a single discrete probability distribution along each execution path. Examples are the discrete variational autoencoder [41], learning to explain [13], and other applications of stochastic softmax tricks [77]. We aim to analyze the training behavior of *complex* computation graphs, that is, graphs with more than one discrete probability distribution in its execution paths. More concretely, we focus on computation graphs where the stochastic nodes are categorical random variables modeled with the Gumbel-softmax trick [41, 64], as introduced in Section 2.3. In this section, we demonstrate analytically that optimizing the parameters of these models poses significant challenges, primarily due to inadequate gradient signals, which are often instigated by local minima and saturation.

4.1.2 Challenges in Training Stochastic Compuation Graphs

Standard neural networks compose basic differentiable functions. These networks, therefore, can be fully described by a directed acyclic graph (the computation graph) that determines the operations executed during the forward and backward passes. Schulman et al. [95] proposed stochastic computation graphs (SCGs) as an extension of neural networks that combine deterministic and stochastic operations – a node in the computation graph can be either a differentiable function or a probability distribution. For a more detailed overview of SCGs and their derivations, we refer the reader to Sections 2.1 and 2.2. In this section, however, we will only provide a brief summary. A stochastic node z is typically a random variable with a parameterized probability distribution $p_{\boldsymbol{\theta}(z)}$ with parameters $\boldsymbol{\theta}$. Suppose that f is a smooth function (such as the loss function of a learning problem), then the gradient of f at the stochastic node is $\nabla_{\boldsymbol{\theta}}\mathbb{E}_{p_{\boldsymbol{\theta}}(z)}[f(z)]$. Kingma and Welling [45] proposed the reparameterization trick, as introduced in Section 2.2.2, to overcome the problem of computing gradients with respect to the parameters of a distribution The idea is to find a function g and distribution ρ such that one can replace $z \sim p_{\boldsymbol{\theta}}(z)$

by $z = g(\epsilon, \boldsymbol{\theta})$ with $\epsilon \sim \rho(\epsilon)$. If this is possible, then one can write

$$\nabla_{\boldsymbol{\theta}} \mathbb{E}_{z \sim p_{\boldsymbol{\theta}}(z)}[f(z)] = \mathbb{E}_{\epsilon \sim \rho(\epsilon)}[\nabla_{\boldsymbol{\theta}} f(g(\epsilon, \boldsymbol{\theta}))] \tag{4.1a}$$

$$\approx \frac{1}{S} \sum_{i=1}^{S} \nabla_{\boldsymbol{\theta}} f(g(\epsilon^i, \boldsymbol{\theta})) \text{ with } \epsilon^i \sim \rho(z). \tag{4.1b}$$

In the following, we address the problem of training stochastic computation graphs where the stochastic nodes are based on categorical (discrete) distributions approximated using stochastic softmax tricks [41, 64, 77]. More specifically, we consider discrete-continuous components modeling a categorical variable X with k possible values $\boldsymbol{z}_1, ..., \boldsymbol{z}_k$. Each of the $\boldsymbol{z}_i$ is the one-hot encoding of the category i. We consider discrete-continuous functions[1] $f : \mathbb{R}^n \to \mathbb{R}^n$ with $\boldsymbol{v} = f(\boldsymbol{u})$ defined as follows

$$\boldsymbol{\theta} = g_{\boldsymbol{w}}(\boldsymbol{u}) \tag{4.2a}$$

$$p(\boldsymbol{z}_i; \boldsymbol{\theta}) = \frac{\exp(\theta_i)}{\sum_{j=1}^{k} \exp(\theta_j)} \tag{4.2b}$$

$$\boldsymbol{z} \sim p(\boldsymbol{z}; \boldsymbol{\theta}) \tag{4.2c}$$

$$\boldsymbol{v} = h_{\boldsymbol{w}'}(\boldsymbol{z}) \tag{4.2d}$$

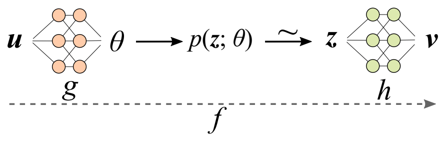


Figure 4.1: Illustration of a generic discrete-continuous component.

Note that, compared to Equation (2.8), here, we treat $\boldsymbol{\theta}$ as logits. We assume that the functions g and h (parameterized by $\boldsymbol{w}$ and $\boldsymbol{w}'$) are expressed using differentiable neural network components. In the majority of cases, h is defined as $\boldsymbol{z}^\intercal \mathbf{w}'$ for a learnable matrix $\mathbf{w}'$, mapping object z_i to its learned vector representation. Figure 4.1 illustrates a generic discrete-continuous component.

In the following, we give a short summary of the reparameterizations using Gumbel variables as introduced in Section 2.3. Let $\text{Gumbel}(0, \beta)$ be the Gumbel distribution with location 0 and scale β. Using the Gumbel-max trick, we can sample $\boldsymbol{z} \sim p(\boldsymbol{z}; \boldsymbol{\theta})$ as follows:

$$\boldsymbol{z} = \boldsymbol{e}^i \quad \text{with} \quad i = \underset{j \in \{1,...,m\}}{\arg\max} \, [\theta_j + \epsilon_j] \text{ where } \epsilon_j \sim \text{Gumbel}(0, 1), \tag{4.3}$$

where $e^i_j = 1$, if $i = j$ and $e^i_j = 0$, otherwise. The Gumbel-softmax trick is a relaxation of the Gumbel-max trick (relaxing the argmax into a softmax with a scaling parameter τ) that allows one to perform standard backpropagation:

$$z_i = \frac{\exp\left((\theta_i + \epsilon_i)/\tau\right)}{\sum_{j=1}^{k} \exp\left((\theta_j + \epsilon_j)/\tau\right)} \text{ where } \epsilon_i \sim \text{Gumbel}(0, 1). \tag{4.4}$$

[1]For the sake of simplicity we assume the same input and output dimensions.

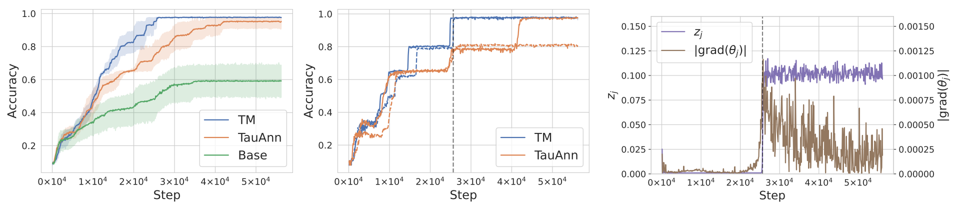


Figure 4.2: A comparison of the learning behavior of different annealing schemes for the MNIST addition task, averaged over 8 trained models. **Left:** The base model (Base), with constant parameters $\tau = 8$ and $\beta = 1$, cannot achieve an accuracy of more than 0.7. Increasing β during training (TM) achieves better results than annealing τ (TauAnn). **Middle:** Two training runs for TM and TauAnn, respectively, where an accuracy of 0.9 was reached the latest. The accuracy curves of TM jump to the next plateau earlier and more consistently. The dotted grey line depicts the moment when a run of TM achieves an accuracy of ~ 1.0. **Right:** For the same run, z_j and the absolute value of $\partial L / \partial \theta_j$ are plotted. The downstream gradient signal for θ_j is largely neutralized by a small value of z_j until the corresponding category is used and its average probability abruptly reaches 0.1.

Hence, instead of sampling a discrete $\boldsymbol{z}$, the Gumbel-Softmax trick computes a relaxed $\boldsymbol{z}$ in its place.

We are concerned with the analysis of the behavior of the Gumbel-softmax trick in more complex stochastic computation graphs. In these computation graphs, multiple sequential Gumbel-softmax components occur on a single execution path. Throughout the remainder of this thesis, we assume that during training, we use the Softmax-trick as outlined above, while at test time, we sample discretely using the Gumbel-max trick. Depending on the use case, we also sometimes compute the argmax at test time instead of samples from the distribution.

Let us first take a look at $\partial \boldsymbol{v} / \partial \boldsymbol{u}$, that is, the partial derivative of the output of the discrete-continuous function f with respect to its input. Using the chain rule (and with a slight abuse of notation to simplify the presentation), we can write $\partial \boldsymbol{v} / \partial \boldsymbol{u} = (\partial \boldsymbol{v} / \partial \boldsymbol{z})(\partial \boldsymbol{z} / \partial \boldsymbol{\theta})(\partial \boldsymbol{\theta} / \partial \boldsymbol{u})$. By assumption $\partial \boldsymbol{v} / \partial \boldsymbol{z}$ and $\partial \boldsymbol{\theta} / \partial \boldsymbol{u}$ exist and are efficiently computable. As mentioned in Section 2.4, we have

$$\frac{\partial z_i}{\partial \theta_j} = \begin{cases} \frac{1}{\tau}\, z_i(1 - z_i) & \text{if } i = j \\ -\frac{1}{\tau}\, z_i z_j & \text{if } i \neq j. \end{cases} \tag{4.5}$$

We have empirically observed that during training, there is a tendency of the models to not utilize all existing categories of the distribution and to wrongfully map different input representations to the same category. For instance, for the MNIST addition problem, all encoded images of two digits are mapped to the same category of the distribution. In these cases, the gradients of the parameters of the unused categories are vanishingly small. In order to analyze this behavior more closely, we derive an upper bound on the gradient of a parameter θ_j of the categorical distribution with respect to a loss function L.

We believe that the core of the problem can be attributed to a unique property of the Jacobian of the softmax function. The gradient of a parameter θ_j of the Gumbel-softmax distribution is constrained by the normalized probability z_j. To be more precise, we propose the following theorem.

Theorem 1. *Let $\boldsymbol{z}$ be a random variable following the Gumbel-Softmax distribution, i.e., $\boldsymbol{z} \sim \mathrm{GS}(\boldsymbol{\theta}, \tau)$ with unnormalized category probabilities $\boldsymbol{\theta} \in (0, \infty)^m$ and temperature $\tau > 0$. If $\left\|\frac{\partial L}{\partial \boldsymbol{z}}\right\|_F < C'$ for some constant $C' \in \mathbb{R}$, then a constant $C \in \mathbb{R}$ exists for all $j = 1, \dots, m$ such that*

$$|\mathrm{grad}(\theta_j)| := \left|\frac{\partial L}{\partial \theta_j}\right| \leq C z_j. \tag{4.6}$$

Here, $\|\cdot\|_F$ is the Frobenius norm. If the condition $\left\|\frac{\partial L}{\partial \boldsymbol{z}}\right\|_F < C'$ is met, the following issue arises: a small value of θ_j (and consequently z_j) leads to a correspondingly small gradient for θ_j. As a consequence, such models are more prone to fall into poor minima during training. Figure 4.2 illustrates an example of such a situation, which we encountered in practice. The small value of a specific z_j leads to a small gradient at θ_j (right) as well as to suboptimal plateauing of the accuracy curve (middle). In other words, the downstream gradients information for θ_j is neutralized by a small z_j.

To provide an intuition for the condition $\left\|\frac{\partial L}{\partial \boldsymbol{z}}\right\|_F < C'$, we examine two common use cases. As previously mentioned, in most scenarios, the function h from Equation (4.2d) is defined as $\boldsymbol{z}^\intercal \mathbf{w}'$ for a learnable matrix $\mathbf{w}'$, mapping the object z_i to its learned vector representation. Consequently, $(\partial h / \partial \boldsymbol{z}) = \mathbf{w}'^\intercal$ is independent of $\boldsymbol{z}$. Therefore, $(\partial L / \partial \boldsymbol{z}) = (\partial L / \partial h)(\partial h / \partial \boldsymbol{z})$ is also independent of $\boldsymbol{z}$. This independence is important as we are particularly interested in the case where z_j is small, i.e., $z_j \to 0$ for some $j = 1, \dots, m$. Thus, in most cases, the condition $\left\|\frac{\partial L}{\partial \boldsymbol{z}}\right\|_F < C'$ is fulfilled, regardless of the behavior of $\boldsymbol{z}$.

In contrast, the condition does not hold for the cross-entropy (CE) loss function, which presents a different use case. The unique behavior of the CE loss counteracts the dependence on $\boldsymbol{z}$ of the derivative of the softmax function. The CE loss is defined as

$$L_{\boldsymbol{y}}(\boldsymbol{z}) = -\sum_{j=1}^{m} y_j \log z_j, \tag{4.7}$$

for a one-hot encoded label vector $\boldsymbol{y}$. The CE loss yields a large value in the event of a small z_j, if $y_j = 1$. Specifically, we have

$$\frac{\partial L_{\boldsymbol{y}}}{\partial \boldsymbol{z}} = -\frac{\boldsymbol{y}}{\boldsymbol{z}}, \tag{4.8}$$

demonstrating the dependence on $\boldsymbol{z}$. Consequently, when z_j is small, i.e., $z_j \to 0$ for some $j = 1, \dots, m$, the condition $\left\| \frac{\partial L}{\partial \boldsymbol{z}} \right\|_F < C'$ is not met.

In the following, we aim to prove Theorem 1. Initially, we seek a sub-multiplicative matrix norm that bounds the Jacobian of the softmax function from Equation (4.5) by $\boldsymbol{z}$. More precisely, we propose the following Lemma.

Lemma 1. *Let $\boldsymbol{z}$ be a random variable following the Gumbel-Softmax distribution, i.e., $\boldsymbol{z} \sim \mathrm{GS}(\boldsymbol{\theta}, \tau)$ with unnormalized category probabilities $\boldsymbol{\theta} \in (0, \infty)^m$ and temperature $\tau > 0$. Let $\|\cdot\|_F$ be the Frobenius norm. Then, for each $j = 1, \dots, m$, it holds that*

$$\left\| \frac{\partial \boldsymbol{z}}{\partial \theta_j} \right\|_F \leq \frac{\sqrt{2}}{\tau} z_j. \tag{4.9}$$

Proof. Let us first take a look at the Jacobian of the softmax function. We have

$$\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{\theta}} = \frac{1}{\tau} \begin{bmatrix} z_1(1-z_1) & -z_1 z_2 & -z_1 z_3 & \dots & -z_1 z_m \\ -z_2 z_1 & z_2(1-z_2) & -z_2 z_3 & \dots & -z_2 z_m \\ \vdots & & \ddots & & \vdots \\ -z_m z_1 & -z_m z_2 & \dots & -z_m z_{m-1} & z_m(1-z_m) \end{bmatrix}.$$

Therefore, for a fixed but arbitrary $j = 1, \dots, m$, we have

$$\frac{\partial \boldsymbol{z}}{\partial \theta_j} = \frac{1}{\tau} \begin{bmatrix} -z_1 z_j & \cdots - z_{j-1} z_j & z_j(1-z_j) & -z_{j+1} z_j & \dots & -z_m z_j \end{bmatrix}^\intercal.$$

The Frobenius Norm of a matrix $A \in \mathbb{K}^{m \times n}$ with matrix elements a_{ij} is defined as $\|A\|_F = \sqrt{\sum_i^n \sum_j^m |a_{ij}|^2}$. For clarity, we consider the squared Frobenius norm of $\frac{\partial \boldsymbol{z}}{\partial \theta_j}$. Simple calculus yields

$$\begin{aligned}
\left\| \frac{\partial \boldsymbol{z}}{\partial \theta_j} \right\|_F^2 &= \left\| \left(\frac{\partial z_1}{\partial \theta_j} \cdots \frac{\partial z_m}{\partial \theta_j} \right)^\intercal \right\|_F^2 \\
&= \sum_{i=1}^{m} \left| \frac{\partial z_i}{\partial \theta_j} \right|^2 \\
&= \left| \frac{1}{\tau} z_j (1 - z_j) \right|^2 + \sum_{i \neq j} \left| \frac{1}{\tau} z_i z_j \right|^2 \\
&= \frac{1}{\tau^2} z_j^2 (1 - z_j)^2 + \sum_{i \neq j} \frac{1}{\tau^2} z_i^2 z_j^2 \\
&= \frac{1}{\tau^2} \big(z_j^2 (1 - z_j)^2 + \sum_{i \neq j} z_i^2 z_j^2 \big) \\
&= \frac{1}{\tau^2} z_j^2 \big((1 - z_j)^2 + \sum_{i \neq j} z_i^2 \big).
\end{aligned}$$

For all $i = 1, \ldots, m$, we have $z_i \in [0, 1]$ and $\sum_{i=1}^{m} z_i = 1$. Therefore, we have $\sum_{i \neq j} z_i^2 \leq \sum_{i \neq j} z_i \leq 1$ and $(1 - z_j)^2 \leq 1$ and thus,

$$\begin{aligned}
\frac{1}{\tau^2} z_j^2 \big((1 - z_j)^2 + \sum_{i \neq j} z_i^2 \big) &\leq \frac{1}{\tau^2} z_j^2 \left((1 - z_j)^2 + 1 \right) \\
&\leq \frac{1}{\tau^2} z_j^2 (1 + 1) \\
&= \frac{2}{\tau^2} z_j^2.
\end{aligned}$$

Taking the square root leads us to the conclusion that $\left\| \frac{\partial \boldsymbol{z}}{\partial \theta_j} \right\|_F \leq \frac{\sqrt{2}}{\tau} z_j$. □

We can now utilize the fact that the Frobenius norm is sub-multiplicative to derive a similar upper bound for $\left| \frac{\partial L}{\partial \theta_j} \right|$, as depicted in the following lemma.

Lemma 2. *Let $\boldsymbol{z}$ be a random variable following the Gumbel-Softmax distribution, i.e., $\boldsymbol{z} \sim \mathrm{GS}(\boldsymbol{\theta}, \tau)$ with unnormalized category probabilities $\boldsymbol{\theta} \in (0, \infty)^m$ and temperature $\tau > 0$. Let $\|\cdot\|_F$ be the Frobenius norm. Then, for each $j = 1, \ldots, m$, it holds that*

$$\left| \frac{\partial L}{\partial \theta_j} \right| \leq \frac{\sqrt{2}}{\tau} \left\| \frac{\partial L}{\partial \boldsymbol{z}} \right\|_F z_j. \tag{4.10}$$

Proof.

$$\left|\frac{\partial L}{\partial \theta_j}\right| = \left\|\frac{\partial L}{\partial \theta_j}\right\|_F = \left\|\frac{\partial L}{\partial \boldsymbol{z}}\frac{\partial \boldsymbol{z}}{\partial \theta_j}\right\|_F \leq \left\|\frac{\partial L}{\partial \boldsymbol{z}}\right\|_F \left\|\frac{\partial \boldsymbol{z}}{\partial \theta_j}\right\|_F \leq \frac{\sqrt{2}}{\tau}\left\|\frac{\partial L}{\partial \boldsymbol{z}}\right\|_F z_j.$$

The first inequality follows from the sub-multiplicativity of the Frobenius norm, while the second inequality is the result of Lemma 1. □

Theorem 1 is now a direct consequence of Lemma 2.

Proof of Theorem 1. If we choose $C = \frac{\sqrt{2}}{\tau}C'$, Lemma 2 yields

$$|\text{grad}(\theta_j)| = \left|\frac{\partial L}{\partial \theta_j}\right| \leq \frac{\sqrt{2}}{\tau}\left\|\frac{\partial L}{\partial \boldsymbol{z}}\right\|_F z_j \leq C z_j.$$

The second inequality stems from the condition $\|\frac{\partial L}{\partial \boldsymbol{z}}\|_F < C'$. □

As discussed before, a small value of θ_j (and consequently z_j) leads to a small gradient for θ_j, which let such models to be more prone to fall into poor minima during training. A related problem is the saturation of probabilities in sequentially connected probabilistic components. This issue parallels the problem of sigmoid activation units in deep neural networks, which have been replaced by ReLUs and other modern activation functions due to their tendency to cause vanishing gradients. Indeed, in several of our experiments, we observed that the Gumbel-softmax distributions saturate and, consequently, insufficient gradient information reaches the parameters of upstream neural network components.

In Section 5.2, we will introduce two strategies to mitigate the vanishing gradient behavior in complex discrete-continuous computation graphs. Firstly, we analyze the interplay between the temperature parameter τ and the scale parameter β of the Gumbel distribution. We discuss the subtle difference between these parameters, which may also be of interest for improving stochastic softmax tricks [77]. By increasing β relative to τ while keeping τ fixed, we enhance the probability of a gradient flow in the event of saturation. Our research indicates that the larger β becomes, the more uniformly distributed (across categories) the discrete samples from the distribution become, thereby increasing the likelihood of escaping poor minima. Secondly, we introduce dropout residual connections. These allow us to establish a lower bound for the gradients in expectation, leading to a more direct gradient flow to parameters of upstream model components.

4.1.3 Conclusion

This section has offered a detailed exploration of the challenges posed by training stochastic computation graphs (SCGs) with discrete representations, answering our research question **RQ1.1**. The issues become particularly apparent when dealing with complex SCGs with sequential discrete components. We highlighted the central concern, which is the tendency of models to underutilize the available categories, often causing multiple distinct input representations to be mapped to a single category.

Our main contribution to this discourse has been the derivation of Theorem 1, which reveals a constraint on the gradient of a Gumbel-softmax parameter by the normalized probability of a category. This observation sheds new light on the difficulties encountered during gradient optimization, where small gradient values can give rise to suboptimal training results.

The insights obtained from this discussion are fundamental for creating novel methods that efficiently train discrete representations in stochastic computation graphs. We will present our contribution to such methods in Section 5.2.

4.2 Understanding the Benefits of Discrete Representations

This section explores the structural benefits of discrete representations, particularly focusing on the impact on disentanglement when substituting the standard variational autoencoder with a slightly modified categorical variational autoencoder. This analysis addresses our research question **RQ1.2:** *Why do discrete representations possess structural advantages over their continuous counterparts?*

Prior works in large-scale image generation and model-based reinforcement learning have leveraged discrete variational autoencoders and demonstrated better generalization behavior [83, 35, 81, 91, 36]. Despite the empirical evidence, the exact reasons behind the advantages of using discrete latent variables remain unclear [35]. We aim to shed light on this by examining the structural effects of discrete representations on the latent space.

The primary result of this section is Theorem 2, which highlights an issue regarding the rotational equivariance of Gaussian latent spaces. When the latent space is represented by a Gaussian distribution, rotations within this space do not change the structure of the distribution itself. However, rotations pose a significant challenge to disentanglement, potentially entangling the generative factors that we aim to separate.

We propose a discrete variational autoencoder variant (D-VAE) that models a joint distribution of Gumbel-softmax random variables, with each category represented by a one-dimensional entity. This structure creates a discrete grid, potentially mitigating the rotational problem in the latent space.

Understanding the structural impacts of discrete representations on the latent space is vital for answering our research question **RQ1.2**. This understanding could lead to the development of more effective and efficient models in the future. We will present such initial methods in Section 5.3. The remainder of this section delves into a detailed explanation of our discrete variational autoencoder and presents a comprehensive analysis of the rotational equivariance issue.

The content of this section primarily draws upon the following publication:

> Friede, D., Reimers, C., Stuckenschmidt, H., & Niepert, M. (2023). *Learning Disentangled Discrete Representations*. Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD

4.2.1 Background

Discrete variational autoencoders based on categorical distributions [41, 64] or vector quantization [110] have enabled recent success in large-scale image generation [110, 83], model-based reinforcement learning [35, 75, 36], and perhaps most notably, in text-to-image generation models like Dall-E [81] and Stable Diffusion [91]. Prior work has argued that discrete representations are a natural fit for complex reasoning or planning [41, 81, 75] and has shown empirically that a discrete latent space yields better generalization behavior [35, 91]. Hafner et al. [35] hypothesize that the sparsity enforced by a vector of discrete latent variables could encourage generalization behavior. However, they admit that ”we do not know the reason why the categorical variables are beneficial.”

We focus on an extensive study of the *structural impact* of discrete representations on the latent space. The disentanglement literature [7, 39, 59] provides a common approach to analyzing the structure of latent spaces. Disentangled representations [7] recover the low-dimensional and independent ground-truth factors of variation of high-dimensional observations. Such representations promise interpretability [39, 1], fairness [58, 17, 106], and better sample complexity for learning [93, 7, 79, 111]. State-of-the-art unsupervised disentanglement methods enrich *Gaussian* variational autoencoders [45] with regularizers encouraging disentangling properties [38, 52, 11, 44, 14]. Locatello et al. [59] showed that unsupervised disentanglement without inductive priors is theoretically impossible. Thus, a recent line of work has shifted to weakly-supervised disentanglement [61, 97, 60, 47].

We focus on the impact on disentanglement of replacing the standard variational autoencoder with a slightly tailored *categorical* variational autoencoder [41, 64]. Most disentanglement metrics assume an ordered latent space, which can be traversed and visualized by fixing all but one latent variable [38, 14, 23]. Conventional categorical variational autoencoders lack sortability since there is generally no order between the categories. In order to enable a direct comparison through the established disentanglement metrics, we modify the categorical variational autoencoder to represent each category with a *one-dimensional* representation. While regularization and supervision have been discussed extensively in the disentanglement literature, the variational autoencoder is a component that has mainly remained constant. At the same time, Watters et. al [116] have observed that Gaussian VAEs might suffer from rotations in the latent space, which can harm disentangling properties. In this section, we analyze the rotational equivariance of multivariate Gaussian distributions in more detail and show that the underlying grid structure of categorical distributions mitigates this problem.

4.2.2 The D-VAE

In the following, we recall the definition of disentanglement and the evidence lower bound (ELBO) from Section 2.5. The disentanglement literature is usually premised on the assumption that a high-dimensional observation $\boldsymbol{x}$ from the data space $\mathcal{X}$ is generated from a low-dimensional latent variable $\boldsymbol{z}$ whose entries correspond to the dataset's ground-truth factors of variation such as position, color, or shape [7, 108]. First, the *independent* ground-truth factors are sampled from some distribution $\boldsymbol{z} \sim p(\boldsymbol{z}) = \prod p(z_i)$. The observation is then a sample from the conditional probability $\boldsymbol{x} \sim p(\boldsymbol{x}|\boldsymbol{z})$. The goal of disentanglement learning is to find a representation $r(\boldsymbol{x})$ such that each ground-truth factor z_i is recovered in one and only one dimension of the representation. The formalism of variational autoencoders [45] enables an estimation of these distributions. Assuming a known prior $p(\boldsymbol{z})$, we can depict the conditional probability $p_\psi(\boldsymbol{x}|\boldsymbol{z})$ as a parameterized probabilistic decoder. In general, the posterior $p_\psi(\boldsymbol{z}|\boldsymbol{x})$ is intractable. Thus, we turn to variational inference and approximate the posterior by a parameterized probabilistic encoder $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ and minimize the Kullback-Leibler (KL) divergence $D_{\mathrm{KL}}\big(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \parallel p_\psi(\boldsymbol{z}|\boldsymbol{x})\big)$. This term, too, is intractable but can be minimized by maximizing the evidence lower bound (ELBO)

$$\mathcal{L}_{\psi,\phi}(\boldsymbol{x}) = \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_\psi(\boldsymbol{x}|\boldsymbol{z})\right] - D_{\mathrm{KL}}\big(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \parallel p(\boldsymbol{z})\big). \tag{4.11}$$

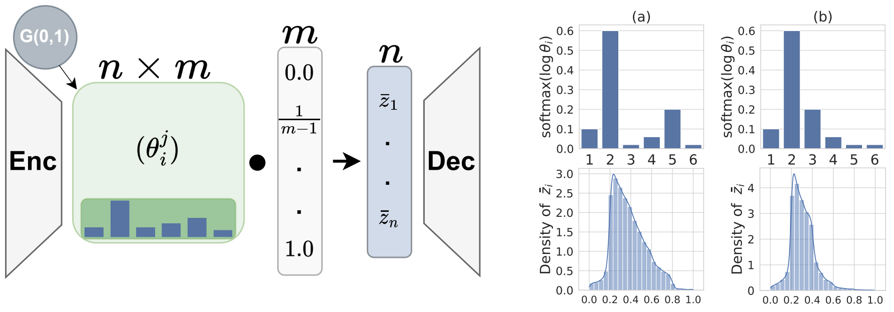


Figure 4.3: We utilize n Gumbel-softmax distributions (GS) to approximate the posterior distribution. **Left:** An encoder learns nm parameters θ_i^j for the n joint distributions. Each m-dimensional sample $\boldsymbol{z}_i \sim \mathrm{GS}(\boldsymbol{\theta}_i)$ is mapped into the one-dimensional unit interval following Equation (4.13). **Right:** Two examples of (normalized) parameters of a single Gumbel-softmax distribution and the corresponding one-dimensional distribution of $\bar{z}_i$.

State-of-the-art unsupervised disentanglement methods assume a Normal prior distribution $p(\boldsymbol{z}) = \mathcal{N}\big(\boldsymbol{0}, \boldsymbol{I}\big)$ and an amortized diagonal Gaussian for the approximated posterior distribution $q_\phi(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}\big(\boldsymbol{z} \mid \boldsymbol{\mu}_\phi(\boldsymbol{x}), \boldsymbol{\sigma}_\phi(\boldsymbol{x})\boldsymbol{I}\big)$. They enrich the ELBO with regularizers encouraging disentangling [38, 52, 11, 44, 14] and choose the representation as the mean of the approximated posterior $r(\boldsymbol{x}) = \boldsymbol{\mu}_\phi(\boldsymbol{x})$ [59].

The Discrete VAE (D-VAE)

We propose a variant of the categorical VAE modeling a joint distribution of n *Gumbel-Softmax* random variables [41, 64]. Let n be the dimension of $\boldsymbol{z}$, m be the number of categories, $\theta_i^j \in (0, \infty)$ be the unnormalized probabilities of the categories and $g_i^j \sim \mathrm{Gumbel}(0, 1)$ be i.i.d. samples drawn from the Gumbel distribution for $i \in [n], j \in [m]$. For each dimension $i \in [n]$, we sample a Gumbel-softmax random variable $\boldsymbol{z}_i \sim \mathrm{GS}(\boldsymbol{\theta}_i)$, as introduced in Section 2.3, over the simplex $\Delta^{m-1} = \{\boldsymbol{y} \in \mathbb{R}^n \mid y^j \in [0, 1], \sum_{j=1}^m y^j = 1\}$ by setting

$$z_i^j = \frac{\exp(\log \theta_i^j + g_i^j)}{\sum_{k=1}^m \exp(\log \theta_i^k + g_i^k)} \tag{4.12}$$

for $j \in [m]$. Here, we define $\mathrm{GS}(\boldsymbol{\theta})$ as $\mathrm{GS}(\boldsymbol{\theta}), 1$ with a constant temperature value of 1. We set the approximated posterior distribution to be a joint distribution of n Gumbel-softmax distributions, i.e., $q_\phi(\boldsymbol{z}|\boldsymbol{x}) = \mathrm{GS}^n\big(\boldsymbol{z} \mid \boldsymbol{\theta}_\phi(\boldsymbol{x})\big)$ and assume a joint discrete uniform prior distribution $p(\boldsymbol{z}) = \mathcal{U}^n\{1, m\}$. Note that $\boldsymbol{z}$ is of dimension

$n \times m$. To obtain the final n-dimensional latent variable $\bar{\boldsymbol{z}}$, we define a function $f : \Delta^{m-1} \to [0,1]$ as the dot product of $\boldsymbol{z}_i$ with the vector $\boldsymbol{v}_m = (v_m^1, \dots, v_m^m)$ of m equidistant entries $v_m^j = \frac{j-1}{m-1}$ of the interval[2] $[0,1]$, i.e.,

$$\bar{z}_i = f(\boldsymbol{z}_i) = \boldsymbol{z}_i \cdot \boldsymbol{v}_m = \tfrac{1}{m-1} \textstyle\sum_{j=1}^{m} j z_i^j \tag{4.13}$$

as illustrated in Figure 4.3.

We define the representation $r(\boldsymbol{x})$ of some data point $\boldsymbol{x}$ by choosing the category with the highest probability for each dimension $i = 1, \dots, n$, i.e., we have

$$r(\boldsymbol{x})_i = v_m^k = \tfrac{k-1}{m-1} \text{ where } k = \underset{j=1,\dots,m}{\arg\max}\, \theta(\boldsymbol{x})_i^j, \tag{4.14}$$

for all $i = 1, \dots, n$. Thus, each representation is an element of the grid $\mathbb{G}^n$ with $\mathbb{G} = \{\frac{j}{m-1}\}_{j=0}^{m-1}$ as illustrated on the right side of Figure 5.6. In the following, we analyze the rotational equivariance of multivariate Gaussian distributions and show that this underlying grid structure of the discrete representation mitigates this problem.

4.2.3 Structural Advantages of the Discrete VAE

Rotations in the latent space present a significant challenge to disentanglement. The problem of these rotations becomes evident when considering the objective of disentanglement: the goal is to represent different generative factors of the data independently in the latent space. The presence of rotations could potentially entangle these generative factors, counteracting the original disentanglement goal. To better understand the influence of rotations on disentanglement, we first discuss the differences between the continuous and discrete approach.

Rotations have been observed in the case of a continuous (Gaussian) latent space [116]. In this context, the ELBO, given by Equation (4.11), depicts an inductive prior that encourages disentanglement by promoting neighboring points in the data space to be represented closely in the latent space [11]. As indicated by the work of Watters et al. [116], this property does not guarantee disentanglement. The feature of nearby points in the observable space mapping to nearby points in latent space is invariant under rotations over $\mathbb{R}^n$, while disentanglement is not.

When the latent space is represented by a general Gaussian distribution, rotations within this space do not change the structure of the distribution itself. Any rotation of the Gaussian distribution has the same likelihood when training the VAE, since the subsequent linear layer can immediately compensate for the rotation.

[2]The choice of the unit interval is arbitrary.

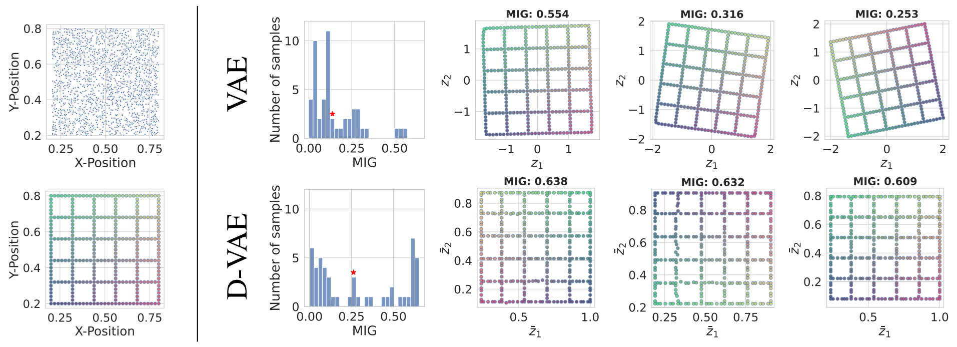


Figure 4.4: Geometry analysis of the latent space of the circles experiment [116]. **Col 1, top:** The generative factor distribution of the circles dataset. **Bottom:** A selective grid of points in generative factor space spanning the data distribution. **Col 2:** The Mutual Information Gap (MIG) [14] for 50 Gaussian VAE (top) and a categorical VAE (bottom), respectively. The red star denotes the median value. **Cols 3 - 5:** The latent space visualized by the representations of the selective grid of points. We show the best, 5th best, and 10th best models determined by the MIG score of the Gaussian VAE (top) and the categorical VAE (bottom), respectively. Unlike the Gaussian latent spaces, the discrete latent spaces are sensitive to the axes and generally yield better disentanglement scores.

Consequently, it is sensible to use a *diagonal* covariance matrix for the approximated posterior $q(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}\big(\boldsymbol{z} \mid \boldsymbol{\mu}(\boldsymbol{x}), \boldsymbol{\sigma}(\boldsymbol{x})\boldsymbol{I}\big)$. Rotations of a diagonal Gaussian distribution are generally not diagonal Gaussians themselves. Unfortunately, there are scenarios where rotations are problematic even with diagonal Gaussian distributions, as the following theorem indicates.

Theorem 2 (Rotational Equivariance)**.** *Let* $\alpha \in [0, 2\pi)$ *and let* $\boldsymbol{z} \sim \mathcal{N}\big(\boldsymbol{\mu}, \Sigma\big)$ *with* $\Sigma = \boldsymbol{\sigma}\boldsymbol{I},\ \boldsymbol{\sigma} = (\sigma_0, \dots, \sigma_n)$*. If* $\sigma_i = \sigma_j$ *for some* $i \neq j \in [n]$*, then* $\boldsymbol{z}$ *is equivariant under any* i, j*-rotation, i.e.,* $R^{\alpha}_{ij}\boldsymbol{z} \overset{d}{=} \boldsymbol{y}$ *with* $\boldsymbol{y} \sim \mathcal{N}\big(R^{\alpha}_{ij}\boldsymbol{\mu}, \Sigma\big)$.

Since, in the Gaussian VAE, the KL-divergence term in Equation (4.11) is invariant under rotations, Theorem 2 implies that its latent space can be arbitrarily rotated in dimensions i, j that have equal variances $\sigma_i = \sigma_j$. As in the case of the general Gaussian distribution, any rotation in dimensions i, j has the same likelihood when training the VAE, since the subsequent linear layer can immediately compensate for the rotation. We give two examples to clarify this property.

Example 1. *Let* $\Sigma = \begin{bmatrix} 0.2 & 0 & 0 \\ 0 & 0.2 & 0 \\ 0 & 0 & 0.4 \end{bmatrix}$ *and let* $R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\frac{\pi}{4} & -\sin\frac{\pi}{4} \\ 0 & \sin\frac{\pi}{4} & \cos\frac{\pi}{4} \end{bmatrix}$ *be the rotational matrix that rotates the last two dimensions* 45 *degrees. Then*

$$R\Sigma R^{\intercal} = \begin{bmatrix} 0.2 & 0 & 0 \\ 0 & 0.3 & -0.1 \\ 0 & -0.1 & 0.3 \end{bmatrix}.$$

Example 1 presents the case of a rotation along two axes with different variance values. The rotation of a diagonal covariance matrix is, in general, not diagonal itself.

Example 2. *Let* $\Sigma = \begin{bmatrix} 0.2 & 0 & 0 \\ 0 & 0.2 & 0 \\ 0 & 0 & 0.4 \end{bmatrix}$ *and let* $R = \begin{bmatrix} \cos\frac{\pi}{4} & -\sin\frac{\pi}{4} & 0 \\ \sin\frac{\pi}{4} & \cos\frac{\pi}{4} & 0 \\ 0 & 0 & 1 \end{bmatrix}$ *be the rotational matrix that rotates the first two dimensions* 45 *degrees. Then*

$$R\Sigma R^{\intercal} = \begin{bmatrix} 0.2 & 0 & 0 \\ 0 & 0.2 & 0 \\ 0 & 0 & 0.4 \end{bmatrix} = \Sigma.$$

Example 2 presents the case of a rotation along two axes with the same variance values. The diagonal covariance matrix is invariant under any rotation along these axes. Thus, any rotation also depicts a diagonal covariance matrix.

Equal variances can occur, for example, when different factors exert a similar influence on the data space, e.g., X-position and Y-position. Another example is factors where the variances are close to zero, i.e., factors where potential confusion during sampling would cause very high log-likelihood costs.

In contrast, the discrete latent space is invariant only under those rotations that are axially aligned. In the discrete case, the latent space is a subset of the grid $\mathbb{G}^n$, as we have introduced in Equation (4.14) and illustrated in Figure 5.6 (right). Distances and rotations exhibit different geometric properties on $\mathbb{G}^n$ than on $\mathbb{R}^n$. First, the closest neighbors are axially aligned. Non-aligned points have a distance at least $\sqrt{2}$ times larger. Secondly, $\mathbb{G}^n$ is invariant only under those rotations that are axially aligned.

We illustrate this with an example in Figure 4.4. Here, we depict the 2-dimensional latent space of a Gaussian VAE and a D-VAE model, respectively. These models are trained on a dataset generated from the two ground-truth factors: X-position and Y-position. We trained 50 copies of each model and depicted the best, the 5th best, and the 10th best latent spaces based on their disentanglement scores, as measured by the Mutual Information Gap (MIG) [14]. All three latent spaces of the Gaussian VAE exhibit rotation, with the disentanglement score strongly correlated with the angle of rotation. This becomes even more visible in Figure B.2 in Appendix B, which illustrates all 50 latent spaces. The rotations appear to be entirely random.

Figure 4.4 (bottom right) illustrates the 2-dimensional latent space of a D-VAE model trained on the same dataset, using the same random seeds as the Gaussian VAE model. In contrast to the Gaussian latent spaces, the discrete latent spaces are axis-sensitive and generally yield better disentanglement scores. The set of all 100 latent spaces is available in Figures B.2 and B.3 in Appendix B.

An important note is that being robust against rotations is not sufficient for achieving disentangling properties. For example, a random shuffling of the latent representations on the grid would remove any disentangling properties. In Section 5.3, we delve into more detail, explaining how exactly the D-VAE model facilitates disentanglement.

In the following, we aim to prove Theorem 2. Initially, we prove the theorem for the case $n = 2$. In this case, $\sigma_1 = \sigma_2$ and thus, the covariance matrix is a multiple of the identity. More precisely, we propose the following Lemma.

Lemma 3. *Let $\alpha \in [0, 2\pi)$ and let $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$ with $\boldsymbol{\mu} = (\mu_1, \mu_2)$ and $\Sigma = \boldsymbol{\sigma I}$, $\boldsymbol{\sigma} = (\sigma_1, \sigma_2)$. If $\sigma_1 = \sigma_2$, then $\boldsymbol{z}$ is equivariant under any rotation, i.e., $R^\alpha \boldsymbol{z} \overset{d}{=} \boldsymbol{y}$ with $\boldsymbol{y} \sim \mathcal{N}(R^\alpha \boldsymbol{\mu}, \Sigma)$.*

Proof. For the sake of clarity, we write $R := R^\alpha$. We know that $R\boldsymbol{z} \overset{d}{=} \boldsymbol{y}'$ with $\boldsymbol{y}' \sim \mathcal{N}(R\boldsymbol{\mu}, R\Sigma R^\intercal)$. Thus, we need to show that $R\Sigma R^\intercal = \Sigma$. Let $\hat{\sigma} := \sigma_1 = \sigma_2$. We have $\boldsymbol{\sigma} = (\hat{\sigma}, \hat{\sigma})$ and

$$R\Sigma R^\intercal = R\boldsymbol{\sigma I}R^\intercal = R\hat{\sigma}\boldsymbol{I}R^\intercal = \hat{\sigma}RR^\intercal = \boldsymbol{\sigma I} = \Sigma.$$

The second last equation follows since every rotation matrix is orthogonal. □

Rotations that act on exactly two axes and keep all other dimensions constant behave very similarly to rotations on the 2-dimensional plane. We use this idea and Lemma 2 to prove Theorem 2 by utilizing a change of basis to rotate over the first two axes.

Proof of Theorem 2. For the sake of clarity, we write $R := R^\alpha_{ij}$. We know that $R\boldsymbol{z} \overset{d}{=} \boldsymbol{y}'$ with $\boldsymbol{y}' \sim \mathcal{N}(R\boldsymbol{\mu}, R\Sigma R^\intercal)$. Thus, we need to show that $R\Sigma R^\intercal = \Sigma$.

Let $\hat{\sigma} := \sigma_i = \sigma_j$. For $n = 1$, there is nothing to prove. We have proven the case of $n = 2$ in Lemma 3. For $n > 2$, we use a change of basis to rotate over the first two axes. Let P be the permutation matrix that swaps $\boldsymbol{e}_1 \leftrightarrow \boldsymbol{e}_i$ and $\boldsymbol{e}_2 \leftrightarrow \boldsymbol{e}_j$. We define $\bar{\boldsymbol{\sigma}} = (\sigma_3, \sigma_4, \ldots, \sigma_{i-1}, \sigma_1, \sigma_{i+1}, \ldots, \sigma_{j-1}, \sigma_2, \sigma_{j+1}, \ldots, \sigma_n)$ to be the (n-2)-dimensional covariance vector without σ_i and σ_j. Furthermore, we have $P = P^{-1} = P^\intercal$ and

$$\begin{aligned}
R\Sigma R^\intercal &= P^\intercal P R P^\intercal P \Sigma P^\intercal P R^\intercal P^\intercal P \\
&= P^\intercal R_P \Sigma_P R_P^\intercal P \\
&= P^\intercal \begin{bmatrix} R_{12} & 0 \\ 0 & \boldsymbol{I}_{n-2} \end{bmatrix} \begin{bmatrix} \hat{\sigma}\boldsymbol{I}_2 & 0 \\ 0 & \bar{\boldsymbol{\sigma}}\boldsymbol{I}_{n-2} \end{bmatrix} \begin{bmatrix} R_{12}^\intercal & 0 \\ 0 & \boldsymbol{I}_{n-2} \end{bmatrix} P \\
&= P^\intercal \begin{bmatrix} R_{12}\hat{\sigma}\boldsymbol{I}_2 R_{12}^\intercal & 0 \\ 0 & \bar{\boldsymbol{\sigma}}\boldsymbol{I}_{n-2} \end{bmatrix} P \\
&= P^\intercal \begin{bmatrix} \hat{\sigma}\boldsymbol{I}_2 & 0 \\ 0 & \bar{\boldsymbol{\sigma}}\boldsymbol{I}_{n-2} \end{bmatrix} P \\
&= P^\intercal \Sigma_P P \\
&= \Sigma.
\end{aligned}$$

Here, we used Lemma 3 for the equality $R_{12}\hat{\sigma}\boldsymbol{I}_2 R_{12}^\intercal = \hat{\sigma}\boldsymbol{I}_2$. □

4.2.4 Conclusion

In this section, we analyzed the structural benefits of discrete representations in deep learning applications, particularly addressing research question **RQ1.2**. As a main result, we analyzed the implications of the rotational equivariance, as demonstrated by Theorem 2. In a Gaussian latent space, which is common in continuous representations, rotations can potentially entangle the generative factors the model aims to separate.

In contrast, we proposed a discrete variational autoencoder (D-VAE), which models a joint distribution of Gumbel-softmax random variables. The D-VAE creates a discrete grid in the latent space that mitigates the rotational problem observed in the continuous counterpart. This structural advantage might explain the observed better generalization behavior in models employing discrete representations.

Although the D-VAE offers an approach to tackle the rotational equivariance problem, as noted, being robust against rotations alone does not suffice to achieve disentangling properties. We will delve into more detail in Section 5.3 on how exactly the D-VAE model encourages disentanglement.

Chapter 5

Methodology

This chapter introduces our methods to efficiently learn the structure and parameters of discrete Stochastic Computation Graphs (SCGs), enhance their integration within deep learning models, and further optimize them using previous findings from the disentanglement literature.

Section 5.1 explains how to use supervised learning to address the difficulties of learning the structure of SCGs in the context of Neural Architecture Search. This strategy aims to capture the underlying distributions of the stochastic nodes, that is, the choice of neural architecture operation, side-stepping the difficulties of differentiating through them.

Section 5.2 presents our strategies to counter the challenges associated with learning the parameters of discrete SCGs. To address common issues such as small gradients and local minima, we propose tweaking the scale parameter of Gumbel noise perturbations and implementing dropout residual connections in discrete-continuous computation graphs.

In Section 5.3, we discuss the integration of discrete representations into deep learning methods based on VAEs. By replacing the standard Gaussian VAE with an adapted categorical one, we aim to obtain discrete latent space with the unique property of well-defined ordered categories encouraging disentanglement.

Lastly, Section 5.4 discusses several strategies to further enhance the efficiency of discrete representations. This includes the introduction of a total correlation (TC) regularizer, leveraging semi-supervised training, and addressing the straight-through gap. These adaptations enable the development of more robust, interpretable, and efficient discrete representations.

5.1 Efficient Structure Learning of Stochastic Computation Graphs

Neural network architectures define the structure and behavior of the models employed to learn from complex, high-dimensional data. As the depth and complexity of these models have increased over time, the task of designing them has progressively shifted from human engineers to automated algorithms. This is broadly referred to as Neural Architecture Search (NAS) [24].

We can connect our research on discrete representation learning with NAS by representing possible architectures as Stochastic Computation Graphs (SCGs). SCGs enable NAS to model the combinatorially complex space of potential architectures and apply stochastic search strategies to explore it. In this framework, the stochastic nodes represent the randomness of selecting specific architecture choices. These often discrete distributions range from categorical for the operation choice, e.g., a convolutional layer or max pooling, to binary for the edge indication.

Previous NAS research is limited by lengthy computation times and the necessity of extensive computational resources for the recurrent search and evaluation of new candidate architectures. We believe that the introduction of *NAS-Bench-101* [123], a dataset encompassing over $423\,000$ fully trained neural architectures, marks a paradigm shift in NAS research, thus potentially mitigating some of the previously mentioned barriers. The ability to experiment with data-driven methods, such as supervised learning to evaluate neural architectures, could enable a broader spectrum of researchers to contribute to this field. This brings us to our research question **RQ2.1:** *How can we effectively learn the structure and the parameters of discrete SCGs?*

In this section, we address the first half of this question by presenting an efficient way of learning the structure of the SCG presenting the neural architecture. We explore the use of supervised learning to capture the underlying distributions of the stochastic nodes, thereby side-stepping the need to differentiate through them directly. By learning these distributions, we can effectively predict the most promising SCGs, making the search process of neural architectures more efficient.

The following subsections detail the challenges and techniques involved in this process, particularly in the context of NAS. We further illustrate how our approach helps to manage the uncertainty related to the stochastic nodes and aids in learning the correct graph structure, thereby overcoming the challenges associated with training stochastic computation graphs.

The content of this section primarily draws upon the following publications:

> Friede, D., Lukasik, J., Stuckenschmidt, H., & Keuper, M. (2019). *A variational-sequential graph autoencoder for neural architecture performance prediction*. arXiv preprint arXiv:1912.05317.
>
> Lukasik, J., Friede, D., Stuckenschmidt, H., & Keuper, M. (2020). *Neural architecture performance prediction using graph neural networks*. In DAGM GCPR (pp. 188-201). Springer, Cham.

5.1.1 Background

Deep learning through convolutional neural architectures has been the primary driver behind the recent advancements in computer vision and related domains. Several interdependent factors such as the growing availability of training data and computing resources have contributed to this success. Arguably, the development of novel neural architectures [50, 28] has had the most significant impact. Thus, the focus of computer vision research has shifted from a feature engineering process to an architecture engineering process. The desire to automate this process using machine learning techniques is a direct result.

Neural Architecture Search (NAS) [24] deals with techniques that automate architecture engineering. Due to the lengthy computation times for the recurrent search and evaluation of new candidate architectures, for instance, in the proposed genetic algorithms or reinforcement learning [128], NAS research has been largely inaccessible for researchers lacking access to extensive computational systems.

However, the introduction of *NAS-Bench-101* [123], a dataset of over 423k fully trained neural architectures, facilitates a paradigm shift in NAS research. Instead of carfeully evaluating each new proposed neural architecture, NAS-Bench-101 allows experimentation with classical data-driven methods such as supervised learning to evaluate neural architectures. Subsequently, we address the task of reconstructing and generating neural architectures in a supervised manner using continuous representations of neural architectures.

Most current neural architectures for computer vision can be represented as directed, acyclic graphs (DAGs). Therefore, we base our approach on Graph Neural Networks. *Graph Neural Networks* (GNNs) [118] have proven to be very effective in understanding local node features and graph substructures. This makes them an extremely useful tool to embed nodes as well as complete graphs like the NAS-Bench-101 architectures into continuous spaces.

In this context, discovering new neural architectures equates to generating new graphs. From the few existing graph-generating models, sequential approaches like [124] or [55] are quite promising. The model *Deep Generative Models of Graphs* (DGMG) in [55], which employs GNNs, demonstrates superiority over Recurrent Neural Network (RNN) methods. In natural language processing, it is typical to use RNNs at both the encoder and decoder levels. Inspired by this approach, we expand the concepts of the DGMG model to create a *Variational-Sequential Graph Autoencoder* (VS-GAE), a variational autoencoder [45] that employs GNNs at both the encoder and decoder levels simultaneously. To the best of our knowledge, we are introducing the first graph autoencoder based on GNNs that operates on graphs of different sizes. This makes it a robust tool for handling neural architectures.

In summary, we make the following contributions: In Section 5.1.2, we further discuss GNNs and graph generating models. Next, we present a graph encoder in Section 5.1.3 based on GNNs and tailored to the NAS-Bench-101 neural architectures. Lastly, we introduce VS-GAE in Section 5.1.4, a novel variational-sequential graph autoencoder. VS-GAE specializes in encoding and decoding graphs of varying lengths using GNNs. This makes it a potent tool for designing new architectures and generating a meaningful graph latent space.

5.1.2 Continuous Graph Representations

The integration of modern machine learning methods with graph-structured data has increasingly gained popularity. This can be interpreted as an extension of deep learning techniques to non-Euclidean data [10] or even as introducing relational biases within deep learning architectures to enable combinatorial generalization [6]. Due to the discrete nature of graphs, they cannot be trivially optimized in differentiable learning methods that operate on continuous spaces. In this paper, we address this issue. We aim to use continuous methods to handle the graphs that characterize neural architectures from the NAS-Bench-101 dataset.

From Discrete to Continuous

The advancements in GNN research have led to breakthroughs in various areas related to graph analysis such as computer vision [119, 53, 121], natural language processing [5], recommender systems [69], chemistry [25], and others. The ability of GNNs to accurately model dependencies between nodes forms the foundation of our research. We leverage them to transition from the discrete graph space to the continuous space, and vice versa.

From Continuous to Discrete

Generating new graphs, especially new neural architectures, is an ambitious task that has to overcome multiple fundamental challenges. The primary focus is on the highly variable graph search space of NAS-Bench-101 and the complex dependencies within a single graph.

Global approaches like the one from Simonovsky and Komodakis [99] are restricted to a fixed and small number of nodes as they employ relaxations of the adjacency matrix which is inherently quadratic in the number of nodes. Luo et al. [63] were the first to use RNNs to generate neural architectures in a sequential manner. Their model acts on graphs with a fixed number of nodes and lacks the capability to induce complex graph structures. These issues were partially addressed by You et al. [124]. This model operates on graphs of variable sizes and uses a second edge-level RNN to capture edge dependencies. The superiority of using GNNs over RNNs during the graph generation process was demonstrated by Li et al. [55].

Our model can be interpreted as an extension of the conditional version of Li et al. [55]. To the best of our knowledge, our model is the first to utilize GNNs at both the encoding and decoding levels, thus creating a variational-sequential graph autoencoder that operates on graphs of different sizes. The model from Zhang et al. [126] is also related to our work; unlike our model, it operates on a fixed number of nodes. In contrast to our method, Zhang et al. [126] constructed a model on an asynchronous message passing scheme that encodes computations rather than graph structures.

5.1.3 The Graph Encoder

In this section we present our GNN-based model to encode the discrete graph space of NAS-Bench-101 into a continuous vector space. One can imagine a single GNN iteration as a two-step procedure. First, each node sends out a message to its neighbors alongside its edges. Second, each node aggregates all incoming messages to update itself. After a final amount of these iteration steps, the individual node embeddings are aggregated into a single graph embedding.

Node-Level Propagation

Let $G = (V, E)$ be a graph with nodes $v \in V$ and edges $e \in E \subseteq V \times V$. We denote $N(v) = \{u \in V \mid (u, v) \in E\}$ and $N^{out}(v) = \{u \in V \mid (v, u) \in E\}$ as the directed neighborhoods of a node $v \in V$. For each node $v \in V$, we associate an initial node embedding $h_v \in \mathbb{R}^{d_n}$. In our experiments we use a learnable look-up table based on the node types. Propagating information through the graph can be

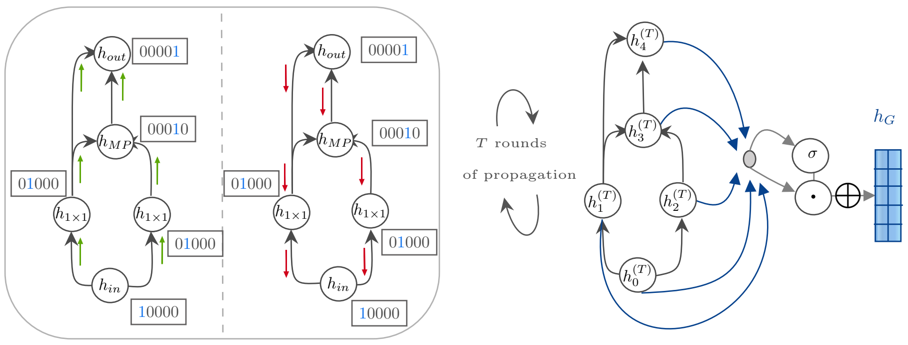


Figure 5.1: Visualization of the graph encoding process consisting of two main stages. **Left:** The process of node-level propagation is presented, which involves T rounds of bidirectional message passing to facilitate the transfer of information between interconnected nodes. The arrows indicate the direction of this message passing. **Right:** The graph-level aggregation is displayed. This stage involves the synthesis of all the individual node embeddings into a single, comprehensive graph embedding, denoted h_G. Together, these stages allow our model to effectively transform discrete graph representations into continuous vector spaces.

seen as an iterative *message-passing* process

$$m_{u\to v} = \Xi_{u\in N(v)}\big(M^{(t)}(h_v^{(t-1)}, h_u^{(t-1)})\big), \tag{5.1}$$

$$h_v^{(t)} = U^{(t)}(h_v^{(t-1)}, m_{u\to v}), \tag{5.2}$$

with a differentiable message module $M^{(t)}$ in (5.1), a differentiable update module $U^{(t)}$ in (5.2) and a differentiable, permutation invariant aggregation function Ξ. The message module $M^{(t)}$ is illustrated by the green arrows in Figure 5.1 (left). To address the directed nature of the NAS-Bench-101 graphs, we add a reverse message module

$$m_{u\to v}^{out} = \Xi_{u\in N^{out}(v)}\big(M_{out}^{(t)}(h_v^{(t-1)}, h_u^{(t-1)}\big), \tag{5.3}$$

$$h_v^{(t)} = U^{(t)}(h_v^{(t-1)}, m_{u\to v}, m_{u\to v}^{out}). \tag{5.4}$$

This is outlined in Figure 5.1 (left) by the red arrows and leads to so-called bidirectional message passing. The update module $U^{(t)}$ utilizes each node's incoming messages to update that node's embedding from $h_v^{(t-1)}$ to $h_v^{(t)}$.

Exploring many different choices for the message and update modules experimentally, we find that the settings similar to Li et al. [55] work best for our needs. We

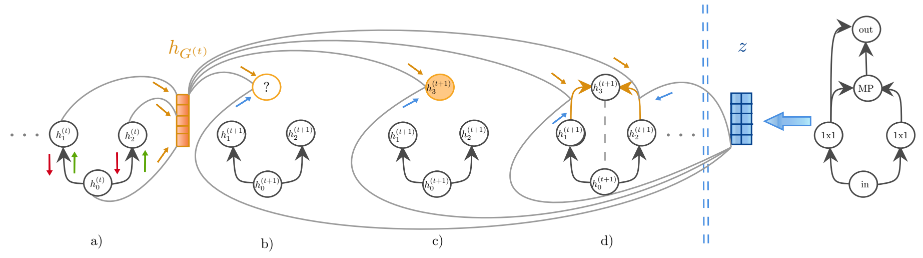


Figure 5.2: Illustration of a single iteration during the graph generation process. **a)** A decoder-level GNN propagates the node embeddings through the partially created graph and aggregates them to form a summary of this graph. **b)** A new node is created, and its node type is selected using the summary of both the partially created and original graphs. **c)** The newly created node is initialized with a node embedding. **d)** A score for all edges connecting the new node is calculated and evaluated, resulting in the set of new edges.

pick a concatenation together with a single linear layer for our message modules. The update module consists of a single Gated Recurrent Unit (GRU) where $h_v^{(t-1)}$ is treated as the hidden state. For the aggregation function, we choose the sum. To increase the capacity of our model, on the one hand, we apply multiple rounds of propagation and on the other hand, we use a different set of parameters for each round.

Graph-Level Aggregation

After the final round of message-passing, the propagated node embeddings $h = (h_v)_{v \in V}$ are aggregated into a single graph embedding $h_G \in \mathbb{R}^{d_g}$, where

$$h_G = A(h), \qquad (5.5) \qquad\qquad h_G^{\text{var}} = \tilde{A}(h). \qquad (5.6)$$

We obtain good results by using a linear layer combined with a gating layer that adjusts each node's fraction in the graph embedding. This aggregation layer A in (5.5) is further illustrated in Figure 5.1 (right). In case that variational outputs are required, we interpret h_G as the mean and add an extra graph aggregation layer $\tilde{A}$ in (5.6) which outputs the variance h_G^{var}.

5.1.4 The VS-GAE

In this section we present our Variational-Sequential Graph Autoencoder (VS-GAE). The VS-GAE is composed of the variational version of the graph-based encoder from Section 5.1.3 and a graph generating decoder using a sequential process. The encoder $q_\phi(\mathbf{z}|G)$ takes graph G with node labels as input and outputs a prior distribution $p(\mathbf{z})$ over the latent space in line with common variational autoencoders. The decoder $p_\psi(G|\mathbf{z})$ takes a sampled point $\mathbf{z}$ from this latent space $p(\mathbf{z})$ as input and generates a graph iteratively as a sequence of operations that add new nodes and edges until the end/output node is generated. Note that the sampled point $\mathbf{z}$ contains a summary of the original graph G.

Graph Generating Process

The decoder consists of multiple modules, mainly describing a distribution over the outcomes of a specific step in the generating sequence. Each module utilizes for each iteration t one or multiple of the following inputs:

$\mathbf{z}$	the sampled point from the latent space,
L	a look-up table based on the node types,
$h^{(t)}$	the embedding of the created nodes,
$G^{(t)}, h_{G^{(t)}}$	the partial graph and its embedding.

Note that the learnable embedding look-up table L is independent of the one in Section 5.1.3. Correspondent to the NAS-Bench-101 graphs, we begin the iteration with the start/input node that receives an initial node embedding $h^{(0)} = (h_0)$ according to the sampled point and the look-up table. With these preparations we can represent the full graph generating process through iterating over the following modules. Such iteration step can be tracked module by module through following Figure 5.2. Note that the modules' weights are shared over different iterations.

GraphProp This module processes the embedding $h^{(t)}$ of the previously created nodes together with their underlying graph structure $G^{(t)}$ for two complementary but distinct tasks. First, the node embeddings are updated by propagating through them. Second, the updated node embeddings are read out and aggregated into a single graph embedding,

$$h_{G^{(t)}}, h_p^{(t)} = f_{\text{prop}}(h^{(t)}, G^{(t)}). \tag{5.7}$$

This is illustrated in Figure 5.2 a). The graph embedding $h_{G^{(t)}}$ can be interpreted as a summary of the hitherto created partial graph. We use a GNN to propagate and aggregate the node embeddings $h^{(t)}$ in (5.7). More precisely, we use an exact copy of our encoder from Section 5.1.3 initialized with its own weights. This is motivated by NLP methods that use two distinct RNNs on the encoder-level and the decoder-level, respectively.

AddNode In this module, a new node is created and its node type is selected. The input is the summary of the original graph, i.e., the sampled point **z** from the latent space as well as the summary $h_{G^{(t)}}$ of the already created partial graph. The intention behind these inputs and the ones for the following modules is always the same: What does the original graph look like? What does the partially created graph look like? What is missing? This concludes in following module,

$$\mathrm{NodeType} = \mathrm{softmax}\big(f_{\mathrm{addNode}}(\mathbf{z}, h_{G^{(t)}})\big). \tag{5.8}$$

The *addNode* module is outlined in Figure 5.2 b). The output of (5.8) is a categorical distribution over all possible node types. Note that sampling over this distribution yields a one-hot encoding of a specific node type. In line with the structure of NAS-Bench-101 graphs, the iteration stops after running through the step that adds the end/output node.

InitNode This module initializes the node embedding of the just created node. The input is the sampled point **z**, the summary $h_{G^{(t)}}$ of the created graph and the embedding of the node type $L[\mathrm{type}]$ in the look-up table. This embedding of the new node is then added to the already existing propagated node embeddings,

$$h_{t+1} = f_{\mathrm{initNode}}(\mathbf{z}, h_{G^{(t)}}, L[\mathrm{type}]), \tag{5.9}$$

$$h^{(t+1)} = (h_{p,1}^{(t)}, \dots, h_{p,t}^{(t)}, h_{t+1}). \tag{5.10}$$

This process is further illustrated in Figure 5.2 c).

AddEdges In this module, the edges towards the newly created node are selected. For this purpose, a score between the new node h_{t+1} and each previous node is calculated, respectively. A high score stands for a high probability of an edge and vice versa. This is illustrated in Figure 5.2 d).

$$s_v = f_{\mathrm{addEdges}}(h_{t+1}, h_v, \mathbf{z}, h_{G^{(t)}}), \;\; h_v \in h_p^{(t)}, \tag{5.11}$$

$$\mathrm{Edges} = \sigma(s), \tag{5.12}$$

where *Edges* is a family of Bernoulli distributed random variables describing a probability for each possible edge e connecting the new node. Sampling over these distributions yields the new set of edges. We interpret each edge as directed towards the new node.

In all our experiments, we let f_{addNode}, f_{initNode} and f_{addEdges} be two-layer MLPs with ReLU nonlinearities.

Training and Loss

Recall that a VAE maximizes the lower bound estimator for a graph G and a latent space representation **z**:

$$\mathcal{L}(\psi, \phi; G) = \mathbb{E}_{q_\phi(\mathbf{z}|G)}\big[\log p_\psi(G|\mathbf{z})\big] + \mathrm{D}_{\mathrm{KL}}(q_\phi(\mathbf{z}|G) \| p_\psi(\mathbf{z})). \tag{5.13}$$

The first term of (5.13) is the model specific reconstruction loss which enforces high similarity between the input graph and the generated graph. The second term is the Kullback–Leibler Divergence which regularizes the latent space. In the following, we will discuss the reconstruction loss of VS-GAE.

We train the encoder and the decoder of VS-GAE jointly in an unsupervised manner. Although the encoder is by construction invariant under graph isomorphisms, the decoder needs a certain ordering over the nodes. To fulfill the prior of the decoder that each edge is directed towards the new node, this ordering has to be in such a manner that the adjacency matrix is an upper triangle matrix. This is, for example, given by the canonical ordering in which the graphs of NAS-Bench-101 are provided.

Given this fixed ordering of the nodes, we know the ground truth of the outputs of *AddNode* (5.8) and *AddEdges* (5.12) during training. One the one hand, we can use this ground truth to calculate a node-level loss $\mathcal{L}_V^i$ and an edge-level loss $\mathcal{L}_E^i$ at each iteration step, respectively. On the other hand, we can replace the model's output by the ground truth such that possible errors will not accumulate through iterations. This is also known as teacher forcing.

In order to calculate the overall reconstruction loss for a graph G, we sum up node losses and edge losses over all iterations

$$\mathcal{L}_{rec} = \mathcal{L}_V + \mathcal{L}_E. \tag{5.14}$$

Following Kingma and Welling [45], we assume $p_\psi(\mathbf{z}) \sim \mathcal{N}(\mathbf{z}; 0, \boldsymbol{I})$ as well as $p_\psi(G|\mathbf{z}) \sim \mathcal{N}(\mu, \Sigma)$. Furthermore, we approximate the posterior by a multivariate Gaussian distribution with diagonal covariance. This can be written as $\log q_\phi(\mathbf{z}|G) = \log \mathcal{N}(\mathbf{z}; \mu, \sigma^2 \boldsymbol{I})$.

5.1.5 Conclusion

The present research advances the Neural Architecture Search (NAS) field by employing supervised learning in the context of Stochastic Computation Graphs (SCGs) to model the combinatorially complex space of potential architectures.

Our study highlights the role of the NAS-Bench-101 dataset [123] as a potential reformation in NAS research. The dataset enables employing data-driven methods like supervised learning to generate these architectures, as we will further show empirically in Section 6.1. Using supervised learning to train the underlying distributions of stochastic nodes mitigates the inherent complexities mentioned in Section 4.1. This technique allows us to predict the most SCGs representing the architectures and effectively guides the search process of neural architectures.

We addressed the first half of the research question **RQ2.1** by presenting an efficient way of learning the structure of the SCG in the realm of NAS. In the following section, we tackle the second half of the research question by assuming a given structure of the SCG and presenting an efficient way of learning the parameters of discrete SCGs. Later, in Section 6.1.1, we will empirically show that it is possible to learn both the structure and the parameters simultaneously.

5.2 Efficient Parameter Learning of Discrete Stochastic Computation Graphs

In order to optimize learning models based on discrete Stochastic Computation Graphs (SCGs), we are interested in the following research question: **RQ2.1:** *How can we effectively learn the structure and the parameters of discrete SCGs?* This section aims to answer the second half of this question by exploring efficient strategies for learning the parameters of models that combine discrete and continuous components. Here, we focus on the common challenges of small gradients and local minima mentioned in Section 4.1.

We propose two novel strategies for mitigating these challenges. The first strategy, outlined in Section 5.2.2, involves tweaking the scale parameter β of the Gumbel noise perturbations. The second strategy, discussed in Section 5.2.3, proposes the implementation of dropout residual connections for discrete SCGs.

The content of this section primarily draws upon the following publication:

> Friede, D., & Niepert, M. (2021). *Efficient Learning of Discrete-Continuous Computation Graphs*. Advances in Neural Information Processing Systems, 34, 6720-6732.

5.2.1 Background

Numerous models for supervised and reinforcement learning benefit from combining discrete and continuous model components [76, 2, 102]. Discrete-continuous models that can be learned end-to-end are compositional, more likely to generalize effectively, and offer improved interpretability. A widely adopted approach for constructing discrete SCGs involves integrating discrete probability distributions into neural networks using stochastic softmax tricks [95, 77]. Previous work has primarily concentrated on computation graphs that contain a single discrete component on each of the graph's execution paths [41, 64]. In contrast, our work examines more complex SCGs, with multiple sequential discrete components.

We have demonstrated in Section 4.1 that optimizing the parameters of these models presents a significant challenge, primarily due to small gradients and the presence of local minima. To overcome these obstacles, we propose two innovative strategies in this section. Firstly, as shown in Section 5.2.2, we find that increasing the scale parameter β of the Gumbel noise perturbations can significantly improve the learning behavior of the models. An increase in β raises the probability of evading local minima during the training phase. Secondly, we suggest the use of dropout residual connections for discrete-continuous computation graphs in Section 5.2.3. By randomly skipping some discrete distributions, we ensure the provision of more informative gradients throughout the entire computation graph.

5.2.2 TEMPMATCH: Temperature Matching

We explore the behavior of two interdependent parameters: the Gumbel-softmax temperature τ and the Gumbel scale parameter β. First, we have the temperature parameter τ from the Gumbel-softmax trick (see Equation (4.4)). The purpose of this parameter is to make the output of the softmax function more or less discrete, that is, more or less concentrated on one of the categories. Second, and this is a new insight, we can adjust the scale parameter β of the Gumbel distribution. For scale parameter β, we sample the noise perturbation i.i.d. as $\epsilon_i \sim \mathrm{Gumbel}(0, \beta)$. If we use the Gumbel-max trick of Equation (4.3) we implicitly generate samples from the distribution

$$p(\boldsymbol{z}_i; \boldsymbol{\theta}) = \frac{\exp(\theta_i/\beta)}{\sum_{j=1}^{k} \exp(\theta_j/\beta)}. \tag{5.15}$$

Increasing the scale parameter β, therefore, makes the Gumbel-max samples more uniform and less concentrated. When using the Gumbel-softmax trick instead of the Gumbel-max trick, we obtain samples $\boldsymbol{z}_i$ that are more uniformly distributed over the categories and more discrete. Indeed, in the limit of $\beta \to \infty$, we obtain

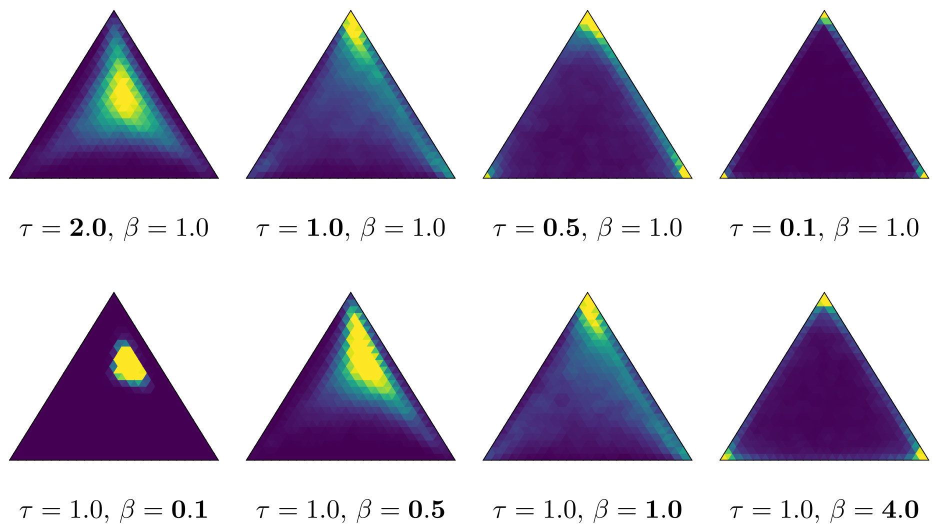


Figure 5.3: Visualization of two annealing schemes for the Gumbel-softmax trick for $\boldsymbol{\theta} = (2.0, 0.5, 1.0)$. Yellow depicts high, and blue depicts low values. **Top:** The standard procedure of annealing the softmax temperature τ as introduced in [64] (see also Equation (4.4)). **Bottom:** Starting with $\beta = 0$ and increasing it during training, the samples are increasingly more discrete but, compared to the standard annealing approach ($\tau \to 0$), more uniformly distributed over the corners of the probability simplex.

discrete samples uniformly distributed over the categories, independent of the logits. For $\beta \to 0$, we obtain the standard softmax function. Figure 5.3 illustrates the impact of the two parameters on the Gumbel-softmax distribution.

Now, the problem of insufficient gradients caused by local minima can be mitigated by increasing the scale parameter β *relative to* the temperature parameter τ. The annealing schedule we follow is the inverse exponential annealing $\beta_t = \tau(1 - e^{-t\gamma})$ for some $\gamma > 0$. Increasing β increases the probability of generating samples whose maximum probabilities are more uniformly distributed over the categories. This has two desired effects. First, samples drawn during training have a higher probability of counteracting poor minima caused at an earlier stage of the training. Figure 4.2 (left) shows that higher values for β allow the model to find its way out of poor minima. A higher value of β makes the model more likely to utilize categories with small θs, allowing the model to obtain improved minima (Figure 4.2 (middle)).

Second, gradients propagate to parameters of the upstream components even af-

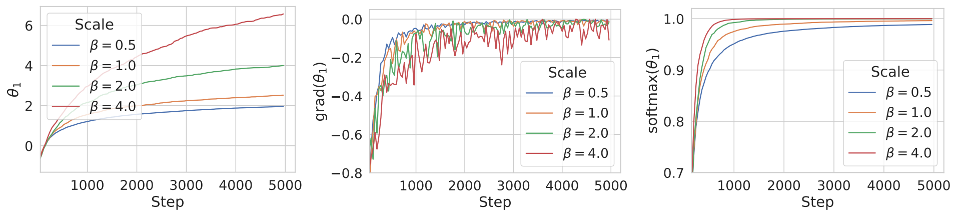


Figure 5.4: The values of the parameters, their gradients, and the softmax probabilities for various values of β (the scale parameter) for a constant Gumbel-Softmax parameter τ. Early during training, a lower scale parameter β (relative to τ) works well and has less variance. Once the probabilities saturate, however, we can only continue to obtain a sufficient gradient signal for larger values of β.

ter downstream components are saturating. To illustrate the second effect, we conducted the toy experiment depicted in Figure 5.4. The model here computes $\text{Softmax}(\boldsymbol{\theta} + \boldsymbol{\epsilon})$ with parameters $\boldsymbol{\theta}^\intercal = (\theta_1, \theta_2)$ and noise variables $\boldsymbol{\epsilon}^\intercal = (\epsilon_1, \epsilon_2)$, where $\epsilon_1, \epsilon_2 \sim \text{Gumbel}(0, \beta)$. The learning problem is defined through a cross-entropy loss between the output probabilities and the target vector $(1.0, 0.0)^\intercal$. We observe that early during training, lower values for β work well and exhibit less variance than higher values. However, once the probabilities and, therefore, the gradients start to saturate, a higher value for β enables the continued training of the model. While the example is artificial, it is supposed to show that larger values of the scale parameter β, sustain gradients for upstream components even if downstream components have saturated. Indeed, even though the softmax probability is close to 1, we observe a continuous increase of the parameter θ_1 and a high enough gradient signal for $\beta \geq 2$. It also shows that annealing the *inverse* scale parameter, that is, increasing the scale parameter relative to τ over time, should be beneficial.

In summary, increasing the Gumbel scale parameter β has positive effects on the training dynamics. We propose an exponential increasing scheme relative to τ. The two parameters τ and β should be *matched*, that is, optimized relative to each other. We show empirically that for more realistic learning problems choosing a higher value for β is beneficial, especially later during training.

5.2.3 DropRes: Residual Dropout Connections

Residual (or shortcut) connections play an important role when training deep neural networks [37]. A standard residual connection for the discrete-continuous components we consider here would be achieved by replacing Equation (4.2d) with

$$\boldsymbol{v} = \boldsymbol{u} + h_{\boldsymbol{w}'}(\boldsymbol{z}). \tag{5.16}$$

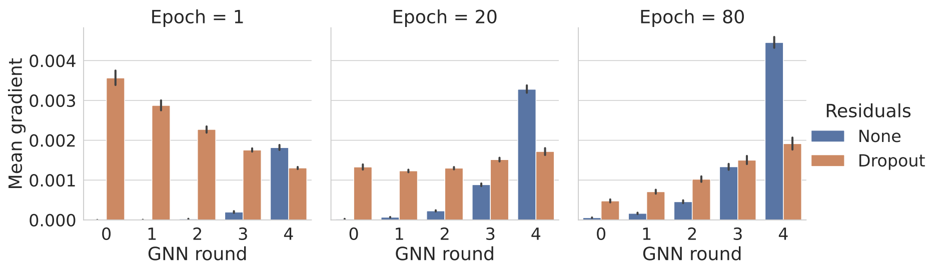


Figure 5.5: The influence of dropout residuals on the mean absolute gradients for ListOps. The values on the x-axis are the GNN message-passing iterations: the higher the number, the closer to the loss function is the corresponding discrete-continuous component. Adding dropout residual connections mitigates the vanishing gradient problem for components with greater distance to the loss.

By creating a direct connection to the continuous input of the discrete distributions, the optimizing problem simplifies. Unfortunately, what sets our problem apart from other end-to-end learnable neural networks is that we want to completely remove the residual connections at test time to obtain pure discrete representations and, therefore, the desired generalization properties. To mitigate the problem of overfitting to the existence of residual connections while at the same time reducing vanishing gradients in expectation, we propose DropRes connections. These are residual connections sampled from a Bernoulli distribution with parameter α. We replace Equation (4.2d) by

$$\boldsymbol{v} = \begin{cases} \boldsymbol{u} + h_{\boldsymbol{w}'}(\boldsymbol{z}) & \text{with probability } 1-\alpha \\ h_{\boldsymbol{w}'}(\boldsymbol{z}) & \text{with probability } \alpha, \end{cases} \tag{5.17}$$

that is, we drop out the residual connection with probability α. With probability $(1-\alpha)$ we obtain a *shortcut connection* between the neural components, effectively bypassing the Gumbel-softmax. With probability α the training path goes exclusively through the Gumbel-softmax distribution. The expectation of the gradients, taken over a Bernoulli random variable with parameter α, is now

$$\mathbb{E}\left[\frac{\partial \boldsymbol{v}}{\partial \boldsymbol{u}}\right] = (1-\alpha)\mathbf{1} + \frac{\partial \boldsymbol{v}}{\partial \boldsymbol{z}}\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{\theta}}\frac{\partial \boldsymbol{\theta}}{\partial \boldsymbol{u}}. \tag{5.18}$$

We propose a simple linear scheme to modify α from $0 \rightarrow 1$ during training. Hence, the model obtains a stronger gradient signal for upstream models *in expectation* in the early phase of training.

To illustrate the impact of dropout residual connections, we analyzed the gradients of message passing steps for the ListOps problem (see experiments) in a discrete-continuous computation graph of the type depicted in Figure 2.4. As we can observe in Figure 5.5, if we do not use dropout residual connections, the greater the distance of a discrete-continuous operation to the loss function (the lower the value on the x-axis), the smaller is the mean absolute value of the gradients reaching it. This illustrates the vanishing gradient problem. When using dropout residual connections, on the other hand, the mean absolute values of the gradients do not vanish proportional to their distance to the loss and are more evenly distributed.

To summarize, optimizing the parameters of models comprised of discrete distributions and continuous neural network components is challenging, mainly due to local minima and vanishing gradients. By setting the Gumbel scale parameter $\beta = 0$ as well as the dropout probability of the residual connections $\alpha = 0$ at the beginning of training, we obtain a continuous and deterministic relaxation of the target model. Increasing the DROPRES parameter α over time makes the model increasingly use the discrete stochastic nodes. Increasing the Gumbel scale parameter β allows the model to escape local minima and small gradients.

5.2.4 Conclusion

This section explored strategies for the efficient learning of the parameters of discrete SCGs that combine discrete and continuous components, thereby addressing the second half of our research question **RQ2.1**.

We have proposed two strategies to counter issues arising in the training of discrete-continuous computation graphs, such as small gradients and local minima. The first strategy, named TEMPMATCH, involves the manipulation of the scale parameter β of the Gumbel noise perturbations. Increasing β can improve the learning behavior of the models by promoting the models' ability to escape poor local minima and facilitating gradient propagation to upstream components, even when downstream components have saturated.

The second strategy, DROPRES, proposes the introduction of dropout residual connections in discrete-continuous computation graphs. By adding residual connections that bypass the Gumbel-softmax operation with a certain probability, we can manage the vanishing gradient problem, particularly prevalent in stochastic nodes with large distances to the loss function in the computation graph. This methodology reduces vanishing gradients, providing a more robust gradient signal during the early training phase.

We illustrate how these strategies support the training of discrete-continuous models in the Section 6.1, where we present empirical evidence demonstrating these models' superior performance and generalization capabilities on various benchmark datasets.

5.3 Learning Disentangled Discrete Representations

In this section, we analyze the integration of discrete representations by replacing the common Gaussian variational autoencoder (VAE) with a categorical one, specifically answering our research question **RQ2.2:** *"How can discrete representations be integrated effectively into existing deep learning methods?"* The focus lies on discussing the disentangling properties of our adapted categorical VAE.

In Section 4.2, we replaced the standard Gaussian VAE with a categorical one and explored how the distinct structure of categorical distributions overcomes the rotational invariance issue often associated with Gaussian distributions. A crucial step in this replacement is modifying the categorical VAE to use a one-dimensional representation for each category. This adaptation aligns the dimension of the categorical VAE with the one of the Gaussian VAE, facilitating a direct substitution.

Furthermore, we will define and discuss neighborhoods in the observable and the discrete latent space and demonstrate how similar observations are encouraged to be represented close together in the latent space. This property is important in understanding disentanglement and has been discussed for the Gaussian case by Burgess et al. [11]. We will show how many of their findings still hold in the case of our tailored categorical VAE, providing a first step to effectively integrating disentangling discrete representations into deep learning methods based on VAEs.

The content of this section primarily draws upon the following publication:

> Friede, D., Reimers, C., Stuckenschmidt, H., & Niepert, M. (2023). *Learning Disentangled Discrete Representations*. Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD

5.3.1 Background

Recent successes in image generation [110, 83], model-based reinforcement learning [35, 75, 36], and text-to-image generation [81, 91] have demonstrated the empirical advantages of discrete latent representations, although the reasons behind their benefits remain unclear [35]. We explore the relationship between discrete latent spaces and disentangled representations by replacing the standard Gaussian variational autoencoder (VAE) with a tailored categorical variational autoencoder. We showed in Section 4.2 that the underlying grid structure of categorical distributions mitigates the problem of rotational invariance associated with multivariate Gaussian distributions. In this section, we will additionally show that a discrete latent space acts as an efficient inductive prior for disentangled representations. We modify the categorical variational autoencoder to represent each category with a one-dimensional representation to inherit the canonical order of the real numbers. On the one hand, this change creates a latent space of the same dimension as the one of the Gaussian VAE, enabling a direct substitution of the VAE's distribution. Specifically, we utilize the same neural architecture for all methods so that all differences solely emerge from the type of distribution of the VAE. On the other hand, we will show that equipping the discrete latent space with an order creates a topology that maps similar observations to neighboring categories encouraging disentanglement.

In this section, we introduce some properties of the discrete latent space. Firstly, we discuss the concept of defining neighborhoods in the observable space. We hypothesize that the closest neighbors of an observation typically differ in only a single dimension of the ground-truth factors. Secondly, we show that mapping the discrete categories into a shared unit interval as in Equation (4.13) causes an ordering of the discrete categories and, in turn, enables a definition of neighborhoods in the latent space. Finally, we derive that the main argument from Burgess et al. [11] still holds in the discrete case, neighboring points in the data space are encouraged to be represented close together in the latent space.

5.3.2 Defining Neighborhoods in the Observable Space

Locatello et al. [59] showed that there is an infinite number of transformations of the ground truth factors $\boldsymbol{z} \sim p(\boldsymbol{z}) = \prod p(z_i)$ that lead to the same data distribution. A representation $r(\boldsymbol{x})$ that is fully disentangled with respect to $\boldsymbol{z}$ might be fully entangled with respect to such a transformation $\hat{\boldsymbol{z}}$. Without any inductive biases, unsupervised disentanglement is theoretically impossible. We will make use of two properties to mitigate this impossibility result. First, we can utilize the reconstruction loss to define *neighboring observations*

$$U_\epsilon(\boldsymbol{x}) = \Big\{\boldsymbol{y} \mid -\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_\psi(\boldsymbol{y}|\boldsymbol{z})\right] \leq \log\epsilon\Big\}. \tag{5.19}$$

Intuitively, the neighborhood $U_\epsilon(\boldsymbol{x})$ of some observation $\boldsymbol{x}$ are those observations/reconstructions $\boldsymbol{y}$ that have a high log-likelihood when encoding $\boldsymbol{x}$. This intuition becomes especially clear in the case of the mean squared error reconstruction loss since this loss function fulfills the properties of a metric. In this case, the neighborhood simplifies to $U_\epsilon(\boldsymbol{x}) = \left\{\boldsymbol{y} \mid \frac{1}{d}\|\boldsymbol{x}-\boldsymbol{y}\|_2^2 \leq \epsilon\right\}$, and neighboring observations are those with similar pixel values. We utilize a second property to

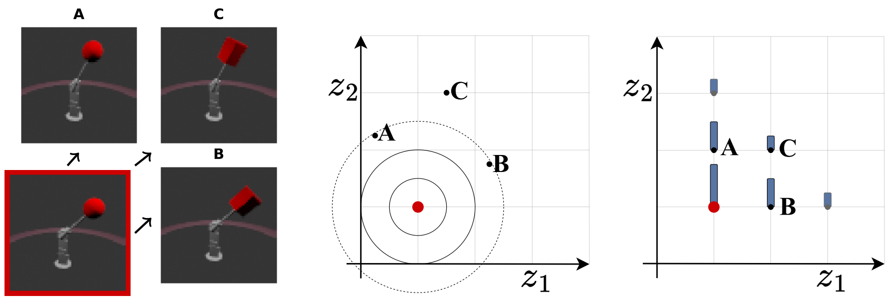


Figure 5.6: Four observations and their latent representation with a Gaussian and discrete VAE. Both VAEs encourage similar inputs to be placed close to each other in latent space. **Left:** Four examples from the MPI3D dataset [27]. The horizontal axis depicts the object's shape, and the vertical axis depicts the angle of the arm. **Middle:** A 2-dimensional latent space of a Gaussian VAE representing the four examples. Distances in the Gaussian latent space are related to the Euclidean distance. **Right:** A categorical latent space augmented with an order of the categories representing the same examples. The grid structure of the discrete latent space makes it more robust against rotations compared to its Gaussian counterpart, constituting a stronger inductive prior for disentanglement.

associate neighboring observations with small changes in the ground truth factors. Many datasets in the disentanglement literature consist of *discrete* ground truth factors [54, 86, 38, 44, 59, 27]. We argue that because of the discrete nature of many datasets, e.g., pixels, even continuous ground truth factors often convert into discrete changes in the data space. For instance, although we sample the X-position in the Circles dataset [116] from a random uniform distribution, we only obtain ~ 40 distinct observations regarding the X-position, see Figure 4.4 (left). As a consequence, we mostly observe incremental changes in the ground truth factors $\boldsymbol{z}$, that is, a small change in a single dimension z_i or z_j or both, but never *half* a change in z_i and z_j as illustrated in Figure 5.6 (left). We hypothesize that, consequently, the closest neighbors $\boldsymbol{x}'$ of $\boldsymbol{x}$ are generally those observations whose ground-truth factors $\boldsymbol{z}'$ differ in only a *single* dimension compared to the ground-truth factor $\boldsymbol{z}$ of $\boldsymbol{x}$. In the following, we will discuss neighborhoods in the latent space and eventually show that neighboring points in the data space are encouraged to be represented close together in the latent space enabling disentangling properties.

5.3.3 Neighborhoods in the Latent Space

In the Gaussian case, neighboring points in the observable space correspond to neighboring points in the latent space. As shown in the Section 5.3.2, the ELBO Loss Equation (4.11), more precisely the reconstruction loss as part of the ELBO, implies a topology of the observable space. In the case, where the approximated posterior distribution, $q_\phi(\boldsymbol{z}|\boldsymbol{x})$, is Gaussian and the covariance matrix, $\Sigma(\boldsymbol{x})$, is diagonal, the topology of the latent space can be defined in a similar way: The negative log-probability is the weighted Euclidean distance to the mean $\boldsymbol{\mu}(\boldsymbol{x})$ of the distribution

$$C - \log q_\phi(\boldsymbol{z}|\boldsymbol{x}) = \frac{1}{2}\left[(\boldsymbol{z} - \boldsymbol{\mu}(\boldsymbol{x}))^\intercal \boldsymbol{\Sigma}(\boldsymbol{x})(\boldsymbol{z} - \boldsymbol{\mu}(\boldsymbol{x}))\right]^2 = \sum_{i=1}^{n} \frac{(z_i - \mu_i(\boldsymbol{x}))^2}{2\sigma_i(\boldsymbol{x})} \tag{5.20}$$

where C denotes the logarithm of the normalization factor in the Gaussian density function. Neighboring points in the observable space will be mapped to neighboring points in the latent space to reduce the log-likelihood cost of sampling in the latent space [11].

In the case of categorical latent distributions, the induced topology is not related to the euclidean distance and, hence, it does not encourage that points that are close in the observable space will be mapped to points that are close in the latent space. The problem becomes explicit if we consider a single categorical distribution. In the latent space, neighbourhoods entirely depend on the shared representation space of the m classes. The canonical representation maps a class j into the one-hot vector $\boldsymbol{e}^j = (e_1, e_2, \ldots, e_m)$ with $e_k = 1$ for $k = j$ and $e_k = 0$ otherwise. The representation space consists of the m-dimensional units vectors, and all classes have the same pairwise distance between each other.

To overcome this problem, we inherit the canonical order of $\mathbb{R}$ by depicting a one-dimensional representation space. We consider the representation $\bar{z}_i = f(\boldsymbol{z}_i)$ from Equation (4.13) that maps a class j on the value $\frac{j-1}{m-1}$ inside the unit interval. In this way, we create an ordering on the classes $1 < 2 < \cdots < m$ and define the distance between two classes by $d(j,k) = \frac{1}{m-1}|j-k|$. In the following, we discuss properties of a VAE using this representation space.

5.3.4 Disentangling Properties of the Discrete VAE

In this section, we show that neighboring points in the observable space are represented close together in the latent space and that each data point is represented discretely by a single category j for each dimension $i \in \{1, \ldots, n\}$. First, we

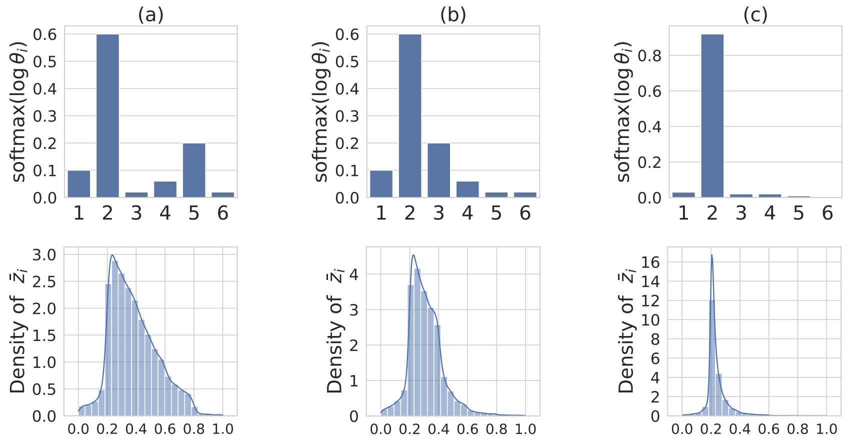


Figure 5.7: Illustration of three examples of (normalized) parameters of a single Gumbel-softmax distribution and the corresponding one-dimensional distribution of $\bar{z}_i$. Each m-dimensional sample $\boldsymbol{z}_i \sim \mathrm{GS}(\boldsymbol{\theta}_i)$ is mapped into the one-dimensional unit interval following Equation (4.13), i.e., via the dot product $\bar{z}_i = \boldsymbol{z}_i \cdot (0, \frac{1}{m-1}, \ldots, 1)$. In order to reduce the variance of $\bar{z}_i$, the model is encouraged to place high θ_i^j values to neighboring categories. Furthermore, having tail categories with $\theta_i^j \approx 0$ decreases the support of $\bar{z}_i$ further reducing its variance.

show that reconstructing under the latent variable $\bar{z}_i = f(\boldsymbol{z}_i)$ encourages each data point to utilize neighboring categories rather than categories with a larger distance. Second, we discuss how the Gumbel-softmax distribution is encouraged to approximate the discrete categorical distribution. For the Gaussian case, this property was shown by Burgess et al. [11]. Here, the ELBO, Equation (4.11) depicts an inductive prior that encourages disentanglement by encouraging neighboring points in the data space to be represented close together in the latent space [11]. To show these properties for the D-VAE, we use the following proposition.

Proposition 2. *Let $\boldsymbol{\theta}_i \in [0, \infty)^m$, $\boldsymbol{z}_i \sim \mathrm{GS}(\boldsymbol{\theta}_i)$ be as in Equation (4.12) and $\bar{z}_i = f(\boldsymbol{z}_i)$ be as in Equation (2.13). Define $j_{min} = \arg\min_j\{\theta_i^j > 0\}$ and $j_{max} = \arg\max_j\{\theta_i^j > 0\}$. Then it holds that*

(a) $\mathrm{supp}(f) = (\frac{j_{min}}{m-1}, \frac{j_{max}}{m-1})$

(b) $\frac{\theta_i^j}{\sum_{k=1}^m \theta_i^k} \to 1 \Rightarrow \mathbb{P}(z_i^j = 1) = 1 \wedge f(\boldsymbol{z}_i) = \mathbb{1}_{\{\frac{j}{m-1}\}}$.

Proposition 2 has multiple consequences. First, a class j might have a high density regarding $\bar{z}_i = f(\boldsymbol{z}_i)$ although $\theta_i^j \approx 0$. For example, if j is positioned between two other classes with large θ_i^k $\big($e.g. $j = 3$ in Figure 5.7 (a)$\big)$. Second, if there is a class j such that $\theta_i^k \approx 0$ for all $k \geq j$ or $k \leq j$, then the density of these classes is also almost zero $\big($Figure 5.7 (a-c)$\big)$. Note that a small support benefits a small reconstruction loss since it reduces the probability of sampling a wrong class. The probabilities of Figure 5.7 (a) and (b) are the same with the only exception that $\theta_i^3 \leftrightarrow \theta_i^5$ are swapped. Since the probability distribution in (b) yields a smaller support and consequently a smaller reconstruction loss while the KL divergence is the same for both probabilities,[1] the model is encouraged to utilize probability (b) over (a). This encourages the representation of similar inputs in neighboring classes rather than classes with a larger distance.

Consequently, we can apply the same argument as in Burgess et al. [11] (Section 4.2) about the connection of the posterior overlap with minimizing the ELBO. Since the posterior overlap is highest between neighboring classes, confusions caused by sampling are more likely in neighboring classes than those with a larger distance. To minimize the penalization of the reconstruction loss caused by these confusions, neighboring points in the data space are encouraged to be represented close together in the latent space. Similar to the Gaussian case [11], we observe an increase in the KL divergence loss during training while the reconstruction loss continually decreases. The probability of sampling confusion and, therefore, the posterior overlap must be reduced as much as possible to reduce the reconstruction loss. Thus, later in training, data points are encouraged to utilize exactly one category while accepting some penalization in the form of KL loss, meaning that $\theta_i^j / (\sum_{k=1}^m \theta_i^k) \to 1$. Consequently, the Gumbel-softmax distribution approximates the discrete categorical distribution, see Proposition 2 (b). An example is shown in Figure 5.7(c). This training behavior results in the unique situation in which the latent space approximates a discrete representation while its classes maintain the discussed order and the property of having neighborhoods.

In the following, we aim to prove Proposition 2. We mainly use properties of the Gumbel-Softmax distribution discussed in Section 2.4. To prove Part (a) of Proposition 2, we need to understand how to sample from $\boldsymbol{z}_i \sim \mathrm{GS}(\boldsymbol{\theta}_i)$ in the case where there exists a category j with $\theta_j = 0$. In this case, the probability of category j is zero, and thus, the jth dimension of $\boldsymbol{z}_i$ will always be zero. This is the same as sampling from the (n-1)-dimensional Gumbel-Softmax distribution $\bar{\boldsymbol{z}}_i \sim \mathrm{GS}(\bar{\boldsymbol{\theta}}_i)$, where $\bar{\boldsymbol{\theta}}_i = (\theta_i^1, \ldots, \theta_i^{j-1}, \theta_i^{j+1}, \ldots, \theta_i^m)$ and re-adding a zero into the dimension j of $\bar{\boldsymbol{z}}_i$. To prove Part (b) of Proposition 2, we initially show an

[1]The KL divergence is invariant under permutation.

interesting property of the Gumbel-Softmax distribution, namely that it is invariant under factor multiplication. More precisely, we propose the following lemma.

Lemma 4. *Let* $\boldsymbol{\theta} \in [0, \infty)^m$*,* $\mathrm{GS}(\boldsymbol{\theta})$ *be as in Equation (4.12). Let* $c > 0$*, then it holds that*

$$\mathrm{GS}(\boldsymbol{\theta}) = \mathrm{GS}(c\boldsymbol{\theta}). \tag{5.21}$$

Proof. By Proposition 1, we know that the density of $\mathrm{GS}(\boldsymbol{\theta}, 1)$ is

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \frac{(m-1)!}{(\sum_{j=1}^{m} \theta_j x_j^{-1})^m} \prod_{l=1}^{m} \frac{\theta_l}{x_l^2}.$$

Thus, simple calculus gives

$$\begin{aligned} p_{c\boldsymbol{\theta}}(\boldsymbol{x}) &= \frac{(m-1)!}{(\sum_{j=1}^{m} c\theta_j x_j^{-1})^m} \prod_{l=1}^{m} \frac{c\theta_l}{x_l^2} \\ &= \frac{(m-1)!}{(c\sum_{j=1}^{m} \theta_j x_j^{-1})^m} c^m \prod_{l=1}^{m} \frac{\theta_l}{x_l^2} \\ &= \frac{c^m(m-1)!}{c^m(\sum_{j=1}^{m} \theta_j x_j^{-1})^m} \prod_{l=1}^{m} \frac{\theta_l}{x_l^2} \\ &= \frac{(m-1)!}{(\sum_{j=1}^{m} \theta_j x_j^{-1})^m} \prod_{l=1}^{m} \frac{\theta_l}{x_l^2} \\ &= p_{\boldsymbol{\theta}}(\boldsymbol{x}). \end{aligned}$$

It follows that $\mathrm{GS}(\boldsymbol{\theta}) = \mathrm{GS}(c\boldsymbol{\theta})$. □

We will now prove Proposition 2 in two parts.

Proof of Proposition 2. For the sake of clarity, we ignore the i-index in our notation and write the j-index as a subscript.

Part (a): Let J be the set of all indices of $\boldsymbol{\theta}$ with $\theta_j = 0$ and let $m' = m - |J|$ be the number of elements of $\boldsymbol{\theta}$ that are non-zero. We will first show that

$$\mathrm{supp}\big(\mathrm{GS}(\boldsymbol{\theta})\big) = \mathrm{int}\{\boldsymbol{y} \in \mathbb{R}^n \mid y_j \in [0,1], \sum_{j=1}^{m} y_j = 1, y_k = 0 \text{ for } k \in J\}.$$

Let $P_{\boldsymbol{\theta}} : \mathbb{R}^m \to \mathbb{R}^{m'}$ be the projection that maps $\boldsymbol{\theta}$ on its non-zero elements $\boldsymbol{\theta}' = P_{\boldsymbol{\theta}}(\boldsymbol{\theta})$ with $\theta'_j \neq 0$ for all $j \in [m']$. We write $P_{\boldsymbol{\theta}}^{-1}(\boldsymbol{\theta}') = \boldsymbol{\theta}$ for the inverse of the projection. Sampling $z \sim \mathrm{GS}(\boldsymbol{\theta})$ is then defined by $P_{\boldsymbol{\theta}}^{-1}(\boldsymbol{z}')$ for $\boldsymbol{z}' \sim \mathrm{GS}(\boldsymbol{\theta}')$. By Proposition 1, we know that the density of $\mathrm{GS}(\boldsymbol{\theta}')$ is

$$p_{\boldsymbol{\theta}'}(\boldsymbol{x}) = \frac{(m'-1)!}{(\sum_{j=1}^{m'} \theta'_j x_j^{-1})^{m'}} \prod_{k=1}^{m'} \frac{\theta'_k}{x_k^2},$$

which is defined for all $x \in \Delta^{m'-1}$ with $x_j > 0$ for all $j \in [m']$. Furthermore, we have $p_{\boldsymbol{\theta}'}(\boldsymbol{x}) > 0$ for all $x \in \operatorname{int} \Delta^{m'-1}$ since, in this case, $p_{\boldsymbol{\theta}'}(\boldsymbol{x})$ consists of a sum and products of a finite number of positive elements. By definition of $z \sim \mathrm{GS}(\boldsymbol{\theta})$ we reverse the projection $P_{\boldsymbol{\theta}}$ to obtain $\operatorname{supp}\big(\mathrm{GS}(\boldsymbol{\theta})\big) = \operatorname{int}\{\boldsymbol{y} \in \mathbb{R}^n \mid y_j \in [0,1], \sum_{j=1}^m y_j = 1, y_k = 0 \text{ for } k \in J\}$.
We will now show that $\operatorname{supp}(f) = (\frac{j_{\min}}{m-1}, \frac{j_{\max}}{m-1})$. First, let $\boldsymbol{z} \in \operatorname{supp}\big(\mathrm{GS}(\boldsymbol{\theta})\big)$, then it holds that

$$f(z) = \frac{1}{m-1} \sum_{j=1}^{m} j z_j = \frac{1}{m-1} \sum_{j=j_{\min}}^{m} j z_j > \frac{1}{m-1} j_{\min}.$$

With the same argument, we can show that $f(z) < \frac{j_{\max}}{m-1}$. Conclusively, we will show that

$$\forall \tilde{z} \in (\frac{j_{\min}}{m-1}, \frac{j_{\max}}{m-1})\ \exists \boldsymbol{z} \in \operatorname{supp}\big(\mathrm{GS}(\boldsymbol{\theta})\big) \text{ with } \tilde{z} = f(\boldsymbol{z}).$$

Let $\tilde{z} \in (\frac{j_{\min}}{m-1}, \frac{j_{\max}}{m-1})$, then there exists $\delta \in (0,1)$ with $\tilde{z} = \delta \frac{j_{\min}}{m-1} + (1-\delta)\frac{j_{\max}}{m-1}$. Choose $\boldsymbol{z}$ with

$$z_j = \begin{cases} \delta, & \text{if } j = j_{\min}, \\ 1-\delta, & \text{if } j = j_{\max}, \\ 0, & \text{otherwise} \end{cases}$$

to conclude the proof of Part (a).

Part (b): Let $c > 0$. By Lemma 4, we know that $\mathrm{GS}(\boldsymbol{\theta}) = \mathrm{GS}(c\boldsymbol{\theta})$. We will now show that $\frac{\theta_k}{\theta_j} \to 0$ for all $j \neq k$ and thus, $\frac{\boldsymbol{\theta}}{\theta_j} \to \boldsymbol{e}^j$ with

$$e_k^j = \begin{cases} 1, & \text{if } k = j, \\ 0, & \text{otherwise} \end{cases}$$

and therefore, $\mathrm{GS}(\boldsymbol{\theta}) = \mathrm{GS}(\frac{1}{\theta_j}\boldsymbol{\theta}) \to \mathrm{GS}(\boldsymbol{e}^j)$ to conclude the proof. By assumption, we have

$$\frac{1}{\sum_{k=1}^m \frac{\theta_k}{\theta_j}} = \frac{\frac{\theta_j}{\theta_j}}{\frac{1}{\theta_j}\sum_{k=1}^m \theta_k} = \frac{\theta_j^{-1}}{\theta_j^{-1}} \frac{\theta_j}{\sum_{k=1}^m \theta_k} = \frac{\theta_j}{\sum_{k=1}^m \theta_k} \to 1$$

and thus, $\sum_{k=1}^{m} \frac{\theta_k}{\theta_j} \to 1$. It holds that $\sum_{k=1}^{m} \frac{\theta_k}{\theta_j} = 1 + \sum_{j \neq k} \frac{\theta_k}{\theta_j}$ and thus, $\sum_{j \neq k} \frac{\theta_k}{\theta_j} \to 0$. Since $\frac{\theta_k}{\theta_j} \geq 0$ for all $j \neq k$, we have $\frac{\theta_k}{\theta_j} \to 0$ and the proof follows. $\square$

5.3.5 Conclusion

The section addresses the research question **RQ2.2**. We detailed the effects of replacing a standard Gaussian Variational Autoencoder (VAE) with an adapted categorical one. By introducing a one-dimensional representation for each category, the discrete method creates a well-defined latent space with ordered categories. This property encourages disentanglement, with the arrangement of the categories reflecting changes in the observable space.

We demonstrated how to define neighborhoods in both the observable and the latent space, with data points that are close together in the data space being represented close in the latent space. This alignment encourages a disentangling effect in our model. We further provided Proposition 2, which showcases how the Gumbel-softmax distribution can approximate the discrete categorical distribution effectively while keeping these disentangling properties.

Our adapted categorical autoencoder illustrates a promising step towards efficiently integrating discrete representations into deep learning methods based on VAEs. As a result, we can transfer findings from the disentanglement theory to further improve discrete representations. We will discuss this in more detail in Section 5.4. In Section 6.2, we will also show empirically how our categorical VAE improves disentangled representations over its Gaussian counterpart.

5.4 Improving Discrete Representations

In Section 5.3, we have shown how to efficiently integrate discrete representations into deep learning methods based on VAEs. Aligning the categorical VAE with the Gaussian one allows us to utilize some of the main results from the disentanglement literature to further improve the discrete representations of categorical VAEs. Thus, the focus of this section is the research question **RQ2.3:** *How can we further enhance the performance and efficiency of common discrete representations?*

While regularizers encouraging disentanglement have been widely discussed [38, 52, 11, 44, 14], their role in models based on discrete latent spaces is minimal. To address this gap, we propose a version of total correlation (TC) [114] regularizer as introduced for Gaussian VAEs by Kim and Mnih [44] and Chen et al. [14]. By regularizing total correlation, each latent dimension can capture a unique and independent aspect of the data variance, leading to a more disentangled latent space.

We then explore semi-supervised training for enhancing discrete representations. In contrast to the Gaussian case [59], we can straight-forwardly incorporate the number of unique variations of ground-truth factors improving disentanglement and interpretability in the discrete case.

The final subsection addresses the straight-through gap, which enables selecting those models based on unsupervised scores whose latent spaces approximate the discrete latent spaces the most to improve disentangling properties.

The content of this section primarily draws upon the following publication:

> Friede, D., Reimers, C., Stuckenschmidt, H., & Niepert, M. (2023). *Learning Disentangled Discrete Representations*. Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD

5.4.1 Background

We have shown in Section 4.2 and in Section 5.3 how to efficiently replace the Gaussian distribution of methods based on Variational Autoencoders (VAEs) with a tailored categorical one. This categorical VAE encourages latent spaces based on disentangled representations. Such representations promise interpretability [39, 1], fairness [58, 17, 106], and better sample complexity for learning [93, 7, 79, 111].

State-of-the-art unsupervised disentanglement methods enrich variational autoencoders based on Gaussian distributions [45] with regularizers encouraging disentangling properties [38, 52, 11, 44, 14]. Locatello et al. [59] have shown that unsupervised disentanglement without inductive priors is theoretically impossible. Thus, a recent line of work has shifted to weakly-supervised disentanglement [61, 97, 60, 47]. While regularization and supervision encouraging disentangling properties have been discussed extensively in the disentanglement literature, they play little to no role in models based on discrete latent spaces.

Locatello et al. [59] have further shown that disentanglement properties strongly depend on the initial parameters of the model (the random seed). They argue that model selection should not depend on the considered disentanglement score since these heavily rely on ground-truth labels, which are not present in real-world datasets. We will show that the categorical variational autoencoder admits an unsupervised disentangling score correlated with several disentanglement metrics. Hence, to the best of our knowledge, we present the first disentangling model selection based on unsupervised scores.

5.4.2 Regularizing the Total Correlation

State-of-the-art unsupervised disentanglement methods enrich the Gaussian ELBO with various regularizers encouraging disentangling properties. Kim and Mnih [44] and Chen et al. [14] penalize the *total correlation* (TC) [114]

$$\mathrm{TC}(\boldsymbol{z}) = D_{\mathrm{KL}}\big(q(\boldsymbol{z}) \parallel \hat{q}(\boldsymbol{z})\big) = \mathbb{E}_{q(\boldsymbol{z})}\left[\log \frac{q(\boldsymbol{z})}{\hat{q}(\boldsymbol{z})}\right] \tag{5.22}$$

where $\hat{q}(\boldsymbol{z}) := \prod_{i=1}^{n} q(z_i)$ to reduce the dependencies between the dimensions of the representation.

The TC quantifies the statistical dependence among multiple variables. It is an extension of the mutual information to more than two variables and is equal to the Kullback-Leibler divergence between the joint distribution of the variables and the product of the marginal distributions [44, 14]. The idea behind regularizing the TC in disentanglement is to minimize the statistical dependencies between the dimensions of the latent representation. In the context of VAEs, this would encourage each latent dimension to capture a unique and independent aspect of the data variance, which leads to a more disentangled latent space. Reducing the TC to zero would imply that the latent dimensions are statistically independent, enabling fully disentangled representations.

Kim & Mnih [44] first sample from $\hat{q}(\boldsymbol{z})$ by randomly shuffling samples from $q(\boldsymbol{z})$ across the batch for each latent dimension [3]. They then utilize the density-ratio trick [71, 101] to estimate the total correlation by training a discriminator D to classify between samples from $q(\boldsymbol{z})$ and $\hat{q}(\boldsymbol{z})$. If $D(\boldsymbol{z})$ estimates the probability that $\boldsymbol{z}$ is a sample from $q(\boldsymbol{z})$ rather than from $\hat{q}(\boldsymbol{z})$, we can approximate

$$\frac{q(\boldsymbol{z})}{\hat{q}(\boldsymbol{z})} \approx \frac{D(\boldsymbol{z})}{\big(1 - D(\boldsymbol{z})\big)}. \tag{5.23}$$

We have shown in Section 5.3 that the representations of our adapted categorical VAE belong to the same vector space as the Gaussian representations. Thus, we can straight-forwardly adopt the same procedure to estimate the total correlation of $q(\bar{z})$ of the D-VAE latent variable. We augment the ELBO of the D-VAE with a total correlation regularizer to obtain the learning objective

$$\mathcal{L}_{\psi,\phi}(\boldsymbol{x}) - \gamma \mathbb{E}_{q(\boldsymbol{z})}\left[\log \frac{D(\bar{z})}{1 - D(\bar{z})}\right] \tag{5.24}$$

for $\gamma > 0$ and name the corresponding model *FactorDVAE*. Finding new regularizers of the total correlation, which are tailored to the D-VAE could be interesting future work.

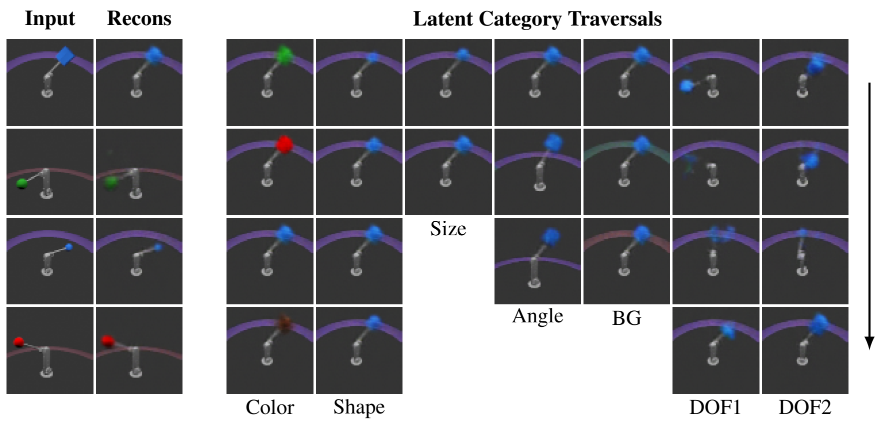


Figure 5.8: Visualization of reconstructions and latent space traversals from the MPI3D dataset [27] of the semi-supervised D-VAE, utilizing masked attentions. The masked attention allows for the incorporation of the number of unique variations, such as two for the object size. We visualize four degrees of freedom (DOF), selected equidistantly from the total of 40. **Left:** The reconstructions are easily recognizable, albeit with blurry details. **Right:** The object color, size, camera angle, and background color (BG) are visually disentangled. The object shape and the DOF factors remain partially entangled.

5.4.3 Semi-Supervised Training

The idea of semi-supervised disentanglement is that incorporating label information of a limited amount of annotated data points during training encourages a latent space with desirable structure with respect to the ground-truth factors of variation [61]. The supervision is incorporated by enriching the ELBO with a regularizer $R_s(r(\boldsymbol{x}), \boldsymbol{z})$, where R_s is a function of the annotated observation-label pairs. Locatello et al. [61] normalize the targets z_i to $[0, 1]$ and propose the binary cross-entropy loss (BCE) or the L_2 loss for R_s. In contrast, we discretize $\boldsymbol{z}$ by binning each dimension z_i into m bins and utilize the cross-entropy loss for R_s obtaining the learning objective

$$\mathcal{L}_{\psi,\phi}(\boldsymbol{x}) + \omega \sum_{i=1}^{n} z_i^j \log \frac{\theta_i^j}{\sum_{k=1}^{m} \theta_i^k} \tag{5.25}$$

where $\omega > 0$ and $z_i^j = 1$ if z_i is in bin j and $z_i^j = 0$ otherwise.

In order to utilize semi-supervised training, a set of data points needs to be annotated beforehand. Different ground-truth factors of variation usually have a specific finite *number of unique values* they can take on, see Table A.3 in Appendix A. It is unclear how to incorporate the knowledge about the number of unique variations in the Gaussian VAE. Thus, previous work dismisses this information entirely [61]. In contrast, it is straightforward to implement this information in the D-VAE using masked attention as introduced for the transformer architecture [112].

If we know that factor z_i can assume a total of $m' < m$ distinct values, we set the set of the m' active categories to be $J_i = \{1 + \lfloor j \frac{m-1}{m'-1} \rceil\}_{j=0}^{m'-1} \subseteq [m]$ and set $\theta_i^j = 0$ for all $j \notin J_i$. The use of masked attention in the D-VAE can be seen as a form of negative constraint that is not straightforwardly replicable in the continuous case.

Being able to include the knowledge of the number of unique variations offers unique advantages, particularly in the context of disentanglement and interpretability. First, incorporating this information allows the model to better align the latent dimensions with the actual ground-truth factors of variation. If we know in advance that a particular factor can only take on a finite set of distinct values, then we can design our model accordingly, such that each unique variation corresponds to a specific value or range in the latent dimension. This facilitates easier and more intuitive manipulation of the generated samples as illustrated in Figure 5.8.

Second, this ability can also lead to better interpretability of the learned latent space, a key goal in disentanglement. By incorporating the knowledge of unique variations, we can help ensure that each unique variation corresponds to a distinct point or region in the latent space, thereby enhancing its interpretability.

5.4.4 The Straight-Through Gap

In Section 5.3, we have seen that a discrete latent space yields favorable disentangling properties. The latent space approximates a discrete representation while its classes maintain the discussed order and the property of having neighborhoods. To further encourage the parameters $\boldsymbol{\theta}$ to become increasingly more discrete, we utilize Temperature Matching as introduced in Section 5.2, i.e., increasing the Gumbel scale parameter during training, which also has positive side-effects on the training dynamics.

We have observed that sometimes the models approach local minima, for which $\boldsymbol{z}$ is not entirely discrete. As per the previous discussion, those models have inferior disentangling properties. We leverage this property by selecting models that yield discrete latent spaces. Similar to the Straight-Through Estimator [8], we round $\boldsymbol{z}$ off using $\arg\max$ and measure the difference between the rounded and original ELBO, i.e.,

$$\mathrm{Gap}_{ST}(\boldsymbol{x}) = |\mathcal{L}_{\psi,\phi}^{ST}(\boldsymbol{x}) - \mathcal{L}_{\psi,\phi}(\boldsymbol{x})|, \tag{5.26}$$

which equals zero if $\boldsymbol{z}$ is discrete. Similar candidates for the measurement could be explored in future work. Figure 6.1 (left) illustrates the Spearman rank correlation between Gap_{ST} and various disentangling metrics on different datasets. A smaller Gap_{ST} value indicates high disentangling scores for most datasets and metrics.

5.4.5 Conclusion

In this section, we have explored several methods to improve the efficiency of common discrete representations. We first introduced the Total Correlation (TC) regularizer, a method adapted from the Gaussian VAE. Applying this regularizer in the discrete context helps ensure that each latent dimension captures a unique and independent aspect of data variance, which leads to a more disentangled latent space.

Next, we explored the advantages of semi-supervised training in enhancing discrete representations. Incorporating the knowledge of the number of unique variations of ground-truth factors offers benefits such as the interpretability of the learned latent space compared to the Gaussian approach. Finally, we addressed the straight-through gap, allowing us to select models that yield the most discrete latent spaces and hence, improve the disentangling properties.

Further study might explore more refined TC regularizers tailored to categorical VAEs or investigate additional candidates for the straight-through gap measure. In Section 6.2, we will show empirically how these improvements to the categorical VAE not only improve disentangled representations over their Gaussian counterpart but also beat state-of-the-art unsupervised disentanglement methods based on Gaussian VAEs.

Chapter 6

Experimental Evaluation

This chapter examines and validates the methods proposed in the preceding chapters analyzing our contributions through extensive experiments and evaluations.

In Section 6.1, we analyze the challenges of training discrete representations. We evaluate the effectiveness of our proposed dropout residual connections and temperature matching methods for enhancing the training of discrete models. Our examination covers multiple stochastic computation graph instances, such as unsupervised parsing of ListOps, multi-hop reasoning over Knowledge Graphs, end-to-end learning of MNIST addition, and the capacity of VS-GAE to generate neural architectures. Our goals are to demonstrate how these models perform in typical application domains, evaluate the potential improvements achieved through our proposed methods, and compare the performance of our discrete-continuous models with state-of-the-art models that lack stochastic components.

Section 6.2 focuses on the benefits of discrete representations. We analyze various aspects of discrete Variational Autoencoder (VAE) models for unsupervised disentanglement. The aim is to understand the advantages of discrete VAE models over Gaussian VAEs, the potential of incorporating label information to enhance discrete representations, and the interpretability of the latent space through visualizations.

In these investigations, we aim to demonstrate the generalization properties, and improved interpretability of our models across a variety of tasks. We believe that the enhancements we have made through dropout residual connections and temperature matching enable the training of models based on discrete representations, which improve over their Gaussian and non-stochastic counterparts.

6.1 Challenges in Training Discrete Representations

This section aims to examine the challenges associated with the training of discrete representations. We believe that by using our proposed approach of dropout residual connections and temperature matching from Section 5.2 and supervised learning of the stochastic nodes from Section 5.1, we can effectively enhance the training of such models. The context of the experiments is threefold.

First, we want to analyze the behavior of complex stochastic computations graphs arising in typical application domains, such as multi-hop reasoning in knowledge graphs and unsupervised parsing of lists of operations. Second, we want to evaluate the behavior of the stochastic computation graphs when incorporating the proposed methods (dropout residual connections, temperature matching, and supervision on the stochastic nodes) to improve the training behavior. Third, we want to compare the resulting discrete-continuous models with state-of-the-art models which do *not* have stochastic components. Here, we are especially interested in analyzing the generalization (extrapolation) behavior of the models under consideration. We present the full implementation details in Appendix A.

The first set of experiments focuses on unsupervised parsing of ListOps [70], a complex dataset containing a list of operations in prefix arithmetic syntax and its unique numerical solution. The objective is to understand how our proposed methods compare to existing stochastic softmax tricks [77] for estimating gradients.

Secondly, we analyze multi-hop reasoning over knowledge graphs (KGs). We consider this a typical complex stochastic computation, and we aim to investigate how our methods can be beneficial in dealing with multi-hop (path) queries in KGs [34]. As part of this experiment, we will evaluate our methods on a newly created dataset based on FB15K-237 [105].

Additionally, we aim to understand the advantages of our proposed method compared to state-of-the-art models that either lack stochastic components or need a hard-coded logic program for the MNIST addition task [66]. On the one hand, our model learns the structure of addition fully in a data-driven manner. On the other hand, we learn discrete representations and, thus, obtain interpretable intermediate outputs for the single MNIST inputs.

Finally, we will explore the behavior of discrete-continuous models whose stochastic nodes are learned in a supervised way. We analyze the ability of VS-GAE to generate neural architectures. Our evaluations will be based on the NAS-Bench-101 dataset [123], aiming to understand the model's capacity to imitate graph generation in an unsupervised manner during test time.

The content of this section primarily draws upon the following publications:

> Friede, D., & Niepert, M. (2021). *Efficient Learning of Discrete-Continuous Computation Graphs*. Advances in Neural Information Processing Systems, 34, 6720-6732.
>
> Friede, D., Lukasik, J., Stuckenschmidt, H., & Keuper, M. (2019). *A variational-sequential graph autoencoder for neural architecture performance prediction*. arXiv preprint arXiv:1912.05317.
>
> Lukasik, J., Friede, D., Stuckenschmidt, H., & Keuper, M. (2020). *Neural architecture performance prediction using graph neural networks*. In DAGM GCPR (pp. 188-201). Springer, Cham.

6.1.1 Unsupervised Parsing on ListOps

We investigate how our method compares to using stochastic softmax tricks as introduced by Paulus et al. [77] for estimating gradients on the simplified variant of the ListOps dataset [70]. The Listops dataset contains sequences in prefix arithmetic syntax such as $\max[\, 2\; 9 \min[\, 4\; 7\,]\; 0\,]$ and its unique numerical solutions (here: 9) [70]. Prior work adapted and used this dataset to evaluate the performance of stochastic softmax tricks [77]. Following this prior work, we first encode the sequence into a directed acyclic graph and then run a graph neural network (GNN) on that graph to compute the solution. Since the resulting dataset was not published, we generated a dataset following their description. In addition to examples of depth $1 \leq d \leq 5$, we further generated test examples of depth $d = 8, 10$ for the extrapolation experiments. More precisely, we used the three operators $\min$, $\max$ and med and capped the maximum length of a sequence at 50. For each depth $d \in \{1, 2, 3, 4, 5\}$ we generated $20,000$ samples for the training set and $2,000$ samples each for the validation set and test set. In order to evaluate the generalization behavior, we further generated $2,000$ test samples for each depth $d \in \{8, 10\}$. All other settings are those of the original code [70].

The arithmetic syntax of each training example induces a directed rooted in-tree, from now on called arborescence, which is the tree along which the message-passing steps of a GNN operate. For instance, in the example mentioned above, there is an edge from each token 2, 9, $\min[$, 0, and the final $]$ directed to $\max[$ and so on. We use the same bi-LSTM encoder as Paulus et al. [77] to compute the logits of all possible edges and the same directed GNN architecture. The original model uses a directed version of Kirchoff's matrix-tree theorem as introduced by Koo et al. [48] to induce the arborescence prior. We simplify this idea by taking, for each node, the Gumbel-softmax over all possible parent nodes. Here we make use

Table 6.1: The results for unsupervised parsing on ListOps [70]. Results taken from Paulus et al. [77] are marked with an asterisk (*) and are based on a different (unpublished) dataset. Column *Inter. acc.* represents the accuracy evaluated on the interpretable intermediate representations. Our discrete models extrapolate much better to depths not seen during training, especially for the task trained on the ground truth graphs (GT) instead of on the sequence.

				Task acc. (extrapolation)	
Model	Task acc.	Edge prec.	Inter. acc.	$d = 8$	$d = 10$
Und.*	$91.2 \pm 1.8^*$	$33.1 \pm 2.9^*$	-	n.a.	n.a.
Arb.*	$95.0 \pm 3.0^*$	$75.0 \pm 7.0^*$	-	n.a.	n.a.
LSTM	91.5 ± 0.3	-	-	83.7 ± 2.0	76.9 ± 3.7
Arb., $\tau = 2$	$\mathbf{96.8} \pm 0.3$	77.4 ± 1.8	-	84.3 ± 1.4	75.4 ± 1.7
Ours, $\tau = 1$	96.1 ± 0.4	$\mathbf{82.3} \pm 1.1$	$\mathbf{70.9} \pm 1.2$	$\mathbf{92.6} \pm 1.1$	$\mathbf{86.9} \pm 4.9$
Ours, $\tau = 2$	$\mathbf{96.3} \pm 0.5$	76.8 ± 2.2	62.9 ± 2.4	$\mathbf{92.7} \pm 1.3$	$\mathbf{88.7} \pm 3.3$
Arb. (GT)	98.7 ± 0.1	100.0	-	86.4 ± 1.1	71.0 ± 2.1
Ours (GT)	99.8 ± 0.1	100.0	99.9 ± 0.0	99.8 ± 0.1	99.9 ± 0.1

of the fact that in an arborescece, each non-root node has exactly one parent. Note that this prior is slightly less strict than the arborescence prior. As in Paulus et al. [77], we exclude edges from the final closing bracket] since its edge assignment cannot be learned from the task objective.

In contrast to Paulus et al. [77], where only the edges of the latent graph are modeled with categorical variables, we also modelled the nodes of the latent graphs with discrete-continuous components as illustrated in Figure 2.4. Let $\mathrm{Num} \in \mathbb{R}^{10\times\mathrm{dim}}$ be the embedding layer that maps the 10 numeral tokens to their respective embedding vectors and let $\mathrm{Pred} : \mathbb{R}^{\mathrm{dim}} \to \mathbb{R}^{10}$ be the classification layer that maps the embedding $\boldsymbol{x} \in \mathbb{R}^{\mathrm{dim}}$ of the final output to the logits of the 10 classes: $\mathrm{Pred}(\boldsymbol{x}) := \mathrm{Lin}^{10\times\mathrm{dim}}(\mathrm{ReLU}(\mathrm{Lin}_B^{\mathrm{dim}\times\mathrm{dim}}(\boldsymbol{x})))$. After each message-passing operation of the GNN, we obtain an embedding $\boldsymbol{u} \in \mathbb{R}^{\mathrm{dim}}$ for each node. By choosing $g_{\boldsymbol{w}} := \mathrm{Pred}$, we obtain the 10 class logits $\boldsymbol{\theta} = g_{\boldsymbol{w}}(\boldsymbol{u}) = \mathrm{Pred}(\boldsymbol{u}) \in \mathbb{R}^{10}$ for the numerals $0, \ldots, 9$. Using the Gumbel-softmax trick with logits $\boldsymbol{\theta}$, scale β, and temperature τ, we obtain $\boldsymbol{z} \in \mathbb{R}^{10}$. We then compute $\boldsymbol{v} = h_{\boldsymbol{w}'}(\boldsymbol{z}) = \boldsymbol{z}^{\intercal}\,\mathrm{Num}$. We apply this on all intermediate node embeddings simultaneously and repeat this after each of the first 4 out of 5 message-passing rounds of the GNN. We use dropout residual connection with increasing dropout probability α. In total, we obtain a model with 5 sets of discrete operations: one set from the graph generation and 4 from the discrete-continuous message-passing steps. At test time, the model per-

Table 6.2: The ablation study for the ListOps task. Temperature matching improves the training behavior of the model. The addition of dropout residual connections is crucial for efficient learning.

				Task acc. (extrapolation)	
Model	Task acc.	Edge prec.	Inter. acc.	$d = 8$	$d = 10$
Ours, $\tau = 1$	96.1 ± 0.4	82.3 ± 1.1	70.9 ± 1.2	92.6 ± 1.1	86.9 ± 4.9
(-) TM	95.1 ± 1.7	76.9 ± 12.1	66.3 ± 9.2	90.5 ± 2.4	83.7 ± 4.0
(-) DR	74.8 ± 11.5	43.9 ± 30.3	29.4 ± 18.6	70.5 ± 9.3	66.7 ± 8.8
(-) DR, TM	56.2 ± 5.0	15.1 ± 7.7	12.7 ± 3.2	53.1 ± 6.6	49.2 ± 7.1

forms discrete and interpretable operations. Figure 2.4 depicts an example stochastic computation graph for the task. We used the same LSTM as in Paulus et al. [77] and a re-implemented version of their model with our customized prior. For ablation purposes, we run all our models with and without dropout residuals and temperature matching, respectively. In the models that use discretization, we slightly modify the pre-message from node j to node i of the GNN's message-passing from $[x_i, x_j]$ to $[x_{\text{emb}}, x_j]$. Since all nodes become intermediate results by construction after the first discretization step, operation nodes would lose the information about their operators otherwise. We also tested this modification on the baselines without discretization but noticed a decline in performance. All models are run for 100 epochs with a learning rate of 0.005 and we select $\tau \in \{1, 2, 4\}$. We keep all other hyperparameters as in Paulus et al. [77]. We evaluate the task accuracy, the edge precision (using the correct graphs), the task accuracy on intermediate operations, and two extrapolation tasks. For the evaluation of the extrapolation task with depth d, we run the GNN d rounds instead of 5.

Table 6.1 compares the proposed models with state of the art approaches. The best performing models are those with discrete-continuous components trained with dropout residuals and temperature matching. While our models do not improve the task accuracy itself, they exceed state of the art methods on all other metrics. Most noticeable is the improved generalization behavior: our models extrapolate much better to expressions with a depth not seen during training. This becomes even more visible when trained on the ground truth graph (GT) directly (instead of on the sequence) where the generalization accuracy is close to 100%. Our model also achieves the best performance on the graph structure recovery with a precision of 82.3%. The mismatch between the task accuracy and the edge precision is mainly due to examples for which the correct latent graph is not required to solve the problem, e.g., for the task $\max[\,2\max[\,4\;7\,]]$ it does not matter whether the

Table 6.3: The results of the path query task on the newly created path query dataset based on FB15K-237. Our model that discretizes the score function of the base model performs much better on the path query task and generalizes perfectly to paths up to length 10.

	MRR					MRR (etrapolation)				
Model	1	2	3	4	5	6	7	8	9	10
ComplEx	**32.9**	30.6	24.7	20.8	16.5	15.1	12.2	10.5	8.8	7.1
ComplEx-C	31.3	35.2	36.9	37.8	35.2	31.9	25.7	19.4	14.1	9.7
Ours, $\tau = 4$	26.8 ± 0.25	**52.3** ± 0.39	**48.8** ± 0.08	**51.4** ± 0.47	**52.6** ± 0.10	**54.2** ± 0.26	**54.8** ± 0.58	**54.7** ± 1.37	**54.9** ± 1.20	**54.7** ± 1.03

7 points to the first or the second $\max[$. Note that our method performs discrete operations at test time which are therefore verifiable and more interpretable. Our model generates intermediate representations that can be evaluated in exactly the same way as the final outputs, see Column *Inter. acc.* in Table 6.1.

The ablation study in Table 6.2 further highlights the challenge of learning discrete-continuous computation graphs with multiple sequential Gumbel-softmax components. It is entirely impossible to train the model without dropout residuals and temperature matching. The analysis in Section 4.1.2 reveal the reason. As illustrated in Figure 5.5, sequential discrete distributions in the computation graph cause vanishing gradients. Temperature matching stabilizes learning somewhat but to avoid vanishing gradients entirely, the use of dropout residuals is necessary.

6.1.2 Multi-Hop Reasoning over Knowledge Graphs

Here we consider the problem of answering multi-hop (path) queries in knowledge graphs (KGs) [34]. The objective is to find the correct object given the subject and a sequence of relations (the path) without knowing the intermediate entities. While traditional knowledge graph completion models like Rescal, [72], DistMult, [120], and ComplEx, [107] show competitive results for the link prediction task, [92], they seem less suited to answer path queries, see Guu et al. [34]. The problem generalizes the standard and popular knowledge graph completion problem. latter published two datasets for the path query answering task based on the subsets of WordNet and Freebase from Socher et al. [100]. To overcome the test leakage problems found in other subsets of Freebase, Toutanova and Chen [105] introduced the FB15K-237 dataset, which has become the most common dataset to evaluate link prediction. We evaluate our approach on a newly created dataset based on

Table 6.4: Results for the path query benchmark [34]. Our proposed model performs significantly better than the baselines KGC models which do not have stochastic components (labeled with -C).

	WordNet		Freebase	
Model	MQ	H@10	MQ	H@10
Bilinear-C [34]	89.4	54.3	83.5	42.1
DistMult-C [34]	90.4	31.1	84.8	38.6
TransE-C [34]	93.3	43.5	88.0	50.5
Path-RNN [18]	**98.9**	–	–	–
ROP [122]	–	–	90.7	56.7
CoKE [113]	94.2	**67.9**	**95.0**	**77.7**
Ours, $\tau = 4$	94.4 ± 0.0	64.3 ± 0.11	89.6 ± 0.46	68.7 ± 0.32

FB15K-237. Particularly, we built the graph consisting of all training triples from the original dataset and sampled a starting entity uniformly. We then sampled an incident relation uniformly and sampled the next entity uniformly from all entities reachable via this relation. Continuing this way, we created a dataset of $272\,115$ training paths of length (number of relations) $2, 3, 4, 5$, respectively. We repeated this procedure on the graph consisting of all triples (not only training triples) and removed all duplicates to create $17\,535$ validation paths of length $2, 3, 4, 5$, and $20\,466$ test paths of length $2, \ldots, 10$. Finally, we added all of the FB15K-237 triples to the dataset as paths of length 1. For comparison with state-of-the-art approaches, we further evaluate various approaches on the standard benchmarks for path queries [34].

We model each application of a relation type to an intermediate entity as a discrete-continuous component that discretely maps to one of the KG entities. The discrete distribution $p(\boldsymbol{z};\boldsymbol{\theta})$, therefore, has as many categories as there are entities in the KG. Let $\mathrm{score}(\boldsymbol{s}, \boldsymbol{r}, \boldsymbol{o}) : \mathbb{R}^{3\times\mathrm{dim}} \to \mathbb{R}$ be a scoring function that maps the embeddings of a triple to the logit (unnormalized probability) of the triple being true. Given the current (intermediate) entity embedding $\boldsymbol{s}$ and relation type embedding $\boldsymbol{r}$ we compute the logits $\boldsymbol{\theta}$ for all possible entities. Hence, the function g computes the logits for all possible entities. Using the Gumbel-softmax trick with parameter $\boldsymbol{\theta}$, scale β, and temperature τ, we obtain the sample $\boldsymbol{z} \in \mathbb{R}^n$. The function h now computes $\boldsymbol{z}^\intercal\mathbf{E}$ where $\mathbf{E}$ be the matrix whose rows are the entity embeddings. We use dropout residual connection between the input and output vectors of the discrete-continuous component during training. Note that we do not increase the

Table 6.5: Results for the extrapolation benchmark [113] for the path query dataset [34]. Our model achieves best results on 10 out of 16 setups. The advantage of discretizing is most visible when trained on short paths.

		WordNet		Freebase	
Paths	Model	MQ	H@10	MQ	H@10
≤ 4	CoKE	93.6	$\mathbf{65.5}$	$\mathbf{93.1}$	$\mathbf{71.2}$
	Ours	$\mathbf{94.3} \pm 0.14$	64.7 ± 0.11	88.8 ± 1.29	69.4 ± 0.72
≤ 3	CoKE	92.6	$\mathbf{65.0}$	$\mathbf{90.6}$	64.6
	Ours	$\mathbf{93.4} \pm 0.04$	$\mathbf{65.0} \pm 0.11$	88.3 ± 1.85	$\mathbf{68.5} \pm 0.46$
≤ 2	CoKE	$\mathbf{90.8}$	49.5	$\mathbf{89.4}$	59.5
	Ours	90.2 ± 0.04	$\mathbf{62.0} \pm 0.04$	87.6 ± 0.62	$\mathbf{68.6} \pm 0.36$
≤ 1	CoKE	73.5	16.7	72.7	37.3
	Ours	$\mathbf{81.1} \pm 0.22$	$\mathbf{52.2} \pm 0.17$	$\mathbf{82.1} \pm 0.54$	$\mathbf{63.9} \pm 0.6$

number of parameters compared to the base scoring models. We use ComplEx [107] as the scoring function.

For the FB15K-237 experiments, we use a slight variation of our model for Freebase and Wordnet to optimize the training in both directions of the path. As baselines, we take the same model without the discretization module and train it first on triples only (ComplEx) and then on all paths (ComplEx-C). We validate the models using filtered MRR (Bordes et al. [9]) on all validation paths and report test results on all paths for each length individually. Table 6.3 lists the results of the experiments on the new FB15K-237 based dataset. The models based on discrete-continuous components achieve improvements of up to 49% compared to the baselines. Even more pronounced is the ability of these models to generalize: the accuracy does not drop even when tested on paths twice the length of those seen during training. The performance gap between 1 relation and 2 relations is mainly due to the fact that the test paths of length 1 purely consist of triples from the test graph while all other paths consist of all triples from the full knowledge graph.

We compare our best performing model on the original dataset from Guu et al. [34] with their proposed evaluation protocol and baselines against RNN models, Path-RNN [18], ROP [122] and the state-of-the-art transformer model CoKE [113] for the standard task as well as in an extrapolation setting.

Table 6.4 lists the results for the proposed model in comparison to several base-

Table 6.6: A comparison of our proposed method with DeepProbLog and a CNN baseline. Compared to DeepProbLog, our model does not need a hard-coded logic program. Instead, it learns addition in a data-driven manner.

Model	Task Acc.	Interpretable	Discrete	Learned Structure
Baseline [66]	89.14 ± 1.22	No	No	No
DeepProbLog [66]	98.17 ± 0.20	Yes	Yes	No
Ours, $\tau = 8$	98.10 ± 0.15	Yes	Yes	Yes

lines. Our proposed model performs significantly better than the baselines KGC models which do not have stochastic components (labeled with -C). On WordNet, our model can even keep pace with the state of the art transformer model CoKE.

Similar to the results on ListOps, the discrete-continuous models have the strongest generalization performance (see Table 6.5). Here, we use the same test set as before but reduce the length of the paths seen during training. Even when trained only on paths of length ≤ 3, our model performs better on the path query task than most baselines trained on paths of all lengths.

6.1.3 End-to-End Learning of MNIST Addition

The MNIST addition task addresses the learning problem of simultaneously (i) recognizing digits from images and (ii) performing the addition operation on the digit's numerical values [66]. The dataset was introduced with DeepProbLog, a system that combines neural elements with a probabilistic logic programming language. In DeepProbLog the program that adds two numbers is provided. We adopt their approach of using a convolutional neural network (CNN) to encode the MNIST images. Contrary to their approach, we learn the operation without any prior knowledge about adding numbers in a data-driven manner.

Our proposed model is trained without a given logic program. The addition operation is modeled as $\mathrm{Add}(\boldsymbol{x}, \boldsymbol{y}) = \mathrm{Lin}_B^{\dim \times 2\dim}(\mathrm{ReLU}(\mathrm{Lin}_B^{2\dim \times 2\dim}([\boldsymbol{x}; \boldsymbol{y}])))$ and we use a single linear layer to compute the logits. Given $\boldsymbol{u} \in \mathbb{R}^{\dim}$, the encoding of the digit obained from the CNN, the function g computes $\boldsymbol{\theta} = \mathrm{Lin}(\boldsymbol{u})$, that is, the 19 logits for all possible numbers. Using the Gumbel-softmax trick with parameter $\boldsymbol{\theta}$, scale β, and temperature τ, we obtain $\boldsymbol{z} \in \mathbb{R}^{19}$. The function h now computes $\boldsymbol{v} = \boldsymbol{z}^{\intercal}\,\mathrm{P} \in \mathbb{R}^{\dim}$ where P is the weight matrix of Lin. We discretize the two outputs of the CNN, execute the addition layer and use the single linear layer P for the classification. This results into a model that has in total fewer parameters than the baseline in [66].

Table 6.6 lists the results on this task. Our proposed model, even though trained without a given logic program and in a data-driven manner, achieves the same accuracy as DeepProbLog. Since we use hard sampling during test time, we can discretely read out the outputs of the internal CNN layer.

6.1.4 VS-GAE's Ability to Generate Neural Architectures

In this section, we evaluate VS-GAE through reconstruction ability and valid generation of neural architectures based on NAS-Bench-101 [123]. At training time, the model learns the graph generation in a supervised manner. This experiment evaluates the model's ability to imitate the graph generation without supervision during test time.

NAS-Bench-101 [123] is a public dataset of neural architectures in a restricted cell structured search space [128] evaluated on the CIFAR-10-classification set [49]. NAS-Bench-101 considers the following constraints to limit the search space: it only considers directed acyclic graphs, the number of nodes is limited to $|V| \leq 7$, the number of edges is limited to $|E| \leq 9$ and only 3 different operations are allowed $\{3 \times 3 \text{ convolution}, 1 \times 1 \text{ convolution}, 3 \times 3 \text{ max} - \text{pool}\}$. These restrictions lead to a total of 423k unique convolutional architectures. The architectures have been trained for four increasing numbers of epochs $\{4, 12, 36, 108\}$. Each of these architectures is mapped to its validation and training measures. In this experiment we use the architectures trained for 108 epochs.

To evaluate VS-GAE by means of reconstruction ability and valid generation of neural architectures, we train the VS-GAE on 90% of the dataset and test it on the 10% held-out data.

We first measure the reconstruction accuracy which describes how often our model can reconstruct the input graphs of the test set perfectly. For this purpose, after calculating the mean h_G and the variance h_G^{var} of the approximated posterior $q_\phi(\mathbf{z}|G)$ for the test set, we sample **z** from the latent representation of each input graph 10 times and decode each sample again 10 times. The average portion of the decoded graphs that are identical to the input ones is then reported as the reconstruction accuracy.

The second ability we are interested in is the prior validity which quantifies how often our model is able to generate valid graphs from the VS-GAE prior distribution. Following Zhang et al. [126], we sample $1,000$ vectors from the latent space with prior distribution $p(\mathbf{z})$ and decode each vector 10 times. The average portion of the decoded graphs that are valid is then reported as the prior validity. For a valid graph by means of the NAS-Bench 101 [123] search space, it has to pass the

Table 6.7: The reconstruction accuracy and prior validity of VS-GAE in %.

Method	Accuracy	Validity
VS-GAE	99.99	95.09

following validity checks: 1) exactly one starting point, i.e., the input node, 2) exactly one ending point, i.e., the output node, 3) there exist no nodes which do not have any predecessors, except for the input node, 4) there exist no nodes which do not have any successors, except for the output node, 5) the graphs are DAGs.

See Table 6.7 for the evaluation results. Our model, VS-GAE, shows a nearly perfect reconstruction accuracy with a high prior validity. Thus, VS-GAE represents a strong tool for further downstream tasks like sampling or neural architecture generation.

6.1.5 Conclusion

We demonstrated the role of our proposed approaches in addressing challenges encountered when training discrete representations. The integration of dropout residual connections and temperature matching has effectively proven to enhance the overall performance and robustness of such models.

Our analysis covers multiple stochastic computation graph instances, featuring varied complex structures. Starting with unsupervised parsing on ListOps, we observe substantial improvement in generalization ability when operating on unseen parsing length. Although the task accuracy was not notably enhanced, the models excelled in graph structure recovery, generalization, and interpretability.

Furthermore, the introduced variations of the model led to an impressive performance gain in the generalization abilities in multi-hop reasoning over knowledge graphs. This ability to perform well even on paths twice the length of those encountered during training further underlines the robustness of our approach.

The MNIST addition task demonstrated the model's capability to combine image recognition and addition operation in a data-driven manner without prior knowledge. Despite the absence of a given logic program, our model matched the accuracy of DeepProbLog, the standard in this domain, while having fewer parameters. This illustrates the model's efficiency and adaptability.

Finally, we evaluated VS-GAE's ability to generate neural architectures based on NAS-Bench-101. The model was trained in a supervised manner, but was tested in an unsupervised setting, showcasing its strong performance in reconstruction and validity of generated neural architectures.

The demonstrated generalization and improved interpretability of our models across a variety of tasks, highlight the advantages of our proposed methods. The enhancements made through dropout residual connections and temperature matching have demonstrated the potential for significant improvements in the training and performance of models based on discrete representations.

6.2 Understanding the Benefits of Discrete Representations

We present a comprehensive analysis of various aspects of discrete VAE models for unsupervised disentanglement as introduced in Sections 5.3 and 5.4. First, we investigate whether a discrete VAE offers advantages over Gaussian VAEs in terms of disentanglement properties, finding that the discrete model generally outperforms its Gaussian counterpart and showing that the FactorDVAE achieves new state-of-the-art MIG scores on most datasets. Next, we explore the usefulness of different disentanglement metrics for downstream tasks, revealing that the MIG score is the most reliable indicator of sample efficiency across different datasets. Additionally, we propose a model selection criterion based on Gap_{ST} to find good discrete models solely using unsupervised scores. Furthermore, we examine how incorporating label information can further enhance discrete representations, discovering that the unregularized discrete method outperforms other methods in the semi-supervised disentanglement task, especially when more labels are available. Lastly, we provide visualizations of the latent categories using a unique approach that allows straightforward traversal in the discrete latent space. This approach leads to better interpretability of the latent space as the choice of the values of different factors of variation becomes intuitive.

The content of this section primarily draws upon the following publication:

> Friede, D., Reimers, C., Stuckenschmidt, H., & Niepert, M. (2023). *Learning Disentangled Discrete Representations*. Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD

In the following, we give a short summary of the experimental setup.

Methods The experiments aim to compare the Gaussian VAE with the discrete VAE. For both VAEs, we consider the unregularized version and the total correlation penalizing method, VAE, D-VAE, FactorVAE [44] and FactorDVAE, see

Equation (5.24). For the semi-supervised experiments, we augment each loss function with the supervised regularizer R_s as in Section 5.4.3. For the Gaussian VAE, we choose the BCE and the L_2 loss for R_s, respectively. For the discrete VAE, we select the cross-entropy loss, once without and once with masked attention where we incorporate the knowledge about the number of unique variations.

Datasets We consider six commonly used disentanglement datasets which offer explicit access to the ground-truth factors of variation: *dSprites* [38], *C-dSprites* [59], *SmallNORB* [54], *Cars3D* [86], *Shapes3D* [44] and *MPI3D* [27]. All datasets are rendered in images of size 64×64 and normalized to $[0, 1]$. As in [59], we directly sample from the generative model, effectively avoiding overfitting. We consider gray-scale datasets dSprites, SmallNORB, and Circles, as well as datasets with three color channels C-dSprites, Cars3D, Shapes3D, and MPI3D. We followed the instructions from [116] to create the Circles dataset utilizing the Spriteworld environment [115], setting the size to 0.2. Table A.3 in Appendix A contains a set of all ground-truth factors of variation for each dataset.

Metrics We consider the commonly used disentanglement metrics that have been discussed in detail in [59] to evaluate the representations: *BetaVAE* metric [38], *FactorVAE* metric [44], *Mutual Information Gap* (MIG) [14], *DCI Disentanglement* (DCI) [23], *Modularity* [88] and *SAP score* (SAP) [52]. As illustrated on the right side of Figure 6.1, the MIG score seems to be the most reliable indicator of *statistical sample efficiency*, that is, the downstream task accuracy based on 100 training samples divided by the accuracy based on $10\ 000$ samples, across different datasets. Therefore, we primarily focus on the MIG disentanglement score. We discuss this in more detail in Section 6.2.2.

Experimental Protocol We adopt the experimental setup of prior work ([59] and [61]) for the unsupervised and for the semi-supervised experiments, respectively. Specifically, we utilize the same neural architecture for all methods so that all differences solely emerge from the distribution of the type of VAE. For both total correlation penalizing methods, we choose 6 values for the hyperparameter γ from Equation (5.24) in Section 5.4. For the unsupervised case, we run each considered method on each dataset for 50 different random seeds. Since the two unregularized methods do not have any extra hyperparameters, we run them for 300 different random seeds instead. For the semi-supervised case, we consider two numbers (100/1000) of perfectly labeled examples and split the labeled examples (90%/10%) into a training and validation set. The model cohorts of the total correlation penalizing methods consist of the 6 hyperparameters γ as well as

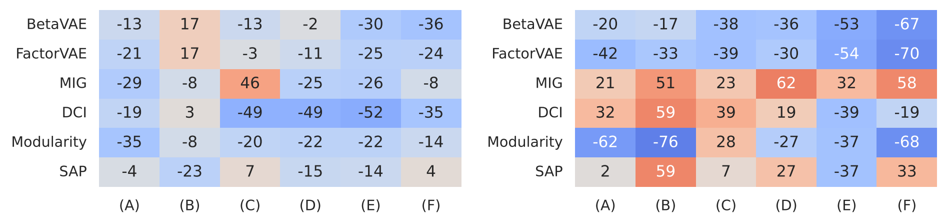
Table 6.8: The median MIG scores in % for state-of-the-art unsupervised methods compared to the discrete methods. Results taken from [59] are marked with an asterisk (*). We have re-implemented all other results with the same architecture as in [59] for the sake of fairness. The last row depicts the scores of the models selected by the smallest Gap_{ST}. The 25% and the 75% quantiles are depicted in the squared brackets below the median values.

Model	dSprites	C-dSprites	SmallNORB	Cars3D	Shapes3D	MPI3D
β-VAE [38]	11.3* [7.5,15.8]	12.5* [9.7,14.6]	20.2* [19.1,22.8]	9.5* [5.6,11.7]	n.a.	n.a.
β-TCVAE [14]	17.6* [13.6,22.2]	14.6* [10.4,18.0]	21.5* [18.3,24.5]	12.0* [7.3,14.0]	n.a.	n.a.
DIP-VAE-I [52]	3.6* [1.9,9.4]	4.7* [2.4,9.0]	16.7* [8.5,20.9]	5.3* [3.4,7.2]	n.a.	n.a.
DIP-VAE-II [52]	6.2* [3.6,8.6]	4.9* [3.2,7.9]	24.1* [22.4,25.4]	4.2* [2.7,6.4]	n.a.	n.a.
Ann.VAE [11]	7.8* [2.9,20.9]	10.7* [4.8,25.7]	4.6* [1.5,8.1]	6.7* [4.6,7.7]	n.a.	n.a.
FactorVAE [44]	17.4 [12.6,26.3]	14.3 [11.7,20.9]	**25.3** [24.0,26.4]	9.0 [7.2,10.6]	34.7 [27.0,44.3]	11.1 [6.9,31.3]
D-VAE	17.4 [13.2,20.0]	9.4 [5.5,13.4]	19.0 [16.3,21.8]	8.5 [5.8,11.1]	28.8 [21.8,34.2]	12.8 [8.9,16.5]
FactorDVAE	**21.7** [14.5,35.7]	**15.5** [11.3,20.3]	23.2 [20.6,24.8]	**14.9** [12.8,16.3]	**42.4** [34.8,48.3]	**30.5** [26.0,32.1]
Selection	39.5	20.0	22.7	19.1	40.1	32.3

6 hyperparameters ω as in Equation (5.25) and one random seed, while the model cohorts of the unregularized methods consist of 6 hyperparameters ω and 6 random seeds. We use the R_s value computed on the validation data to select the best model of each cohort, respectively. We present the full implementation details in Appendix A.

6.2.1 Improvement in Unsupervised Disentanglement

Comparison of the Unregularized Models In the first experiment, we aim to answer our main research question of whether discrete latent spaces yield structural advantages over their Gaussian counterparts. Figure 6.2 depicts the comparison



	(A)	(B)	(C)	(D)	(E)	(F)
BetaVAE	-13	17	-13	-2	-30	-36
FactorVAE	-21	17	-3	-11	-25	-24
MIG	-29	-8	46	-25	-26	-8
DCI	-19	3	-49	-49	-52	-35
Modularity	-35	-8	-20	-22	-22	-14
SAP	-4	-23	7	-15	-14	4

	(A)	(B)	(C)	(D)	(E)	(F)
BetaVAE	-20	-17	-38	-36	-53	-67
FactorVAE	-42	-33	-39	-30	-54	-70
MIG	21	51	23	62	32	58
DCI	32	59	39	19	-39	-19
Modularity	-62	-76	28	-27	-37	-68
SAP	2	59	7	27	-37	33

Figure 6.1: The Spearman rank correlation between various disentanglment metrics and Gap_{ST} (**left**) and the statistical sample efficiency, i.e., the downstream task accuracy based on 100 samples divided by the one on 10 000 samples (**right**) on different datasets: dSprites (A), C-dSprites (B), SmallNORB (C), Cars3D (D), Shapes3D (E), MPI3D (F). **Left:** Correlation to Gap_{ST} indicates the disentanglement skill. **Right:** Only a high MIG score reliably leads to a higher sample efficiency over all six datasets.

regarding the disentanglement scores (left) and the datasets (right). The discrete model achieves a better score on the MPI3D dataset for each metric with median improvements ranging from 2% for Modularity to 104% for MIG. Furthermore, the discrete model yields a better score for all datasets but SmallNORB with median improvements ranging from 50% on C-dSprites to 336% on dSprites. More detailed results can be found in Figure B.4, and Figure B.5 in Appendix B. Taking into account all datasets and metrics, the discrete VAE improves over its Gaussian counterpart in 31 out of 36 cases.

Comparison of the Total Correlation Regularizing Models For each VAE, we choose the same 6 values of hyperparameter γ for the total correlation penalizing method and train 50 copies, respectively. The right side of Figure 6.3 depicts the comparison of FactorVAE and FactorDVAE w.r.t. the MIG metric. The discrete model achieves a better score for all datasets but SmallNORB with median improvements ranging from 8% on C-dSprites to 175% on MPI3D. These findings indicate that discrete latent spaces indeed yield structural advantages over their Gaussian counterparts.

6.2.2 Choice of the Disentanglement Metric

In this experiment, we want to determine which disentanglement metric indicates a sound discrete latent space with respect to downstream tasks. We follow the simple downstream classification task from [59] of recovering the true factors of variations from the learned representation using either multi-class logistic regression (LR) or

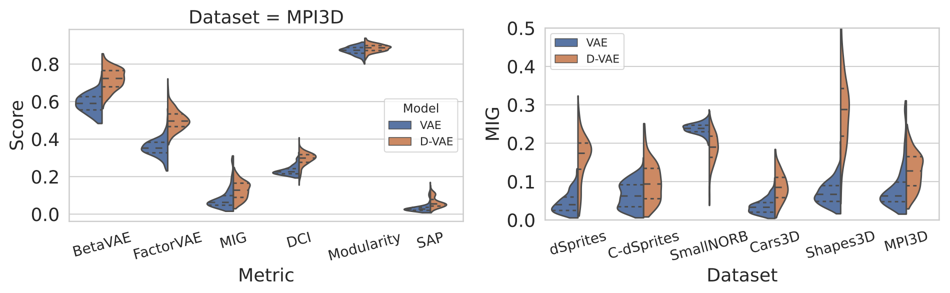


Figure 6.2: Comparison between the unregularized Gaussian VAE and the discrete VAE by kernel density estimates of 300 runs, respectively. **Left:** Comparison on the MPI3D dataset w.r.t. the six disentanglement metrics. The discrete model yields a better score for each metric, with median improvements ranging from 2% for Modularity to 104% for MIG. **Right:** Comparison on all six datasets w.r.t. the MIG metric. With the exception of SmallNORB, the discrete VAE yields a better score for all datasets with improvements of the median score ranging from 50% on C-dSprites to 336% on dSprites.

gradient-boosted trees (GBT). More precisely, we sample training sets of two different sizes, 100 and 10 000, and evaluate the average test accuracy across factors on a test set of size 5 000, respectively. To analyze the sample complexity, we measure the Spearman rank correlation between the different disentanglement metrics and the statistical sample efficiency that is, the test accuracy based on 100 training samples divided by the accuracy based on 10 000 samples. The right side of Figure 6.1 depicts this correlation regarding the LR task for all six datasets. We can observe a high variance of the correlation depending on the selected disentanglement metric. The correlation with the DCI, Modularity, and SAP scores depends on the data, while a high BetaVAE or FactorVAE score even negatively impacts the sample efficiency. Only a high MIG score reliably leads to a higher sample efficiency over all six datasets. The experiments regarding the GBT task in Figure B.1 in Appendix B mostly confirm this finding. Consequently, we are mainly interested in the structural behavior of discrete representations regarding the MIG disentanglement score.

6.2.3 Match State-of-the-Art Disentanglement Methods

Current state-of-the-art unsupervised disentanglement methods enrich Gaussian VAEs with various regularizers encouraging disentangling properties. Table 6.8 depicts the MIG scores of all methods as reported in [59] utilizing the same ar-

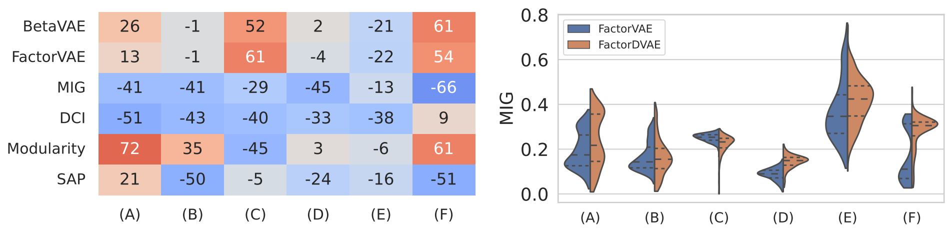


Figure 6.3: Disentangling properties of FactorDVAE on different datasets: dSprites (A), C-dSprites (B), SmallNORB (C), Cars3D (D), Shapes3D (E), MPI3D (F). **Left:** The Spearman rank correlation between various disentangling metrics and Gap_{ST} of D-VAE and FactorDVAE combined. A small Gap_{ST} indicates high disentangling scores for most datasets regarding the MIG, DCI, and SAP metrics. **Right:** A comparison of the total correlation regularizing Gaussian and the discrete model w.r.t. the MIG metric. The discrete model yields a better score for all datasets but SmallNORB with median improvements ranging from 8% on C-dSprites to 175% on MPI3D.

chitecture as us. We have seen in Experiment 6.2.1 that we can also utilize TC regularization to improve the discrete VAE further. While the unregularized discrete VAE yields MIG scores that are on par with the regularized state-of-the-art methods, the TC regularizing discrete method improves these scores heavily. FactorDVAE achieves new state-of-the-art MIG scores on all datasets but SmallNORB, improving the previous best scores by over 17% on average. These findings suggest that incorporating results from the disentanglement literature might lead to even stronger models based on discrete representations.

6.2.4 Unsupervised Model Selection for Disentanglement

A remaining challenge in the disentanglement literature is selecting the hyperparameters and random seeds that lead to good disentanglement scores [61]. The Gaussian VAE yields unsupervised scores like the KL divergence, the reconstruction loss, or the ELBO, which are not helpful for model selection [59]. We showed in Section 4.2.3 that a discrete latent space yields favorable disentangling properties. We propose a model selection based on an unsupervised score measuring the discreteness of the latent space utilizing Gap_{ST} from Section 5.4.4. The left side of Figure 6.3 depicts the Spearman rank correlation between various disentangling metrics and Gap_{ST} of D-VAE and FactorDVAE combined. Note that the unregularized D-VAE model can be identified as a FactorDVAE model with $\gamma = 0$. A small Straight-Through Gap corresponds to high disentangling scores for most

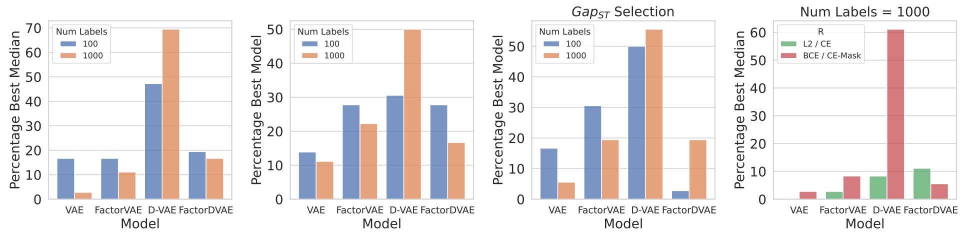


Figure 6.4: The percentage of each semi-supervised method being the best over all datasets and disentanglement metrics for different selection methods: median, lowest R_s, lowest Gap_{ST}, median for 1000 labels. The unregularized discrete method outperforms the other methods in semi-supervised disentanglement task. Utilizing the masked regularizer improves over the unmasked one.

datasets regarding the MIG, DCI, and SAP metrics. This correlation is most vital for the MIG metric. We anticipate finding good hyperparameters by selecting those models yielding the smallest Gap_{ST}. The last row of Table 6.8 confirms this finding. This model selection yields MIG scores that are, on average, 22% better than the median score and not worse than 6%.

6.2.5 Semi-Supervised Disentanglement Methods

Locatello et al. [61] suggest utilizing the semi-supervised regularizer R_s from Equation 5.25 in two different ways. They employ the semi-supervised regularizer R_s by including 90% of the label information during training and utilizing the remaining 10% for a model selection. We also experiment with a model selection based on the Gap_{ST} value. Figure 6.4 depicts the percentage of each semi-supervised method being the best over all datasets and disentanglement metrics. The unregularized discrete method surpasses the other methods on the semi-supervised disentanglement task. In contrast to the unsupervised setting, the D-VAE unexpectedly outperforms FactorDVAE. We hypothesize that the CE regularizer is a strong inductive prior for disentanglement itself and gets diminished by the additional TC regularizer. The advantage of the discrete models is more significant for the median values than for the model selection. This may also explain why the advantage of the discrete models is more evident on the median values since 10% label data might not suffice for the model selection. Utilizing Gap_{ST} for selecting the discrete models only partially mitigates this problem. Incorporating the number of unique variations by utilizing the masked regularizer improves the disentangling properties significantly, showcasing another advantage of the discrete latent space.

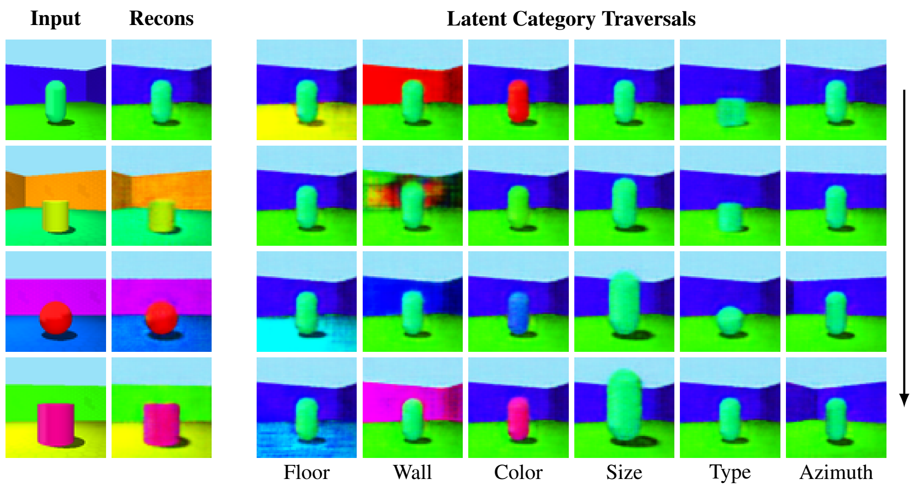


Figure 6.5: Reconstructions and latent space traversals from the Shapes3D dataset [44] of the semi-supervised D-VAE, utilizing masked attentions with the lowest R_s value. The masked attention allows for the incorporation of the number of unique variations, such as four for the object type. For the other factors of variation, We visualize four categories, selected equidistantly. **Left:** The reconstructions are easily recognizable, albeit with blurry details. **Right:** All factors of variation are visually disentangled.

6.2.6 Visualization of the Latent Categories

Prior work uses latent space traversals for qualitative analysis of representations [38, 11, 44, 116]. A latent vector $\boldsymbol{z} \sim q_\phi(\boldsymbol{z}|\boldsymbol{x})$ is sampled, and each dimension z_i is traversed while keeping the other dimensions constant. The traversals are then reconstructed and visualized. Unlike the Gaussian case, the D-VAE's latent space is known beforehand, allowing straightforward traversal along the categories. Knowing the number of unique variations lets us use masked attention to determine the number of each factor's categories, improving latent space interpretability. Figure 5.8 and Figure 6.5 illustrates the reconstructions of four random inputs and latent space traversals of the semi-supervised D-VAE utilizing masked attentions for the the MPI3D dataset [27] and the Shapes3d dataset [44], respectively. While the reconstructions are easily recognizable, their details can be partially blurry, particularly concerning the shapes. Most factors of variation, e.g., the object color, object size, camera angle, and background color are visually disentangled, and their categories can be selected straightforwardly to create targeted observations.

The Shapes3d dataset in Figure 6.5 is visually disentangled. Each categorical distribution belongs to a single factor of variation. For the MPI3D dataset depicted in Figure 5.8, the partially entangled factors, such as object shape and the two-degree-of-freedom (DOF) factors, are precisely those also entangled in the observation space. The observations are two-dimensional projections of a three-dimensional robotic arm, meaning that different settings of the DOF factors might result in very similar or even identical projections. Moreover, the projection of the object shape changes depending on the DOF factors. This finding perfectly aligns with the discussion in Section 5.3, where we proposed that the model maps observations with similar pixel values close together in the latent space and learns a disentangled representation by finding those observations with the smallest difference in the pixel values.

6.2.7 Conclusion

Our experimental evaluation and analysis of discrete VAE models have revealed several key insights into their benefits for unsupervised disentanglement. The results demonstrate that discrete VAE models generally outperform their Gaussian counterparts, and the FactorDVAE achieves new state-of-the-art MIG scores across most datasets. This underlines the structural advantages of discrete latent spaces in representing data with higher disentanglement quality.

Furthermore, incorporating label information can improve discrete representations, with the discrete method demonstrating superior performance in semi-supervised disentanglement, especially when more labels are available. Our findings suggest that integrating the advancements from the disentanglement literature can lead to even more robust models based on discrete representations.

Lastly, the visualization approach we presented allows for intuitive traversal of the discrete latent space, enhancing model interpretability.

Chapter 7

Conclusion

The present thesis, *”Exploring Discrete Representations in Stochastic Computation Graphs: Challenges, Benefits, and Novel Strategies,”* undertakes a comprehensive study of discrete representations with Stochastic Computation Graphs (SCGs). Five research questions guided the study, focusing on the challenges in training models using discrete representations, their potential benefits, and novel strategies to overcome the challenges and maximize the performance and efficiency of these representations.

In Chapter 2, the background for the thesis was established by investigating the role of SCGs in deep learning, including within Variational Autoencoders (VAEs). This investigation further introduced and discussed the concepts of Gumbel-Max and Gumbel-Softmax tricks and disentanglement in VAEs. We then explored prior work in Chapter 3, providing context and demonstrating the relevance of this research.

Our theoretical analysis in Chapter 4 confirmed that the training process with discrete representations presents unique challenges. These challenges stem from inherent complexities in differentiating through stochastic nodes and the complications of the Gumbel-softmax trick, such as the underutilization of categories during training. Despite these challenges, we have shown that discrete representations have distinct structural advantages over their continuous counterparts. Primarily, we found that discrete representations enhance disentanglement, offering a potential explanation for the recent successes of discrete-representations-based methods.

We developed several methods in Chapter 5 to address these challenges and further optimize the use of discrete representations within SCGs. The first method involved an efficient method for learning the structure of SCGs using supervised learning for efficient Neural Architecture Search. Further methods covered the parameter learning of discrete SCGs. To counter common issues such as small gradients and local minima, we proposed altering the scale parameter of Gumbel noise perturbations and implementing dropout residual connections in discrete-continuous computation graphs. We also aimed to enhance disentanglement by substituting standard Gaussian VAEs with a categorical VAE, encouraging the creation of a discrete latent space.

In Chapter 6, we conducted the experimental evaluation of these methods across a diverse range of domains, demonstrating their effectiveness. The experiments proved that the challenges in training discrete representations could be successfully mitigated. Moreover, introducing discrete representations improved the models' interpretability and generalization behavior. Our findings also revealed that the underlying grid structure of categorical distributions mitigates the rotational invariance issue associated with multivariate Gaussian distributions, thus serving as an efficient inductive prior for disentangled representations.

In conclusion, this thesis contributes a robust theoretical understanding, novel methods, and a comprehensive evaluation of discrete representations with SCGs. It presents the unique challenges of discrete representations, their benefits, and strategies for maximizing them. The empirical evaluations validate the proposed methods and strategies, signifying the importance and potential of discrete representations in deep learning. Future research may build on these findings, contributing further to the development and refinement of discrete representations and their applications.

7.1 Key Findings

We offer a comprehensive overview of our research's significant insights on using discrete representations with Stochastic Computation Graphs (SCGs). Our investigation focused on two core sets of research questions, primarily concerned with the challenges, advantages, and potential improvements of discrete representations. Our first research showcased critical insights about discrete representations. In our second research, we explored effective structure and parameter learning strategies for discrete SCGs and further enhancements for discrete representations.

In response to our first research question, **RQ1.1:** *Why is the training of models based on discrete SCGs inherently challenging?*, our analysis, as detailed in Section 4.1, illustrates some of the intrinsic difficulties associated with training models using discrete representations. The main issue arising from such models is the underutilization of the available categories, causing multiple distinct input representations to be mapped to a single category. This creates a challenge because the model cannot effectively distinguish between unique input data.

We have addressed these concerns with the development of Theorem 1, which exposes a crucial constraint on the gradient of a Gumbel-softmax parameter determined by the normalized probability of a category. This theorem clarifies the issues presented during the gradient optimization, which can produce suboptimal training results due to small gradient values. A key implication of this finding is that it provides a better understanding of why discrete representations can be challenging to train and guides us toward developing methods for more efficient training processes in Section 5.2.

To answer our second research question, **RQ1.2:** *Why do discrete representations possess structural advantages over their continuous counterparts?*, our analysis in Section 4.2 explores the structural benefits that discrete representations hold over continuous ones. Analyzing the rotational equivariance demonstrated by Theorem 2, we find that continuous representations in the form of Gaussian distributions can potentially entangle generative factors the model intends to disentangle.

We proposed a discrete variational autoencoder (D-VAE) that models a joint distribution of Gumbel-softmax random variables to address this issue. This D-VAE constructs a discrete grid in the latent space that mitigates the rotational problem, a structural advantage that may account for better generalization behavior in models utilizing discrete representations.

However, it is important to recognize that being robust against rotations alone does not guarantee achieving disentanglement properties. For this, we present further analysis of discrete representations' disentanglement properties in Section 5.3.

Regarding our second set of research questions, our investigations began with **RQ2.1:** *How can we effectively learn the structure and the parameters of discrete SCGs?* The corresponding discussions in Sections 5.1 and 5.2 helped us answer this question. We proposed strategies such as leveraging supervised learning to efficiently learn the structure of SCGs within Neural Architecture Search, as discussed in Section 5.1. However, a limitation is that our strategy of employing supervised learning for training SCGs depends on the availability of a densely sampled space, like NAS-Bench-101, which may not be available for real-world applications.

We further proposed two novel strategies for mitigating the challenges associated with learning the parameters of discrete SCGs in Section 5.2. By adjusting the scale parameter of the Gumbel noise perturbations and implementing dropout residual connections in the discrete-continuous computation graphs, we were able to mitigate problems such as small gradients and local minima. These solutions greatly improved the training behavior of the models and substantially increased their efficiency. In experiments across diverse domains, we demonstrated in Section 6.1 that our methods enable the training of complex discrete-continuous models, which could not be learned before. These models beat prior state-of-the-art models without discrete components on several benchmarks and showed a remarkable gain in generalization behavior.

We answered the research question **RQ2.2:** *How can discrete representations be integrated effectively into existing deep learning methods?* in Section 5.3. We developed a modified categorical VAE that leverages one-dimensional representation for each category. This method allows for effectively integrating discrete representations into deep learning methods. Furthermore, it creates a well-defined and ordered latent space that encourages disentanglement. However, it is worth noting that the implementation of such models may face limitations in terms of reconstruction quality.

Finally, we were able to answer the research question **RQ2.3:** *How can we further enhance the performance and efficiency of common discrete representations?* effectively through our discussions in Section 5.4. We proposed several improvements for common discrete representations, including introducing a Total Correlation (TC) regularizer, applying semi-supervised training, and addressing the straight-through gap. All these enhancements have improved the interpretability and efficiency of the discrete representations, outperforming Gaussian-based methods in terms of disentanglement quality, as we have shown in Section 6.2.

Implications of these findings suggest that discrete representations in deep learning have a significant potential for enhancing model interpretability and generalization behavior. However, they also highlight the need for continuous development and refinement of the training methodologies and techniques to ensure efficient learning and integration of discrete representations. The limitations indicate that while the proposed methods have shown remarkable results, they are not without their challenges and should be considered carefully in practical applications. Overall, these findings pave the way for further research into more innovative strategies and methods for integrating and enhancing discrete representations in deep learning.

7.2 Future Work

Building on the key findings of this thesis, we offer several promising directions for future work that can help further the understanding and application of discrete representations in deep learning. These include expanding their applications, improving training techniques, developing more powerful models with discrete latent spaces, and enhancing accuracy and reconstruction capabilities.

Exploring the expansion and application of discrete representations presents opportunities for future work. In particular, the potential of applying the proposed methods to reinforcement learning problems can be further investigated, effectively sidestepping the problem of using the high variance score function estimator. Furthermore, integrating discrete nodes into computation graphs of existing and new models can be considered. Combining the automatic assembly and execution of SCGs, as shown in the ListOps experiments, offers significant potential.

Another important aspect for future research is the improvement of training techniques and strategies for these representations. Despite the advances presented in this thesis, further challenges in training discrete representations still need to be solved, such as additional complexities of the Gumbel-Softmax trick. Future research can focus on discovering new model selection strategies tailored to discrete VAEs. These and other advancements could mitigate the training challenges and enhance the effectiveness of the discrete representation models.

Developing more powerful models with discrete latent spaces is another promising direction for future work. We have shown how to integrate discrete representations into existing VAE models used for disentanglement. Insights from the disentanglement literature could be beneficial in creating advanced models and further improving the use of discrete representations. Future work could develop novel regularizers, particularly those tackling total correlation tailored to discrete VAEs.

Lastly, further work might focus on enhancing the accuracy and reconstruction capabilities in discrete representation models. Although our models based on discrete representations demonstrate good generalization properties and better interpretability, they lag in task accuracy and reconstruction quality compared to their continuous counterparts. Therefore, future work should concentrate on improving these aspects, ensuring discrete representations offer a competitive and advantageous choice in all aspects.

In conclusion, exploring discrete representations in deep learning presented in this thesis opens up many opportunities for future research. The identified limitations and challenges provide a starting point for future work that, once addressed, can further the potential and applicability of discrete representations in deep learning.

Bibliography

[1] T. Adel, Z. Ghahramani, and A. Weller. Discovering interpretable representations for both deep generative and discriminative models. In *International Conference on Machine Learning*, pages 50–59. PMLR, 2018.

[2] J. Andreas, M. Rohrbach, T. Darrell, and D. Klein. Neural module networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[3] M. A. Arcones and E. Gine. On the bootstrap of u and v statistics. *The Annals of Statistics*, pages 655–674, 1992.

[4] P. Baldi. Autoencoders, unsupervised learning, and deep architectures. In *Proceedings of ICML workshop on unsupervised and transfer learning*, pages 37–49. JMLR Workshop and Conference Proceedings, 2012.

[5] J. Bastings, I. Titov, W. Aziz, D. Marcheggiani, and K. Sima'an. Graph convolutional encoders for syntax-aware neural machine translation. *arXiv preprint arXiv:1704.04675*, 2017.

[6] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[7] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.

[8] Y. Bengio, N. Léonard, and A. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv:1308.3432*, 2013.

[9] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, pages 1–9, 2013.

[10] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

[11] C. P. Burgess, I. Higgins, A. Pal, L. Matthey, N. Watters, G. Desjardins, and A. Lerchner. Understanding disentangling in β-vae. *arXiv preprint arXiv:1804.03599*, 2018.

[12] R. Cartuyvels, G. Spinks, and M.-F. Moens. Discrete and continuous representations and processing in deep learning: Looking forward. *AI Open*, 2: 143–159, 2021.

[13] J. Chen, L. Song, M. Wainwright, and M. Jordan. Learning to explain: An information-theoretic perspective on model interpretation. In *International Conference on Machine Learning*, pages 883–892. PMLR, 2018.

[14] R. T. Chen, X. Li, R. B. Grosse, and D. K. Duvenaud. Isolating sources of disentanglement in variational autoencoders. In *Advances in Neural Information Processing Systems*, volume 31, pages 2615–2625, 2018.

[15] J. Choi, K. M. Yoo, and S.-g. Lee. Learning to compose task-specific tree structures. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[16] G. M. Correia, V. Niculae, W. Aziz, and A. F. Martins. Efficient marginalization of discrete and structured latent variables via sparsity. In *Advances in Neural Information Processing Systems*, 2020.

[17] E. Creager, D. Madras, J.-H. Jacobsen, M. Weis, K. Swersky, T. Pitassi, and R. Zemel. Flexibly fair representation learning by disentanglement. In *International Conference on Machine Learning*, pages 1436–1445. PMLR, 2019.

[18] R. Das, A. Neelakantan, D. Belanger, and A. McCallum. Chains of reasoning over entities, relations, and text using recurrent neural networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 132–141, 2017.

[19] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[20] X. Dong and Y. Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. In *International Conference on Learning Representations*, 2020.

[21] G. Dulac-Arnold, N. Levine, D. J. Mankowitz, J. Li, C. Paduraru, S. Gowal, and T. Hester. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Machine Learning*, 110:2419–2468, 2021.

[22] E. Dupont. Learning disentangled joint continuous and discrete representations. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

[23] C. Eastwood and C. K. Williams. A framework for the quantitative evaluation of disentangled representations. In *International Conference on Learning Representations*, 2018.

[24] T. Elsken, J. H. Metzen, and F. Hutter. Neural architecture search: A survey. *The Journal of Machine Learning Research*, 20(1):1997–2017, 2019.

[25] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, pages 1263–1272. PMLR, 2017.

[26] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal. Explaining explanations: An overview of interpretability of machine learning. In *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*, pages 80–89. IEEE, 2018.

[27] M. W. Gondal, M. Wuthrich, D. Miladinovic, F. Locatello, M. Breidt, V. Volchkov, J. Akpo, O. Bachem, B. Schölkopf, and S. Bauer. On the transfer of inductive bias from simulation to the real world: a new disentanglement dataset. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[28] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[29] W. Grathwohl, D. Choi, Y. Wu, G. Roeder, and D. Duvenaud. Backpropagation through the void: Optimizing control variates for black-box gradient estimation. In *International Conference on Learning Representations*, 2018.

[30] S. Gu, S. Levine, I. Sutskever, and A. Mnih. Muprop: Unbiased backpropagation for stochastic neural networks. In *International Conference on Learning Representations*, 2016.

[31] J. Gui, Z. Sun, Y. Wen, D. Tao, and J. Ye. A review on generative adversarial networks: Algorithms, theory, and applications. *IEEE Transactions on Knowledge and Data Engineering*, 35(4):3313–3332, 2021.

[32] E. J. Gumbel. Les valeurs extrêmes des distributions statistiques. In *Annales de l'institut Henri Poincaré*, volume 5, pages 115–158, 1935.

[33] E. J. Gumbel. *Statistical theory of extreme values and some practical applications: a series of lectures*, volume 33. US Government Printing Office, 1954.

[34] K. Guu, J. Miller, and P. Liang. Traversing knowledge graphs in vector space. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015.

[35] D. Hafner, T. Lillicrap, M. Norouzi, and J. Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.

[36] D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

[37] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

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

[39] I. Higgins, D. Amos, D. Pfau, S. Racanière, L. Matthey, D. J. Rezende, and A. Lerchner. Towards a definition of disentangled representations. *arXiv preprint arXiv:1812.02230*, 2018.

[40] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851, 2020.

[41] E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*, 2017.

[42] Y. Jeong and H. O. Song. Learning discrete and continuous factors of data via alternating disentanglement. In *International Conference on Machine Learning*, pages 3091–3099. PMLR, 2019.

[43] K. Kandasamy, W. Neiswanger, J. Schneider, B. Poczos, and E. P. Xing. Neural architecture search with bayesian optimisation and optimal transport. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

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

[45] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2013.

[46] D. P. Kingma, T. Salimans, and M. Welling. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems*, volume 28, pages 2575–2583, 2015.

[47] D. A. Klindt, L. Schott, Y. Sharma, I. Ustyuzhaninov, W. Brendel, M. Bethge, and D. M. Paiton. Towards nonlinear disentanglement in natural data with temporal sparse coding. In *International Conference on Learning Representations*, 2021.

[48] T. Koo, A. Globerson, X. Carreras, and M. Collins. Structured prediction models via the matrix-tree theorem. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 141–150, 2007.

[49] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. *University of Toronto*, 2009.

[50] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[51] T. D. Kulkarni, W. F. Whitney, P. Kohli, and J. Tenenbaum. Deep convolutional inverse graphics network. In *Advances in Neural Information Processing Systems*, volume 28, 2015.

[52] A. Kumar, P. Sattigeri, and A. Balakrishnan. Variational inference of disentangled latent concepts from unlabeled observations. In *International Conference on Learning Representations*, 2017.

[53] L. Landrieu and M. Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4558–4567, 2018.

[54] Y. LeCun, F. J. Huang, and L. Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 2, pages II–104. IEEE, 2004.

[55] Y. Li, O. Vinyals, C. Dyer, R. Pascanu, and P. Battaglia. Learning deep generative models of graphs. *arXiv preprint arXiv:1803.03324*, 2018.

[56] H. Liu, K. Simonyan, O. Vinyals, C. Fernando, and K. Kavukcuoglu. Hierarchical representations for efficient architecture search. In *International Conference on Learning Representations*, 2018.

[57] H. Liu, K. Simonyan, and Y. Yang. Darts: Differentiable architecture search. In *International Conference on Learning Representations*, 2019.

[58] F. Locatello, G. Abbati, T. Rainforth, S. Bauer, B. Schölkopf, and O. Bachem. On the fairness of disentangled representations. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[59] F. Locatello, S. Bauer, M. Lucic, G. Raetsch, S. Gelly, B. Schölkopf, and O. Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. In *International Conference on Machine Learning*, pages 4114–4124. PMLR, 2019.

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

[61] F. Locatello, M. Tschannen, S. Bauer, G. Rätsch, B. Schölkopf, and O. Bachem. Disentangling factors of variations using few labels. In *International Conference on Learning Representations*, 2020.

[62] J. Lucas, G. Tucker, R. B. Grosse, and M. Norouzi. Don't blame the elbo! a linear vae perspective on posterior collapse. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[63] R. Luo, F. Tian, T. Qin, E. Chen, and T.-Y. Liu. Neural architecture optimization. In *Advances in Neural Information Processing Systems*, pages 7816–7827, 2018.

[64] C. J. Maddison, A. Mnih, and Y. W. Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations*, 2017.

[65] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey. Adversarial autoencoders. In *International Conference on Learning Representations*, 2016.

[66] R. Manhaeve, S. Dumancic, A. Kimmig, T. Demeester, and L. De Raedt. Deepproblog: Neural probabilistic logic programming. In *Advances in Neural Information Processing Systems*, volume 31, pages 3749–3759, 2018.

[67] A. Mnih and K. Gregor. Neural variational inference and learning in belief networks. In *International Conference on Machine Learning*, pages 1791–1799. PMLR, 2014.

[68] S. Mohamed, M. Rosca, M. Figurnov, and A. Mnih. Monte carlo gradient estimation in machine learning. *The Journal of Machine Learning Research*, 21(1):5183–5244, 2020.

[69] F. Monti, M. Bronstein, and X. Bresson. Geometric matrix completion with recurrent multi-graph neural networks. In *Advances in Neural Information Processing Systems*, pages 3697–3707, 2017.

[70] N. Nangia and S. Bowman. Listops: A diagnostic dataset for latent tree learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 92–99, 2018.

[71] X. Nguyen, M. J. Wainwright, and M. I. Jordan. Estimating divergence functionals and the likelihood ratio by convex risk minimization. *IEEE Transactions on Information Theory*, 56(11):5847–5861, 2010.

[72] M. Nickel, V. Tresp, and H.-P. Kriegel. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th International Conference on Machine Learning*, 2011.

[73] V. Niculae and A. Martins. Lp-sparsemap: Differentiable relaxed optimization for sparse structured prediction. In *International Conference on Machine Learning*, pages 7348–7359. PMLR, 2020.

[74] V. Niculae, A. Martins, M. Blondel, and C. Cardie. Sparsemap: Differentiable sparse structured inference. In *International Conference on Machine Learning*, pages 3799–3808. PMLR, 2018.

[75] S. Ozair, Y. Li, A. Razavi, I. Antonoglou, A. Van Den Oord, and O. Vinyals. Vector quantized models for planning. In *International Conference on Machine Learning*, pages 8302–8313. PMLR, 2021.

[76] E. Parisotto, A.-r. Mohamed, R. Singh, L. Li, D. Zhou, and P. Kohli. Neuro-symbolic program synthesis. *arXiv preprint arXiv:1611.01855*, 2016.

[77] M. B. Paulus, D. Choi, D. Tarlow, A. Krause, and C. J. Maddison. Gradient estimation with stochastic softmax tricks. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

[78] A. Pervez, T. Cohen, and E. Gavves. Low bias low variance gradient estimates for boolean stochastic networks. In *International Conference on Machine Learning*, pages 7632–7640. PMLR, 2020.

[79] J. Peters, D. Janzing, and B. Schölkopf. *Elements of causal inference: Foundations and learning algorithms*. The MIT Press, 2017.

[80] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean. Efficient neural architecture search via parameters sharing. In *International Conference on Machine Learning*, pages 4095–4104. PMLR, 2018.

[81] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages 8821–8831. PMLR, 2021.

[82] R. Ranganath, S. Gerrish, and D. Blei. Black box variational inference. In *Artificial Intelligence and Statistics*, pages 814–822. PMLR, 2014.

[83] A. Razavi, A. Van den Oord, and O. Vinyals. Generating diverse high-fidelity images with vq-vae-2. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[84] T. RC and M. Lázaro-Gredilla. Local expectation gradients for black box variational inference. In *Advances in Neural Information Processing Systems*, volume 28, 2015.

[85] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. V. Le, and A. Kurakin. Large-scale evolution of image classifiers. In *International Conference on Machine Learning*, pages 2902–2911. PMLR, 2017.

[86] S. E. Reed, Y. Zhang, Y. Zhang, and H. Lee. Deep visual analogy-making. In *Advances in Neural Information Processing Systems*, volume 28, 2015.

[87] D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, pages 1278–1286. PMLR, 2014.

[88] K. Ridgeway and M. C. Mozer. Learning deep disentangled embeddings with the f-statistic loss. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

[89] H. Robbins and S. Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[90] T. Rocktäschel and S. Riedel. End-to-end differentiable proving. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[91] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. 2022 ieee. In *CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10674–10685, 2022.

[92] D. Ruffinelli, S. Broscheit, and R. Gemulla. You {can} teach an old dog new tricks! on training knowledge graph embeddings. In *International Conference on Learning Representations*, 2020.

[93] B. Schölkopf, D. Janzing, J. Peters, E. Sgouritsa, K. Zhang, and J. M. Mooij. On causal and anticausal learning. In *International Conference on Machine Learning*, 2012.

[94] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

[95] J. Schulman, N. Heess, T. Weber, and P. Abbeel. Gradient estimation using stochastic computation graphs. In *Advances in Neural Information Processing Systems*, volume 28, 2015.

[96] O. Shayer, D. Levi, and E. Fetaya. Learning discrete weights using the local reparameterization trick. In *International Conference on Learning Representations*, 2018.

[97] R. Shu, Y. Chen, A. Kumar, S. Ermon, and B. Poole. Weakly supervised disentanglement with guarantees. In *International Conference on Learning Representations*, 2020.

[98] N. Siddharth, B. Paige, J.-W. van de Meent, A. Desmaison, N. Goodman, P. Kohli, F. Wood, and P. Torr. Learning disentangled representations with semi-supervised deep generative models. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[99] M. Simonovsky and N. Komodakis. Graphvae: Towards generation of small graphs using variational autoencoders. In *International Conference on Artificial Neural Networks*, pages 412–422. Springer, 2018.

[100] R. Socher, D. Chen, C. D. Manning, and A. Ng. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems*, pages 926–934, 2013.

[101] M. Sugiyama, T. Suzuki, and T. Kanamori. Density-ratio matching under the bregman divergence: a unified framework of density-ratio estimation. *Annals of the Institute of Statistical Mathematics*, 64(5):1009–1044, 2012.

[102] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[103] M. Titsias and J. Shi. Double control variates for gradient estimation in discrete latent variable models. In *International Conference on Artificial Intelligence and Statistics*, pages 6134–6151. PMLR, 2022.

[104] S. Tokui and I. Sato. Evaluating the variance of likelihood-ratio gradient estimators. In *International Conference on Machine Learning*, pages 3414–3423. PMLR, 2017.

[105] K. Toutanova and D. Chen. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pages 57–66. Association for Computational Linguistics, 2015.

[106] F. Träuble, E. Creager, N. Kilbertus, F. Locatello, A. Dittadi, A. Goyal, B. Schölkopf, and S. Bauer. On disentangled representations learned from correlated data. In *International Conference on Machine Learning*, pages 10401–10412. PMLR, 2021.

[107] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard. Complex embeddings for simple link prediction. In *Proceedings of the 33nd International Conference on Machine Learning*, pages 2071–2080. PMLR, 2016.

[108] M. Tschannen, O. Bachem, and M. Lucic. Recent advances in autoencoder-based representation learning. *arXiv preprint arXiv:1812.05069*, 2018.

[109] G. Tucker, A. Mnih, C. J. Maddison, D. Lawson, and J. Sohl-Dickstein. Rebar: Low-variance, unbiased gradient estimates for discrete latent variable models. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[110] A. Van Den Oord, O. Vinyals, et al. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[111] S. Van Steenkiste, F. Locatello, J. Schmidhuber, and O. Bachem. Are disentangled representations helpful for abstract visual reasoning? In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[112] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[113] Q. Wang, P. Huang, H. Wang, S. Dai, W. Jiang, J. Liu, Y. Lyu, Y. Zhu, and H. Wu. Coke: Contextualized knowledge graph embedding. *arXiv preprint arXiv:1911.02168*, 2019.

[114] S. Watanabe. Information theoretical analysis of multivariate correlation. *IBM Journal of Research and Development*, 4(1):66–82, 1960.

[115] N. Watters, L. Matthey, S. Borgeaud, R. Kabra, and A. Lerchner. Spriteworld: A flexible, configurable reinforcement learning environment, 2019.

[116] N. Watters, L. Matthey, C. P. Burgess, and A. Lerchner. Spatial broadcast decoder: A simple architecture for learning disentangled representations in vaes. *arXiv preprint arXiv:1901.07017*, 2019.

[117] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Reinforcement learning*, pages 5–32, 1992.

[118] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, 2020.

[119] D. Xu, Y. Zhu, C. B. Choy, and L. Fei-Fei. Scene graph generation by iterative message passing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5410–5419, 2017.

[120] B. Yang, W.-t. Yih, X. He, J. Gao, and L. Deng. Embedding entities and relations for learning and inference in knowledge bases. In *International Conference on Learning Representations*, 2015.

[121] L. Yi, H. Su, X. Guo, and L. J. Guibas. Syncspeccnn: Synchronized spectral cnn for 3d shape segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2282–2290, 2017.

[122] W. Yin, Y. Yaghoobzadeh, and H. Schütze. Recurrent one-hop predictions for reasoning over knowledge graphs. *arXiv preprint arXiv:1806.04523*, 2018.

[123] C. Ying, A. Klein, E. Christiansen, E. Real, K. Murphy, and F. Hutter. Nas-bench-101: Towards reproducible neural architecture search. In *International Conference on Machine Learning*, pages 7105–7114. PMLR, 2019.

[124] J. You, R. Ying, X. Ren, W. Hamilton, and J. Leskovec. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *International Conference on Machine Learning*, pages 5708–5717. PMLR, 2018.

[125] A. Zela, J. Siems, and F. Hutter. Nas-bench-1shot1: Benchmarking and dissecting one-shot neural architecture search. In *International Conference on Learning Representations*, 2020.

[126] M. Zhang, S. Jiang, Z. Cui, R. Garnett, and Y. Chen. D-vae: A variational autoencoder for directed acyclic graphs. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[127] B. Zoph and Q. V. Le. Neural architecture search with reinforcement learning. In *International Conference on Learning Representations*, 2017.

[128] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8697–8710, 2018.

Appendix A

Implementation Details

A.1 Efficient Parameter Learning of Discrete Stochastic Computation Graphs

All our methods were implemented with PyTorch and were run on a GeForce RTX 2080 Ti GPU. For the experiments we picked the best performing learning rate out of $\{1e^{-4}, 3e^{-4}, 5e^{-4}, 1e^{-3}\}$ and the best performing softmax temperature τ out of $\{1, 2, 4\}$. We found a temperature of $\tau = 4$ not sufficiently high enough for the MNIST addition experiments and picked $\tau = 8$ for these experiments.

A.1.1 Unsupervised Parsing on ListOps

For all ListOps experiments, we use a hidden dimension $\mathrm{dim} = 60$, a batch size of 100, and train the model for a total number of 100 epochs. We use the Adam optimizer with a learning rate of 0.0005 to minimize the cross-entropy loss. After every epoch, we evaluate on the validation set and save the best model. We use discretization after the first 4 out of the 5 GNN message passes. We always use an exponential function to increase β and a linear function to increase α (during experiments for which they are not constant). The temperatures are updated 10 times per epoch; we use $\beta_t = \tau(1 - e^{-t\gamma})$ for $\gamma = 0.008,\ \tau = 1$ and $\alpha_t = \max(1, 0.002t)$. We repeat every experiment 8 times.

Our model is taken from Paulus et al. [77] and has following structure. For the encoder, we start with an embedding layer for the 14 tokens. We use two independent one-directional LSTMs with a single layer, respectively. We use a token-wise multiplication of the two sequence outputs to obtain a latent graph and use Gumbel-softmax on this latent graph:

$$\begin{aligned} x &= E_1[14, \dim](tok) \\ q &= LSTM_1(x) \\ k &= LSTM_2(x) \\ A'_{ij} &= qk^\intercal \\ A_{ij} &= \frac{1}{\lambda}(A'_{ij} + \text{Gumbel}(0, \tau)). \end{aligned}$$

To obtain the edge precision, we compare the latent graph A_{ij} with the ground truth adjacency matrix B_{ij}. Due to our arborescence prior as described in the main paper, we always have the same number of edges resulting in equality of edge precision and edge recall. For the GNN we use a second, independent embedding layer for the tokens. For each GNN message pass, we use the latent representation after the embedding layer of a token and concatenate it with the representation of the node of the incoming message. In the baseline experiments (Arb.), we use the current state of the token instead of the first embedding during all message passes except the first one. The message is transformed by a message MLP with dropout probability 0.1 and is summed up regarding the edge weights of the latent graph A^{ij}. The new node state is summed with the one before the message pass:

$$\begin{aligned} x_e &= E_2[14, \dim](tok) \\ p'' &= [x_e; x_j] \quad \text{or} \quad [x_i; x_j] \\ p' &= \text{Dropout}(\text{ReLU}(\text{Lin}_{1,Bias}^{\dim \times 2\dim}(p''))) \\ p &= \text{ReLU}(\text{Lin}_{2,Bias}^{\dim \times \dim}(p')) \\ m &= A_i p \\ x'_i &= x_i + m. \end{aligned}$$

In those experiments that utilize the ground truth edges (GT) instead of the latent graph, we replace the second last line $m = A_i p$ by $m = B_i p$ with B being the ground truth adjacency matrix. To create Figure 5.5, we read out the mean absolute values of the gradient at x_e as well as x'_i for each of the first 4 GNN massage passes. For the discretization, we use the classification layer that maps the embedding of the final output to the logits of the 10 classes and the weights of the embedding layer that map the 10 numeral tokens to their respective embeddings:

$$\begin{aligned}
\boldsymbol{\theta} &= \mathrm{Lin}_4^{10\times\dim}(\mathrm{ReLU}(\mathrm{Lin}_{3,Bias}^{\dim\times\dim}(x_i'))) \\
\boldsymbol{z} &= \frac{1}{\lambda}(\boldsymbol{\theta} + \mathrm{Gumbel}(0,\tau)) \\
\boldsymbol{v} &= \boldsymbol{z}^\intercal E_2([0,\dots,9]) \\
x_i^{\text{next}} &= \begin{cases} \boldsymbol{v} & \text{with probability } \alpha, \\ x_i' + \boldsymbol{v} & \text{else.} \end{cases}
\end{aligned}$$

To read out the intermediate results, we compare the final (after the last GNN message pass) $\boldsymbol{\theta}$ of all operation tokens with the ground truth intermediate results obtained by executing the operations. The experiments without dropout residuals are always set to utilize $\alpha = 1$. To obtain the task class of the full example, we utilize the same MLP from the discretization as the classification layer, i.e., $\mathrm{Lin}_4^{10\times\dim}(\mathrm{ReLU}(\mathrm{Lin}_{3,Bias}^{\dim\times\dim}(x_0^{\text{last}})))$.
The baseline LSTM model consists of an one-directional LSTM with a single layer. The hidden state of the final token is fed to a classification MLP, i.e., $\mathrm{Lin}^{10\times\dim}(\mathrm{ReLU}(\mathrm{Lin}_{Bias}^{\dim\times\dim}(h_{-1})))$.

A.1.2 Multi-Hop Reasoning over Knowledge Graphs

For all Knowledge Graph experiments, we use a hidden dimension $\dim = 256$, a batch size of 512, and train the model for a total of 200 epochs. We use the Adam optimizer with a learning rate of 0.001 to minimize the cross-entropy loss. We use a randomized grid search training on paths of length 1 and validating hitset 10 on paths of length 2 for the L2 regularization of the entities and the relations between $1e^{-20}$, ..., $1e^{-5}$ and for the dropout probabilities for the subject, object and relations between 0, ..., 0.8, respectively. This results in an entity regularization of $1e^{-15}$, a relation regularization of $1e^{-9}$, a subject dropout of 0.7 $[0.1]$, an object dropout of 0.1 $[0.6]$ and a relation dropout of 0.5 $[0.2]$ for WordNet [Freebase]. After every 10th epoch, we evaluate on the validation set and save the best model. We use discretization after every single relation. We always use an exponential function to increase β and a linear function to increase α (during experiments for which they are not constant). The temperatures are updated 3 times per epoch; we use $\beta_t = \tau(1 - e^{-t\gamma})$ for $\gamma = 0.008,\ \tau = 1$ and $\alpha_t = \max(1, 0.005t)$. We repeat every experiment 4 times.

Our model is based on the ComplEx model from Trouillon et al. [107]. Given the current (intermediate) entity embedding $\boldsymbol{s}$ and relation type embedding $\boldsymbol{r}$ we compute the logits $\boldsymbol{\theta}$ for all possible entities with the ComplEx scoring function

$\text{score} = \text{Re} < \boldsymbol{s}, \boldsymbol{r}, \cdot >$. Hence, we obtain the logits for all possible entities. Using the Gumbel-softmax trick with parameter $\boldsymbol{\theta}$, scale β, and temperature τ, we obtain the sample $\boldsymbol{z} \in \mathbb{R}^n$. The function h now computes $\boldsymbol{z}^\intercal \mathbf{E}$ where $\mathbf{E}$ be the matrix whose rows are the entity embeddings. We use dropout residual connections between the input and output vectors of the discrete-continuous component during training. Guu et al. [34] introduced their own evaluation protocol for multi-hop reasoning that we adopted. On the one hand, we calculate all possible objects that can be reached traversing each path. These are the positives and are filtered during evaluation. On the other hand, we calculate all possible objects that can be reached by the final relation of the path individually. These are the negatives that we rank our prediction against. We compare our best performing model against two RNN models, Path-RNN [18] and ROP [122] as well as against the state-of-the-art transformer model CoKE [113].

We use a slightly different setup for the FB15K237 experiments in Section 6.1.2. Here, we train the same model for a total number of 100 epochs and for each path from subject to object as well as from object to subject using reciprocal relations [92]. The evaluation is copied from the standard link prediction task [9], that is, we evaluate all paths in the forward direction from subject to object as well as in the backward direction from object to subject. We also use the filter method for the positives, but we compare against all other possible objects and not only the ones reachable by the last relation.

A.1.3 End-to-End Learning of MNIST Addition

For the MNIST addition experiments, we use a batch size of 16 and train the model for a total number of 30 epochs. We use the Adam optimizer with a learning rate of 0.0001 to minimize the cross-entropy loss. We use the dataset from Manhaeve et al. [66]. Since they do not offer a validation set, we validate the model twice per epoch on the test set and record the best test accuracy for all models. We use discretization after the CNN encoding layer, i.e., before the addition layer. The temperatures are updated 8 times per epoch; we increase β and α by $\beta_t = \tau(1 - e^{-t\gamma})$ for $\gamma = 0.008, \tau = 8$ and $\alpha_t = \max(1, 0.002t)$. We repeat every experiment 8 times.

Our model is based on the baseline model used by Manhaeve et al. [66] and has the following structure. The CNN encoder consists of two convolutional layers with kernel size 5 and filter size 6 and 16, respectively. Each convolutional layer is followed by 2D max-pooling layer of size 2×2 and a ReLU activation function.

An MLP with 3 layers transforms the output to an embedding size of 84.

$$\begin{aligned}
e' &= \mathrm{ReLU}(\mathrm{maxpool}_{2\times 2}(\mathrm{Conv2d}_{6,5}(inp)))\\
e &= \mathrm{ReLU}(\mathrm{maxpool}_{2\times 2}(\mathrm{Conv2d}_{16,5}(e')))\\
x' &= \mathrm{Lin}^{84\times 84}_{Bias}(\mathrm{ReLU}(\mathrm{Lin}^{84\times 120}_{Bias}(\mathrm{ReLU}(\mathrm{Lin}^{120\times 256}_{Bias}(e))))).
\end{aligned}$$

For the discretization we use a single classification matrix $C \in \mathbb{R}^{19\times 84}$. This matrix is also used for the classification after the addition layer.

$$\begin{aligned}
\boldsymbol{\theta} &= Cx'\\
\boldsymbol{z} &= \frac{1}{\lambda}(\boldsymbol{\theta} + \mathrm{Gumbel}(0,\tau))\\
\boldsymbol{v} &= \boldsymbol{z}^{\intercal} C\\
x &= \begin{cases} \boldsymbol{v} & \text{with probability } \alpha,\\ x' + \boldsymbol{v} & \text{else.}\end{cases}
\end{aligned}$$

The addition layer concatenates the final embeddings of both images and transform them through a MLP with 2 layers into a single representation.

$$x_{1,2} = \mathrm{ReLU}(\mathrm{Lin}^{84\times 168}_{Bias}(\mathrm{ReLU}(\mathrm{Lin}^{168\times 168}_{Bias}([x_1; x_2])))).$$

We obtain the final class log-probabilities by computing $Cx_{1,2}$. This results in a model with a total of $94,900$ parameters. We use the code offered by Manhaeve et al. [66] to run the DeepProbLog experiments as well as the baseline model they compared to. These two models are executed in a single epoch and use a batch size of 1 and 2, respectively. The DeepProbLog model uses a similar encoder to get predictions for each of the images individually and has the following structure.

$$\begin{aligned}
e' &= \mathrm{ReLU}(\mathrm{maxpool}_{2\times 2}(\mathrm{Conv2d}_{6,5}(inp)))\\
e &= \mathrm{ReLU}(\mathrm{maxpool}_{2\times 2}(\mathrm{Conv2d}_{16,5}(e')))\\
out &= \mathrm{Lin}^{10\times 120}(\mathrm{ReLU}(\mathrm{Lin}^{120\times 256}_{Bias}(e))),
\end{aligned}$$

The 2 image outputs are then fed into the probabilistic logic program ProbLog together with the following annotated disjunction, which handles the logic of addition to obtain the final predictions.

$$\begin{aligned}
&\mathrm{nn}(\mathrm{mnist\_net}, [X], Y, [0,1,2,3,4,5,6,7,8,9]) :: \mathrm{digit}(X, Y);\\
&\mathrm{add}(X, Y, Z) :- \mathrm{digit}(X, X2), \mathrm{digit}(Y, Y2), Z \text{ is } X2 + Y2.
\end{aligned}$$

Different to DeepProbLog or our model, the baseline model concatenates the images beforehand and uses the following layers.

$$\begin{aligned} e' &= \mathrm{ReLU}(\mathrm{maxpool}_{2\times 2}(\mathrm{Conv2d}_{6,5}(inp))) \\ e &= \mathrm{ReLU}(\mathrm{maxpool}_{2\times 2}(\mathrm{Conv2d}_{16,5}(e'))) \\ out &= \mathrm{Lin}^{19\times 84}(\mathrm{ReLU}(\mathrm{Lin}^{84\times 120}_{Bias}(\mathrm{ReLU}(\mathrm{Lin}^{120\times 704}_{Bias}(e))))), \end{aligned}$$

which results in a model with a total of $98,951$ parameters.

For the experiments in Figure 4.2, we also use a batch size of 16, set $\tau = 8.0$, use a learning rate of 0.0003 and train the model for a total number of 30 epochs. If we update temperatures, we also update them 8 times per epoch. We do not use dropout residuals. For the experiment Base, we use a constant $\tau = 8.0$, $\beta = 1.0$ and $\gamma = 0.008$. For the experiment TauAnn, we set $\beta = 1.0$ constant and anneal τ by $\tau_t = \max(1.0,\ \tau e^{-t\gamma})$. For the experiment TM, we set $\tau = 8.0$ constant and increase β by $\beta_t = \tau(1 - e^{-t\gamma})$.

A.2 Efficient Structure Learning of Stochastic Computation Graphs

The node and graph dimensions, $d_n = 250$ and $d_g = 56$, are chosen as in [126] to attain comparability. For all experiments, we used the Adam optimizer with no dropouts. For training VS-GAE, we used a learning rate of $1e^{-4}$ for a total amount of 300 epochs. Whenever the loss did not drop for 10 epochs, we decreased the learning rate by a factor of $1e^{-1}$.

A.2.1 The Encoder

Message The message module $M^{(t)}$ concatenates the embedding of the considered node $h_v^{(t-1)}$ as well as the incoming embedding $h_u^{(t-1)}$, each of dimension d_n. It further performs a linear transformation on the concatenated embedding. The reverse message module $M^{(t)}_{out}$ is a clone of $M^{(t)}$ initialized with its own weights,

$$\begin{aligned} M^{(t)} &= \mathrm{Lin}_{2d_n\times 2d_n}\big([h_v^{(t-1)}, h_u^{(t-1)}]\big), \\ M^{(t)}_{out} &= \mathrm{Lin}'_{2d_n\times 2d_n}\big([h_v^{(t-1)}, h_u^{(t-1)}]\big). \end{aligned}$$

The message module (green) and the reverse message (red) can be seen on the left side of Figure 5.1 in the main paper.

Update The update module $U^{(t)}$ is a single GRU cell. First, the incoming messages $m_{u\to v}$ and $m^{out}_{u\to v}$ are added and handled as the GRU input. Second, the node embedding $h_v^{(t-1)}$ is treated as the hidden state and is updated,

$$U^{(t)} = \mathrm{GRUCell}_{2d_n,d_n}\big(m_{u\to v} + m^{out}_{u\to v},\ h_v^{(t-1)}\big).$$

Aggregation We use two rounds of propagation before aggregating the node embeddings into a single graph embedding. This graph aggregation consists of two parts. First, a linear layer transforms the node embeddings to the required graph embedding dimension d_g. Second, another linear layer combined with a sigmoid handles each node's fraction in the graph embedding,

$$\begin{aligned} A_1 &= \mathrm{Lin}_{d_n\times 2d_n}(h_v^{(2)}),\\ A_2 &= \sigma\big(\mathrm{Lin}_{d_n\times 1}(h_v^{(2)})\big),\\ A &= \sum_v A_1 \odot A_2. \end{aligned}$$

The aggregation function for the variational outputs $\tilde{A}$ is an exact copy of A with its own weights. An illustration of the aggregation module is given in Figure 5.1 (right).

A.2.2 VS-GAE

InitNode The learnable embedding look-up table E consists of five embeddings of size d_n, one for each of the five node types. It is initialized from $\mathcal{N}(0,1)$. The InitNode module concatenates the sampled point of the latent space **z** of size d_g, the summary of the partially created graph $h_{G^{(t)}}$ of size d_g and the node embedding of the picked node type,

$$\begin{aligned} f^1_{\mathrm{initNode}} &= \mathrm{Lin}_{2d_g+d_n\times d_g+d_n}([\mathbf{z}, h_{G^{(t)}}, L(\mathrm{type})]),\\ f_{\mathrm{initNode}} &= \mathrm{Lin}_{d_g+d_n\times d_n}\big(\mathrm{ReLU}(f^1_{\mathrm{initNode}})\big). \end{aligned}$$

This can be seen in Figure 5.2 c). For the very first embedding, we exclude the partially created graph,

$$\begin{aligned} f^1_{\mathrm{startNode}} &= \mathrm{Lin}_{d_g+d_n\times d_g+d_n}([\mathbf{z}, L(\mathrm{type})]),\\ f_{\mathrm{startNode}} &= \mathrm{Lin}_{d_g+d_n\times d_n}\big(\mathrm{ReLU}(f^1_{\mathrm{startNode}})\big). \end{aligned}$$

GraphProp The GraphProp module consists of two rounds of message passing with the exact same modules from above and a variance-free graph aggregation. Each of these modules is initialized with its own weights, see Figure 5.2 a).

AddNode The AddNode module concatenates the sampled point of the latent space $\mathbf{z}$ and the summary of the partially created graph $h_{G^{(t)}}$ and outputs logits over all five possible node types,

$$\begin{aligned} f^1_{\text{addNode}} &= \text{Lin}_{2d_g \times d_g}([\mathbf{z}, h_{G^{(t)}}]), \\ f_{\text{addNode}} &= \text{Lin}_{d_g \times 5}\big(\text{ReLU}(f^1_{\text{addNode}})\big). \end{aligned}$$

This can be seen in Figure 5.2 b).

AddEdges The AddEdges module concatenates the embedding of the newly created node h_{t+1} and each previous node h_v as well as the sampled point of the latent space $\mathbf{z}$ and the summary of the partially created graph $h_{G^{(t)}}$,

$$\begin{aligned} f^1_{\text{addEdges}} &= \text{Lin}_{2d_g+2d_n \times d_g+d_n}([h_{t+1}, h_v, \mathbf{z}, h_{G^{(t)}}]), \\ f_{\text{addEdges}} &= \text{Lin}_{d_g+d_n \times 1}\big(\text{ReLU}(f^1_{\text{addEdges}})\big). \end{aligned}$$

The output is the score that describes the probability of an edge in logits. This process is illustrated in Figure 5.2 d).

A.3 Learning Disentangled Discrete Representations

Locatello et al. [59] unified the choice of architecture, batch size, and optimizer to guarantee a fair comparison among the different methods. We adopt these unifications and describe them here for the sake of completeness. The only differences emerge from the Gumbel-softmax distribution from Equation 4.12. For all experiments, we choose the same number of $m = 64$ categories. If not mentioned differently, we utilize the symmetric interval $[-1, 1]$ for the latent variable. We utilize a constant Gumbel-softmax temperature of $\lambda = 1.0$ and, instead, increase the scale parameter of the Gumbel distribution from 0.5 to 2.0 w.r.t. a cosine annealing and set the scale parameter to 0.0 at test time. We found this annealing scheme to improve training stability while encouraging discrete representations. The implementation of the architectures is depicted in Table A.1, all hyperparameters can be found in Table A.2. We utilize the spatial broadcast decoder [116] for the Circles experiments with a latent space dimension of $n = 2$. The implementations for the Circles experiments can be found in Table A.4. If not mentioned differently, we utilize the ReLU activation function.

A.3.1 Dataset Details

Table A.3 contains a set of all ground-truth factors of variation for each dataset.

Table A.1: The architectures of the encoders and the decoder for the main experiments in Section 6.2.

Encoder (Gaussian)	**Encoder** (Discrete)	**Decoder**
Input: $64 \times 64 \times C$	Input: $64 \times 64 \times C$	Input: 10
Conv($4 \times 4,\ 32,\ s = 2$)	Conv($4 \times 4,\ 32,\ s = 2$)	FC(256)
Conv($4 \times 4,\ 32,\ s = 2$)	Conv($4 \times 4,\ 32,\ s = 2$)	FC($4 \times 4 \times 64$)
Conv($4 \times 4,\ 64,\ s = 2$)	Conv($4 \times 4,\ 64,\ s = 2$)	DeConv($4 \times 4,\ 64,\ s = 2$)
Conv($4 \times 4,\ 64,\ s = 2$)	Conv($4 \times 4,\ 64,\ s = 2$)	DeConv($4 \times 4,\ 32,\ s = 2$)
FC(256)	FC(256)	DeConv($4 \times 4,\ 32,\ s = 2$)
FC(2×10)	FC(10×64)	DeConv($4 \times 4,\ C,\ s = 2$)

Table A.2: The hyperparameters of the D-VAE model.

Parameter	**Model**	**Values**
Decoder type		Bernoulli
Batch size		64
Latent space dim.		10
Optimizer		Adam
Adam: β_1		0.9
Adam: β_2		0.999
Learning rate		$1e^{-4}$
Training steps		300 000
Latent space dim. (Circles)	Circles	2
Number of categories	discrete	64
Gumbel scale: init	discrete	0.5
Gumbel scale: final	discrete	2.0
Disc. Adam: β_1	TC regularizing	0.5
Disc. Adam: β_2	TC regularizing	0.9
γ	TC regularizing	$[10, 20, 30, 40, 50, 100]$
ω	semi-supervised	$[1, 2, 4, 6, 8, 16]$

Table A.3: The ground-truth factors of the datasets.

Dataset	**Ground-truth factor**	**Number of values**
dSprites	Shape	3
	Scale	6
	Orientation	40
	X-Position	32
	Y-Position	32
C-dSprites	Shape	3
	Scale	6
	Orientation	40
	X-Position	32
	Y-Position	32
	Color	Uniform$(0.5, 1.0)^3$
SmallNORB	Category	5
	Elevation	9
	Azimuth	18
	Lighting condition	6
Cars3D	Elevation	4
	Azimuth	24
	Object type	183
Shapes3D	Floor color	10
	Wall color	10
	Object color	10
	Object size	8
	Object type	4
	Azimuth	15
MPI3D	Object color	4
	Object shape	4
	Object size	2
	Camera height	3
	Background colors	3
	First DOF	40
	Second DOF	40
Circles	X-Position	Uniform$(0.2, 0.8)$
	Y-Position	Uniform$(0.2, 0.8)$

Table A.4: The architectures of the discriminator for the TC regularizing experiments and the spatial broadcast decoder [116] for the Circles experiments.

Discriminator	**Decoder** (Circles)
FC(1000), leaky ReLU	Input: 2
FC(1000), leaky ReLU	Tile($64 \times 64 \times 10$)
FC(1000), leaky ReLU	Concat. coordinate channels
FC(1000), leaky ReLU	Conv($4 \times 4,\ 64,\ s = 1$)
FC(1000), leaky ReLU	Conv($4 \times 4,\ 64,\ s = 1$)
FC(1000), leaky ReLU	Conv($4 \times 4,\ C,\ s = 1$)
FC(2)	

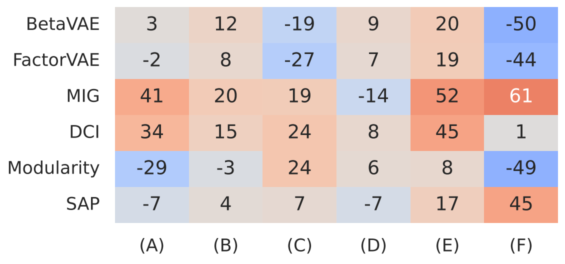


Figure B.1: The statistical efficiency of the simple downstream classification task of recovering the true factors of variations from the learned representation using gradient boosted trees (GBT). A high MIG score reliably leads to a higher sample efficiency for all datasets but Cars3D. The DCI score yields a positive correlation with the statistical efficiency.

Appendix B

Detailed Experimental Results

B.1 Statistical Sample Efficiency

We follow the simple downstream classification task from [59] of recovering the true factors of variations from the learned representation using gradient-boosted trees (GBT). Figure B.1 depicts this correlation regarding the GBT task for all six datasets. We can observe a high variance of the correlation depending on the selected disentanglement metric. The correlation with the MIG and DCI leads to a higher sample efficiency over most datasets.

B.2 Circles Experiment

The latent space visualizations of the circles experiment [116], sorted by the MIG score of all 50 models of the Gaussian VAE and the discrete VAE, respectively. Figure B.2 depicts the Gaussian latent spaces. Even the latent spaces yielding the best MIG scores are affected by rotation. Figure B.3 depicts the discrete latent spaces. More than 25% of the latent spaces lie parallel to the axes.

B.3 Comparison of the Unregularized Models

Figure B.4 and Figure B.5 depict the comparison of the unregularized models as violin plots for all datasets and metrics. The discrete VAE improves over its Gaussian counterpart in 31 out of 36 cases.

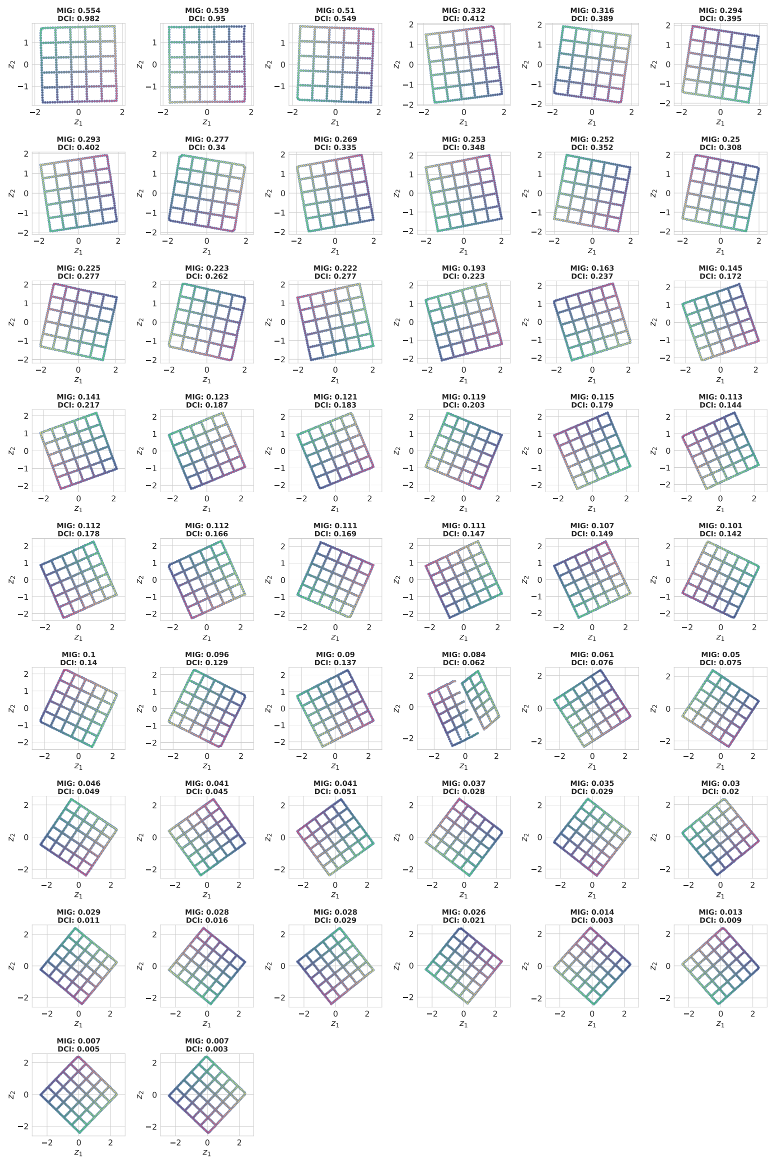


Figure B.2: A latent space geometry analysis of the circles experiment [116] including the MIG and DCI scores. We depict the latent space visualizations of all 50 models of the Gaussian VAE sorted by the MIG score.

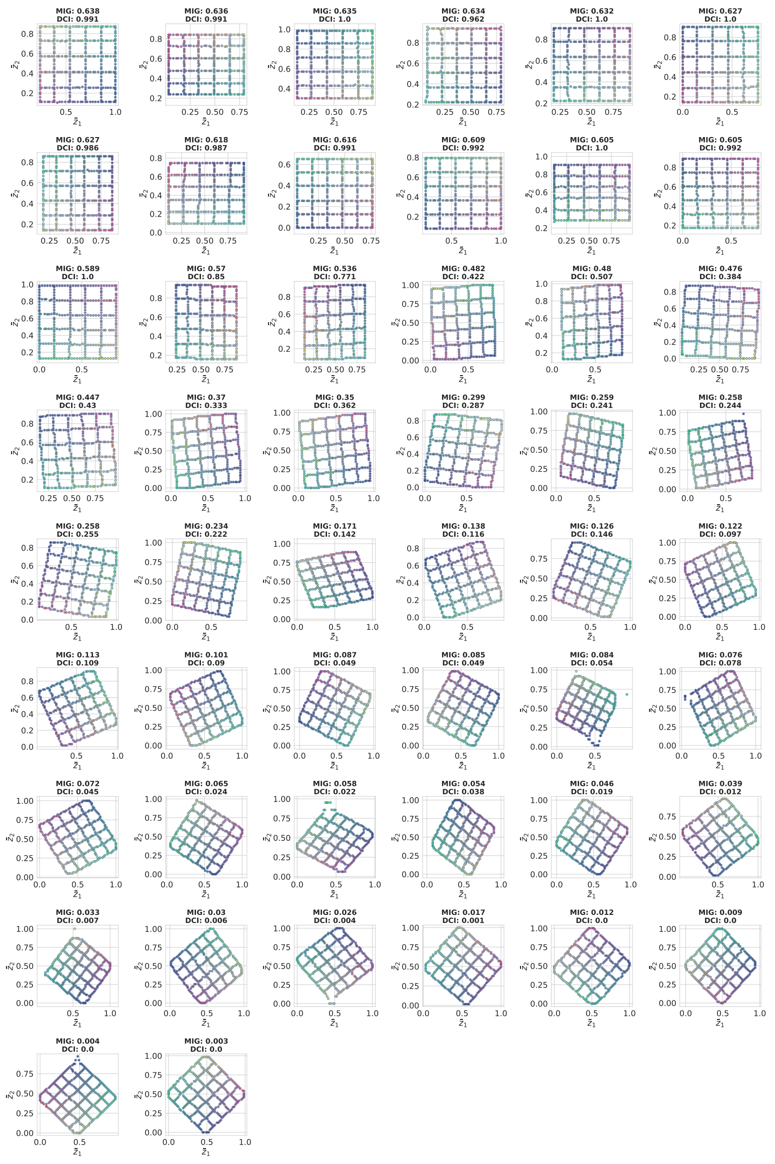


Figure B.3: A latent space geometry analysis of the circles experiment [116] including the MIG and DCI scores. We depict the latent space visualizations of all 50 models of the discrete VAE sorted by the MIG score.

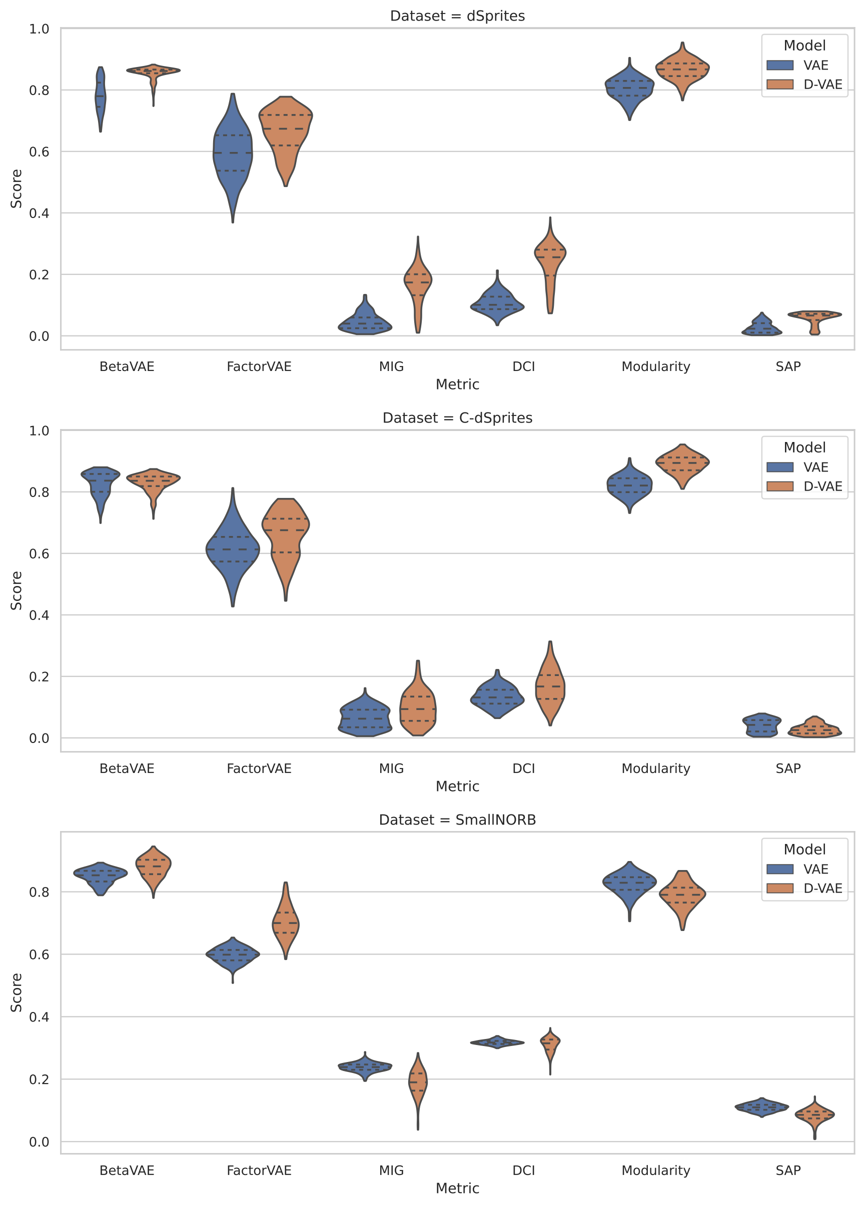


Figure B.4: A comparison of unregularized Gaussian VAE and the discrete VAE w.r.t. the 6 disentanglement metrics on dSprites, C-dSprites, SmallNORB.

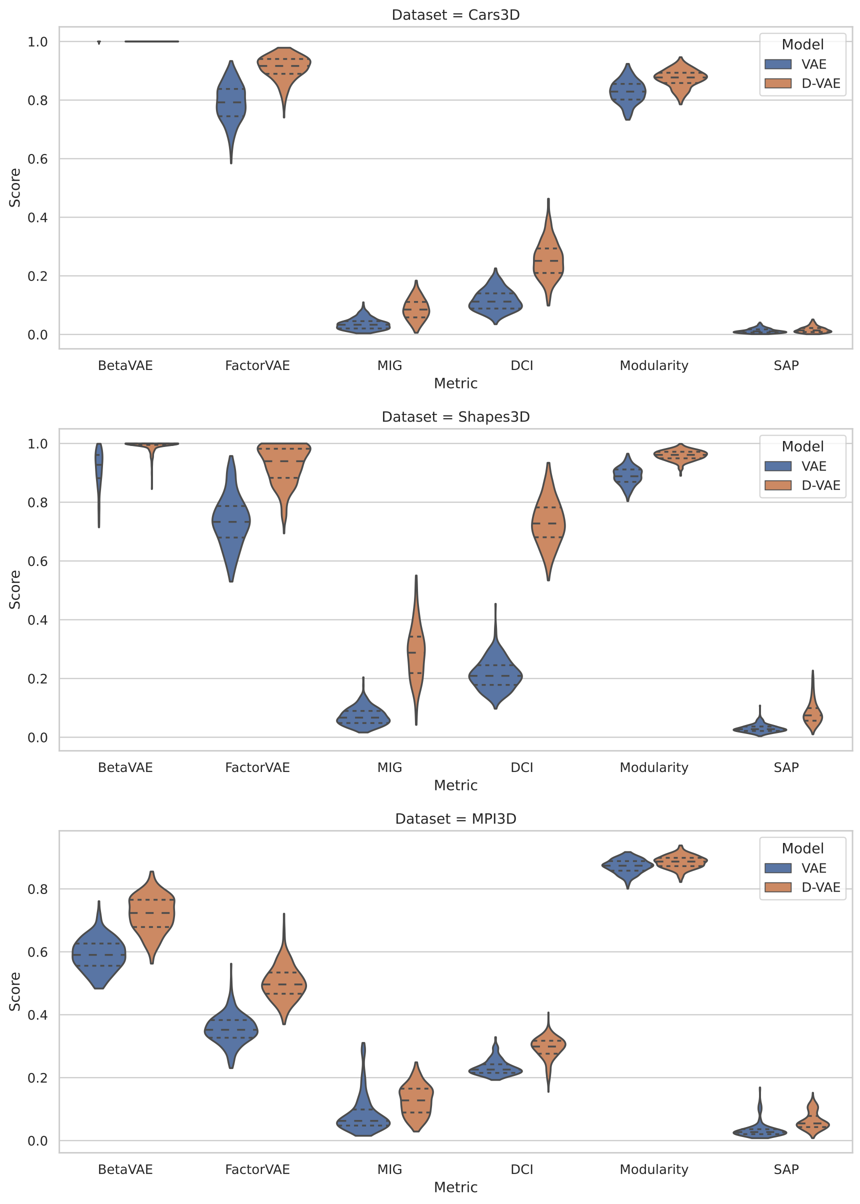


Figure B.5: A comparison of unregularized Gaussian VAE and the discrete VAE w.r.t. the 6 disentanglement metrics on Cars3D, Shapes3D, MPI3D.