On Impact of Network Architecture for Deep Learning

by

Hao Fu

Department of Electrical and Computer Engineering
Duke University

Date: _____

Approved:

_____
Yiran Chen, Supervisor

_____
Hai Li

_____
Neil Zhenqiang Gong

_____
Vahid Tarokh

_____
David Carlson

Dissertation submitted in partial fulfillment of the
requirements for the degree of Doctor of Philosophy
in the Department of Electrical and Computer Engineering
in the Graduate School of
Duke University

2023

ABSTRACT

On Impact of Network Architecture for Deep Learning

by

Hao Fu

Department of Electrical and Computer Engineering
Duke University

Date: _____

Approved:

_____
Yiran Chen, Supervisor

_____
Hai Li

_____
Neil Zhenqiang Gong

_____
Vahid Tarokh

_____
David Carlson

An abstract of a dissertation submitted in partial fulfillment of the
requirements for the degree of Doctor of Philosophy
in the Department of Electrical and Computer Engineering
in the Graduate School of
Duke University

2023

Abstract

The architecture of neural networks is a crucial factor in the success of deep learning models across a range of fields, including computer vision and natural language processing (NLP). Specific architectures are tailored to address particular tasks, and the selection of architecture can significantly affect the training process, model performance, and robustness.

In the field of NLP, we address the training deficiency of text VAEs with autoregressive decoders through two approaches. First, we introduce a cyclical annealing schedule that enables progressive learning of meaningful latent codes by leveraging informative representations from previous cycles as warm restarts. Second, we propose semi-implicit (SI) representations for the latent distributions of natural languages, which extend the commonly used Gaussian distribution family by mixing the variational parameter with a flexible implicit distribution. Our proposed methods are demonstrated to be effective in text generation tasks such as dialog response generation, with significant performance improvements compared to other training techniques.

In the field of computer vision, we investigate the intrinsic influence of network structure on a model's robustness in addressing data distribution shifts. We propose a novel paradigm, Dense Connectivity Search of Outlier Detector (DCSOD), that automatically explores the dense connectivity of CNN architectures on Out-of-Distribution (OOD) detection tasks using Neural Architecture Search (NAS). To improve the quality of evaluation on OOD detection during the search, we propose evolving distillation based on our multi-view feature learning explanation. Experimental results show that DCSOD achieves remarkable performance over widely used architectures and previous NAS baselines.

Acknowledgements

The journey of my PhD study is full of challenges, yet also exciting moments, made possible with the valuable help of my advisors, lab mates, collaborators, friends, and family.

Firstly, I would like to express my sincere gratitude to both my current advisor, Prof. Yiran Chen, and my former advisor Prof. Lawrence Carin, for their invaluable guidance, mentoring, supervision, and financial support. I would also like to extend my thanks to the other members of my committee, Prof. Li, Prof. Tarokh, Prof. Carlson, Prof. Gong, for their time and efforts serving as my committee members, providing crucial advice that contributed significantly to the completion of this dissertation.

Secondly, I would like to show my appreciation to all of my co-authors and collaborators who provided invaluable insights and assistance to my research study. Without their contributions, I would not have been able to complete my work. In chronological order, they are: Chunyuan Li, Yunchen Pu, Zhe Gan, Yizhe Zhang, Chenyang Tao, Yulai Cong, Hongteng Xu, Qian Yang, Dong Wang, Wenlin Wang, Dinghan Shen, Guoyin Wang, Jianqiao Li, Kevin J Liang, Ruiyi Zhang, Shuyang Dai, Liqun Chen, Yitong Li, Xinyuan Zhang, Ke Bai, Pengyu Cheng, Siyang Yuan, Yuewei Yang, Xiaodong Liu, Jianfeng Gao, Weizhu Chen, Jiang Liu, Peggy Yao, Tunhou Zhang, Jingyang Zhang, Randy Linderman, Shiyu Li, Jingwei Sun, Wei Wen, Ang Li.

Last but not least, I would like to extend my deepest appreciation to my parents, my sister, and my friends who have always stood by my side, providing me with their unwavering love, care, and support throughout my PhD studies. I feel immensely fortunate and grateful to have had all of you in my life during this journey.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Over the past decade, deep learning has had a significant impact on various fields including computer vision [1], natural language processing (NLP) [2], recommender systems [3], autonomous driving [4], biomedical and healthcare [5, 6]. The sophisticated design of neural network architecture is the driving force behind the huge success of deep learning. In computer vision, Convolutional Neural Network (CNN) [7] architectures have achieved remarkable results, such as VGG [8], ResNet [9], and DenseNet [10] for classification tasks, and Fully Convolutional Networks (FCN) [11], U-Net [12], and Mask R-CNN [13] for segmentation tasks, as well as Faster R-CNN [14] and YOLO [15] for object detection. In NLP area, recurrent neural networks (RNNs), including GRU [16], LSTM [17], and Bi-LSTM [18], were invented to process sequential data. Transformers [19] were proposed for advanced machine translation systems, and more recently, large-scale pre-trained transformers such as BERT [20] and Roberta [21] for language understanding, Generative Pre-trained Transformer (GPT) [22, 23, 24] for text generation, and BART [25] and T5 [26] for text-to-text generation tasks, have been widely applied in general NLP tasks. Additionally, neural architecture search (NAS) [27], which automatically searches for optimal structures, has emerged as an alternative to hand-crafted architectures that rely on human expertise. Popular NAS methods include reinforcement learning [28, 29], differentiable-based search [30, 31], predictor-based search [32, 33], and one-shot search [34, 35], which have delivered state-of-the-art vision models.

Despite the advancements in architecture design, there are still several challenges that need to be addressed in both NLP and vision areas. One major challenge is training deep generative models to generate text sequences. Text generative models can capture the global features of natural languages using continuous latent variables, providing an effective mechanism for abstract reasoning with text. However, popular auto-regressive decoders,

including RNNs and transformers, have exhibited training deficiencies. Addressing this issue is crucial for enabling controllable text generation. In the vision area, most architecture design focuses on improving accuracy, and the robustness of these models in addressing data distribution shifts is rarely studied. The reliability of machine learning systems depends not only on their accuracy but also on their ability to detect and handle outliers successfully. Therefore, a systematic approach that thoroughly studies the relationship between CNN architecture designs and outlier detectors would be beneficial in pushing the boundaries of outlier detection.

1.1 Problem Definition and Contributions

Training Deficiency in VAE Variational autoencoders (VAEs) [36] with auto-regressive decoders have been successfully applied in various natural language processing (NLP) tasks. However, one significant training deficiency that arises with auto-regressive decoders is the Kullback–Leibler (KL) term in training objective tend to vanish [37]. This can result in learned latent codes that are almost identical to the prior distribution. To address this issue, we proposed two effective methods in this dissertation.

First, we investigated scheduling schemes for the weighting hyper-parameter β before the KL regularization term. Our analysis revealed that KL vanishing occurs due to the lack of informative latent codes during the early stages of optimization. To overcome this, we introduced a cyclical annealing schedule that allows for the progressive learning of meaningful latent codes by leveraging informative representations from previous cycles as warm re-starts. We validated the effectiveness of cyclical annealing on various NLP tasks, including language modeling, dialog response generation, and unsupervised language pre-training.

Second, we hypothesized that the Gaussian approximate posterior used in variational inference may contribute to the training deficiency. To address this, we introduced semi-implicit (SI) representations for the latent distributions of natural languages. This approach

extends the commonly used Gaussian distribution family by mixing the variational parameter with a flexible implicit distribution. Our experiments demonstrated that SI provides more informative latent codes for language modeling and generates more diverse dialog responses, indicating its superior representational power compared to the Gaussian distribution.

NAS for OOD Detection Recent advances in Out-of-Distribution (OOD) Detection [38] is the driving force behind safe and reliable deployment of Convolutional Neural Networks (CNNs) in real-world applications. However, existing studies focus on OOD detection through confidence score and deep generative model-based methods, without considering the impact of DNN structures, especially dense connectivity in architecture fabrications. In addition, existing outlier detection approaches exhibit high variance in generalization performance, lacking stability and confidence in evaluating and ranking different outlier detectors. In this work, we propose a novel paradigm, Dense Connectivity Search of Outlier Detector (DCSOD), that automatically explore the dense connectivity of CNN architectures on near-OOD detection task using Neural Architecture Search (NAS). We introduce a hierarchical search space containing versatile convolution operators and dense connectivity, allowing a flexible exploration of CNN architectures with diverse connectivity patterns. To improve the quality of evaluation on OOD detection during search, we propose evolving distillation based on our multi-view feature learning explanation. Evolving distillation stabilizes training for OOD detection evaluation, thus improves the quality of search. We thoroughly examine DCSOD on CIFAR benchmarks under OOD detection protocol.Experimental results show that DCSOD achieve remarkable performance over widely used architectures and previous NAS baselines. Notably, DCSOD achieves state-of-the-art (SOTA) performance on CIFAR benchmark, with AUROC improvement of $\sim$1.0%.

1.2 Dissertation Outline

The remainder of this dissertation is organized as follows:

Chapter 2 provides a novel explanation for the KL-vanishing issue in VAEs. Specifically, we introduce the concept of "two paths generation" to help develop an understanding of the strengths and weaknesses of existing scheduling methods. We further propose cyclical annealing schedule that allows for the progressive learning of informative latent codes by leveraging the representations of previous cycles as warm re-starts. We demonstrate the effectiveness of our proposed method on a broad range of tasks, including language modeling, dialog response generation, and unsupervised language pre-training.

Chapter 3 develops a semi-implicit (SI) approximation for posterior inference in VAEs. The SI representation provides a trade-off between expressive power and computational cost, which is controlled by a hyper-parameter K. We use this to derive an efficient inference procedure that mitigates the KL vanishing issue with SI while maintaining low computational cost. We demonstrate the effectiveness of our proposed method on two text generation tasks, showing significant performance improvements compared to the standard VAE.

Chapter 4 proposes DCSOD, a novel paradigm that leverages Neural Architecture Search (NAS) to explore optimal outlier detectors within a highly flexible search space. For the high variance issue in out-of-distribution (OOD) detection evaluation, we provide an explanation based on multi-view feature learning. We further propose an outlier distillation loss along with an evolving distillation algorithm to stabilize training and reduce variance in OOD evaluations. The best models crafted by DCSOD achieve state-of-the-art AUROC on CIFAR near-OOD benchmarks, demonstrating the effectiveness of our proposed approach.

Chapter 5 summarizes the dissertation and discusses the future works.

Chapter 2

KL Annealing Methods for Text VAE

2.1 Introduction

Variational autoencoders (VAEs) [36, 39] have been applied in many NLP tasks, including language modeling [37, 40], dialog response generation [41, 42], semi-supervised text classification [43], controllable text generation [44], and text compression [45]. A prominent component of a VAE is the distribution-based latent representation for text sequence observations. This flexible representation allows the VAE to explicitly model holistic properties of sentences, such as style, topic, and high-level linguistic and semantic features. Samples from the prior latent distribution can produce diverse and well-formed sentences through simple deterministic decoding [37].

Due to the sequential nature of text, an auto-regressive decoder is typically employed in the VAE. This is often implemented with a recurrent neural network (RNN); the long short-term memory (LSTM) [17] RNN is used widely. This introduces one notorious issue when a VAE is trained using traditional methods: the decoder ignores the latent variable, yielding what is termed the *KL vanishing* problem.

Several attempts have been made to ameliorate this issue [46, 47, 41, 48]. Among them, perhaps the simplest solution is monotonic KL annealing, where the weight of the KL penalty term is scheduled to gradually increase during training [37]. While these techniques can effectively alleviate the KL-vanishing issue, a proper unified theoretical interpretation is still lacking, even for the simple annealing scheme.

In this work, we analyze the variable dependency in a VAE, and point out that the auto-regressive decoder has two paths (formally defined in Section 2.3.1) that work together to generate text sequences. One path is conditioned on the latent codes, and the other path is conditioned on previously generated words. KL vanishing happens because (i) the first

path can easily get blocked, due to the lack of good latent codes at the beginning of decoder training; (*ii*) the easiest solution that an expressive decoder can learn is to ignore the latent code, and relies on the other path only for decoding. To remedy this issue, a promising approach is to remove the blockage in the first path, and feed meaningful latent codes in training the decoder, so that the decoder can easily adopt them to generate controllable observations [37].

This work makes the following contributions:

- We provide a novel explanation for the KL-vanishing issue, and develop an understanding of the strengths and weaknesses of existing scheduling methods (*e.g.*, constant or monotonic annealing schedules).
- Based on our explanation, we propose a cyclical annealing schedule. It repeats the annealing process multiple times, and can be considered as an inexpensive approach to leveraging good latent codes learned in the previous cycle, as a warm restart, to train the decoder in the next cycle.
- We demonstrate that the proposed cyclical annealing schedule for VAE training improves performance on a large range of tasks (with negligible extra computational cost), including text modeling, dialog response generation, and unsupervised language pre-training.

2.2 Preliminaries

2.2.1 The VAE model

To generate a text sequence of length T, $\boldsymbol{x} = [x_1, \cdots, x_T]$, neural language models [49] generate every token x_t conditioned on the previously generated tokens:

$$p(\boldsymbol{x}) = \prod_{t=1}^{T} p(x_t|x_{<t}),$$

where $x_{<t}$ indicates all tokens before t.

The VAE model for text consists of two parts, generation and inference [36, 39, 37]. The *generative model* (*decoder*) draws a continuous latent vector $\boldsymbol{z}$ from prior $p(\boldsymbol{z})$, and generates the text sequence $\boldsymbol{x}$ from a conditional distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$; $p(\boldsymbol{z})$ is typically assumed a multivariate Gaussian, and $\boldsymbol{\theta}$ represents the neural network parameters. The following auto-regressive decoding process is usually used:

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) = \prod_{t=1}^{T} p_{\boldsymbol{\theta}}(x_t|x_{<t}, \boldsymbol{z}). \tag{2.1}$$

Parameters $\boldsymbol{\theta}$ are typically learned by maximizing the marginal log likelihood $\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \log \int p(\boldsymbol{z}) p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) \mathrm{d}\boldsymbol{z}$. However, this marginal term is intractable to compute for many decoder choices. Thus, variational inference is considered, and the true posterior $p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}) \propto p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})$ is approximated via the variational distribution $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ is (often known as the *inference model* or *encoder*), implemented via a $\boldsymbol{\phi}$-parameterized neural network. It yields the *evidence lower bound* (ELBO) as an objective:

$$\begin{aligned} &\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) \geq \mathcal{L}_{\text{ELBO}} = \\ &\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\big[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})\big] - \mathrm{KL}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})) \end{aligned} \tag{2.2}$$

Typically, $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ is modeled as a Gaussian distribution, and the re-parametrization trick is used for efficient learning [36].

2.2.2 Training Schedules and KL Vanishing

There is an alternative interpretation of the ELBO: the VAE objective can be viewed as a regularized version of the autoencoder (AE) [50]. It is thus natural to extend the negative of $\mathcal{L}_{\text{ELBO}}$ in (2.2) by introducing a hyper-parameter β to control the strength of regularization:

$$\mathcal{L}_\beta = \mathcal{L}_E + \beta \mathcal{L}_R, \quad \text{with} \tag{2.3}$$

$$\mathcal{L}_E = -\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\big[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})\big] \tag{2.4}$$

$$\mathcal{L}_R = \mathrm{KL}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})) \tag{2.5}$$

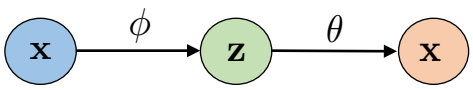
where $\mathcal{L}_E$ is the reconstruction error (or negative log-likelihood (NLL)), and $\mathcal{L}_R$ is a KL regularizer.

The cost function $\mathcal{L}_\beta$ provides a unified perspective for understanding various autoencoder variants and training methods. When $\beta = 1$, we recover the VAE in (2.2). When $\beta = 0$, and $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ is a delta distribution, we recover the AE. In other words, the AE does not regularize the variational distribution toward a prior distribution, and there is only a point-estimate to represent the text sequence's latent feature. In practice, it has been found that learning with an AE is prone to overfitting [37], or generating plain dialog responses [41]. Hence, it is desirable to retain meaningful posteriors in real applications. Two different schedules for β have been commonly used for a text VAE.

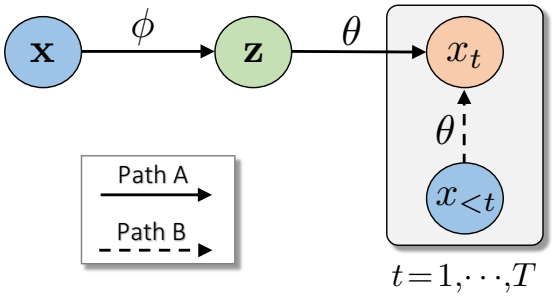
Constant Schedule The standard approach is to keep $\beta = 1$ fixed during the entire training procedure, as it corresponds to optimizing the true VAE objective. Unfortunately, instability on text analysis has been witnessed, in that the KL term $\mathcal{L}_R$ becomes vanishingly small during training [37]. This issue causes two undesirable outcomes: (*i*) an encoder that produces posteriors almost identical to the Gaussian prior, for all observations (rather than a more interesting posterior); and (*ii*) a decoder that completely ignores the latent variable $\boldsymbol{z}$, and a learned model that reduces to a simpler language model. This is known as the *KL vanishing* issue in text VAEs.

Monotonic Annealing Schedule. A simple remedy has been proposed in [37] to alleviate KL collapse. It sets $\beta = 0$ at the beginning of training, and gradually increases β until $\beta = 1$ is reached. In this setting, we do not optimize the proper lower bound in (2.2) during the early stages of training, but nonetheless improvements on the value of that bound are observed at convergence in previous work [37, 41].

The monotonic annealing schedule has become the *de facto* standard in training text VAEs, and has been widely adopted in many NLP tasks. Though simple and often effective, this heuristic still lacks a proper justification. Further, how to best schedule β is largely unexplored.



(a) Traditional VAE



(b) VAE with an auto-regressive decoder

Figure 2.1: (a) Standard VAE learning, with only one path via $\{\boldsymbol{\phi}, \boldsymbol{\theta}\}$ from $\boldsymbol{x}$ to its reconstruction; (b) VAE learning with an auto-regressive decoder.

2.3 Cyclical Annealing Schedule

2.3.1 Identifying Sources of KL Vanishing

The learning parameters $\{\boldsymbol{\phi}, \boldsymbol{\theta}\}$ in the two different paradigms is illustrated in Figure 2.1. Starting from the observation $\boldsymbol{x}$ in blue circle, a VAE infers its latent code $\boldsymbol{z}$ in the green circle, and further generates its reconstruction in the red circle. In the traditional VAE [36], $\boldsymbol{z}$ generates $\boldsymbol{x}$ directly, and the reconstruction depends only on one path of $\{\boldsymbol{\phi}, \boldsymbol{\theta}\}$ passing through $\boldsymbol{z}$, as shown in Figure 2.1(a). Hence, $\boldsymbol{z}$ can largely determine the reconstructed $\boldsymbol{x}$. In contrast, when an auto-regressive decoder is used in a text VAE [37], there are two paths from $\boldsymbol{x}$ to its reconstruction, as shown in Figure 2.1(b). Path A is the same as that in the standard VAE, where $\boldsymbol{z}$ is the global representation that controls the generation of $\boldsymbol{x}$; Path B leaks the partial ground-truth information of $\boldsymbol{x}$ at every time step of the sequential decoding. It generates x_t conditioned on $x_{<t}$. Therefore, Path B can potentially bypass Path A to generate $\boldsymbol{x}$, leading to KL vanishing.

From this perspective, we hypothesize that the model-collapse problem is related to the low quality of $\boldsymbol{z}$ at the beginning phase of decoder training. A lower quality $\boldsymbol{z}$ introduces more difficulties in reconstructing $\boldsymbol{x}$ via Path A. As a result, the model is forced to learn an easier solution to decoding: generating $\boldsymbol{x}$ via Path B only.

We argue that this phenomenon can be easily observed due to the powerful representation capability of the auto-regressive decoder. It has been shown empirically that auto-regressive decoders are able to capture highly-complex distributions, such as natural language sentences [49]. This means that Path B alone has enough capacity to model $\boldsymbol{x}$, even though the decoder takes $\{x_{<t}, \boldsymbol{z}\}$ as input to produce x_t. Zhang et al. [51] has shown that flexible deep neural networks can easily fit randomly labeled training data, and here the decoder can learn to rely solely on $x_{<t}$ for generation, when $\boldsymbol{z}$ is of low quality.

We use our hypothesis to explain the learning behavior of different scheduling schemes for β as follows.

Constant Schedule The two loss terms in (2.2) are weighted equally in the constant schedule. At the early stage of optimization, $\{\boldsymbol{\phi}, \boldsymbol{\theta}\}$ are randomly initialized and the latent codes $\boldsymbol{z}$ are of low quality. The KL term $\mathcal{L}_R$ pushes $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ close to an uninformative prior $p(\boldsymbol{z})$: the posterior becomes more like an isotropic Gaussian noise, and less representative of their corresponding observations. In other words, $\mathcal{L}_R$ blocks Path A, and thus $\boldsymbol{z}$ remains uninformative during the entire training process: it starts with random initialization and then is regularized towards a random noise. Although the reconstruction term $\mathcal{L}_E$ can be satisfied via two paths, since $\boldsymbol{z}$ is noisy, the decoder learns to discard Path A (*i.e.,* ignores $\boldsymbol{z}$), and chooses Path B to generate the sentence word-by-word.

Monotonic Annealing Schedule The monotonic schedule sets β close to 0 in the early stage of training, which effectively removes the blockage $\mathcal{L}_R$ on Path A, and the model reduces to a denoising autoencoder [1]. $\mathcal{L}_E$ becomes the only objective, which can be reached by both paths. Though randomly initialized, $\boldsymbol{z}$ is learned to capture useful information for

[1]The Gaussian sampling remains for $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$

reconstruction of $\boldsymbol{x}$ during training. At the time when the full VAE objective is considered ($\beta = 1$), $\boldsymbol{z}$ learned earlier can be viewed as the VAE initialization; such latent variables are much more informative than random, and thus are ready for the decoder to use.

To mitigate the KL-vanishing issue, it is key to have meaningful latent codes $\boldsymbol{z}$ at the beginning of training the decoder, so that $\boldsymbol{z}$ can be utilized. The monotonic schedule under-weights the prior regularization, and the learned $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ tends to collapse into a point estimate (*i.e.,* the VAE reduces to an AE). This underestimate can result in sub-optimal decoder learning. A natural question concerns how one can get a better distribution estimate for $\boldsymbol{z}$ as initialization, while retaining low computational cost.

2.3.2 Cyclical Annealing Schedule

Our proposal is to use $\boldsymbol{z} \sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$, which has been trained under the full VAE objective, as initialization. To learn to progressively improve latent representation $\boldsymbol{z}$, we propose a cyclic annealing schedule. We start with $\beta = 0$, increase β at a fast pace, and then stay at $\beta = 1$ for subsequent learning iterations. This encourages the model to converge towards the VAE objective, and infers its first raw full latent distribution.

Unfortunately, Path A is blocked at $\beta = 1$. The optimization is then continued at $\beta = 0$ again, which perturbs the VAE objective, dislodges it from the convergence, and reopens Path A. Importantly, the decoder is now trained with the latent code from a full distribution $\boldsymbol{z} \sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$, and both paths are considered. We repeat this process several times to achieve better convergences.

Formally, β has the form:

$$\beta_t = \begin{cases} f(\tau), & \tau \leq R \\ 1, & \tau > R \end{cases} \quad \text{with} \tag{2.6}$$

$$\tau = \frac{\mathrm{mod}(t-1, \lceil T/M \rceil)}{T/M} \ , \tag{2.7}$$

where t is the iteration number, T is the total number of training iterations, f is a monotonically increasing function, and we introduce two new hyper-parameters associated with

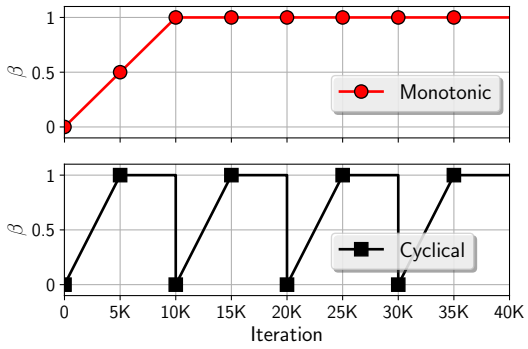


Figure 2.2: Comparison between (a) traditional monotonic and (b) proposed cyclical annealing schedules. In this figure, $M = 4$ cycles and $R = 0.5$.

the cyclical annealing schedule:

- M: number of cycles (default $M = 4$);
- R: proportion used to increase β within a cycle (default $R = 0.5$).

In other words, we split the training process into M cycles, each starting with $\beta = 0$ and ending with $\beta = 1$. We provide an example of a cyclical schedule in Figure 2.2(b), compared with the monotonic schedule in Figure 2.2(a). Within one cycle, there are two consecutive stages (divided by R):

- **Annealing**. β is annealed from 0 to 1 in the first $R\lceil T/M \rceil$ training steps over the course of a cycle. For example, the steps $[1, 5K]$ in the Figure 2.2(b). $\beta = f(0) = 0$ forces the model to learn representative $\boldsymbol{z}$ to reconstruct $\boldsymbol{x}$. As depicted in Figure 2.1(b), there is no interruption from the prior on Path A, $\boldsymbol{z}$ is forced to learn the global representation of $\boldsymbol{x}$. By gradually increasing β towards $f(R) = 1$, $q(\boldsymbol{z}|\boldsymbol{x})$ is regularized to transit from a point estimate to a distribution estimate, spreading out to match the prior.
- **Fixing**. As our ultimate goal is to learn a VAE model, we fix $\beta = 1$ for the rest of training steps within one cycle, *e.g.,* the steps $[5K, 10K]$ in Figure 2.2(b). This drives the model to optimize the full VAE objective until convergence.

As illustrated in Figure 2.2, the monotonic schedule increasingly anneals β from 0 to 1 once, and fixes $\beta = 1$ during the rest of training. The cyclical schedules alternatively repeats the annealing and fixing stages multiple times.

A Practical Recipe The existing schedules can be viewed as special cases of the proposed cyclical schedule. The cyclical schedule reduces to the constant schedule when $R = 0$, and it reduces to an monotonic schedule when $M = 1$ and R is relatively small [2]. In theory, any monotonically increasing function f can be adopted for the cyclical schedule, as long as $f(0) = 0$ and $f(R) = 1$. In practice, we suggest to build the cyclical schedule upon the success of monotonic schedules: we adopt the same f, and modify it by setting M and R (as default). Three widely used increasing functions for f are linear [52, 53], Sigmoid [37] and Consine [54]. We present the comparative results using the linear function $f(\tau) = \tau/R$ in Figure 2.2, and show the complete comparison for other functions in Figure 2.7 of the Supplementary Material (SM).

2.3.3 On the impact of β

This section derives a bound for the training objective to rigorously study the impact of β; the proof details are included in SM. For notational convenience, we identify each data sample with a unique integer index $n \sim q(n)$, drawn from a uniform random variable on $\{1, 2, \cdots, N\}$. Further we define $q(\boldsymbol{z}|n) = q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}_n)$ and $q(\boldsymbol{z}, n) = q(\boldsymbol{z}|n)q(n) = q(\boldsymbol{z}|n)\frac{1}{N}$. Following [55], we refer to $q(\boldsymbol{z}) = \sum_{n=1}^{N} q(\boldsymbol{z}|n)q(n)$ as the aggregated posterior. This marginal distribution captures the aggregated $\boldsymbol{z}$ over the entire dataset. The KL term in (3.5) can be decomposed into two refined terms [56, 57]:

$$\begin{aligned} \mathcal{F}_R &= \mathbb{E}_{q(n)}[\mathrm{KL}(q(\boldsymbol{z}|n)||p(\boldsymbol{z}))] \\ &= \underbrace{I_q(\boldsymbol{z}, n)}_{\mathcal{F}_1\text{: Mutual Info.}} + \underbrace{\mathrm{KL}(q(\boldsymbol{z})||p(\boldsymbol{z}))}_{\mathcal{F}_2\text{: Marginal KL}} \end{aligned} \tag{2.8}$$

[2] In practice, the monotonic schedule usually anneals in a very fast pace, thus R is small compared with the entire training procedure.

where $\mathcal{F}_1$ is the mutual information (MI) measured by q. Higher MI can lead to a higher correlation between the latent variable and data variable, and encourages a reduction in the degree of KL vanishing. The marginal KL is represented by $\mathcal{F}_2$, and it measures the fitness of the aggregated posterior to the prior distribution.

The reconstruction term in (3.5) provides a lower bound for MI measured by q, based on Corollary 3 in [58]:

$$\begin{aligned}\mathcal{F}_E &= E_{q(n),\boldsymbol{z}\sim q(\boldsymbol{z}|n)}(\log p(n|\boldsymbol{z}))] + H_q(n) \\ &\leq I_q(\boldsymbol{z}, n)\end{aligned} \tag{2.9}$$

where $H(n)$ is a constant.

Analysis of β When scheduled with β, the training objective over the dataset can be written as:

$$\mathcal{F} = -\mathcal{F}_E + \beta\mathcal{F}_R \tag{2.10}$$

$$\geq (\beta - 1)I_q(\boldsymbol{z}, n) + \beta\mathrm{KL}(q(\boldsymbol{z})||p(\boldsymbol{z})) \tag{2.11}$$

To reduce KL vanishing, we desire an increase in the MI term $I(\boldsymbol{z}, n)$, which appears in both $\mathcal{F}_E$ and $\mathcal{F}_R$, modulated by β. It shows that reducing KL vanishing is inversely proportional with β. When $\beta = 0$, the model fully focuses on maximizing the MI. As β increases, the model gradually transits towards fitting the aggregated latent codes to the given prior. When $\beta = 1$, the implementation of MI becomes implicit in $\mathrm{KL}(q(\boldsymbol{z})||p(\boldsymbol{z}))$. It is determined by the amortized inference regularization (implied by the encoder's expressivity) [59], which further affects the performance of the generative density estimator.

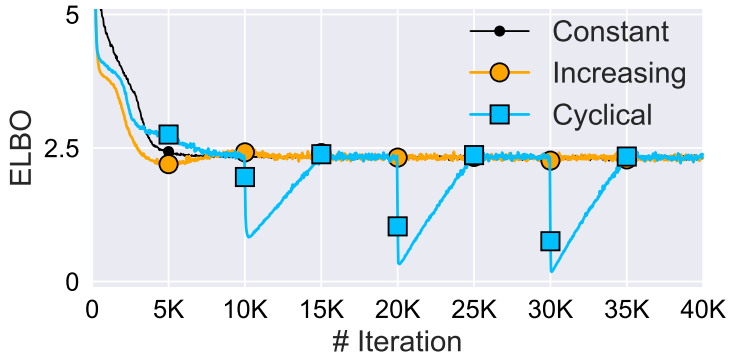
2.4 Visualization of Latent Space

We compare different schedule methods by visualizing the learning processes on an illustrative problem. Consider a dataset consisting of 10 sequences, each of which is a

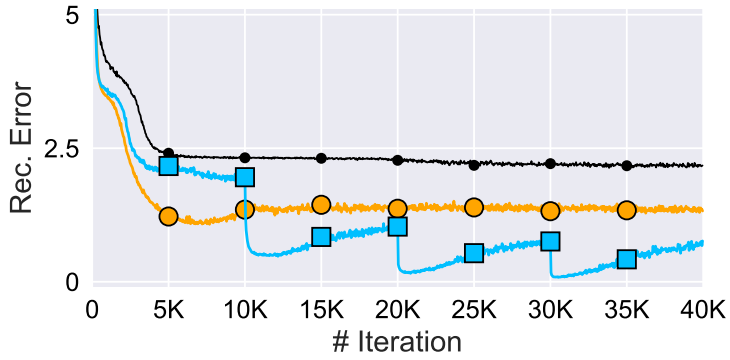
10-dimensional one-hot vector with the value 1 appearing in different positions. A 2-dimensional latent space is used for the convenience of visualization. Both the encoder and decoder are implemented using a 2-layer LSTM with 64 hidden units each. We use $T = 40$K total iterations, and the scheduling schemes in Figure 2.2.

The learning curves for the ELBO, reconstruction error, and KL term are shown in Figure 2.3. The three schedules share very similar values. However, the cyclical schedule provides substantially lower reconstruction error and higher KL divergence. Interestingly, the cyclical schedule improves the performance progressively: it becomes better than the previous cycle, and there are clear periodic patterns across different cycles. This suggests that the cyclical schedule allows the model to use the previously learned results as a warm-restart to achieve further improvement.

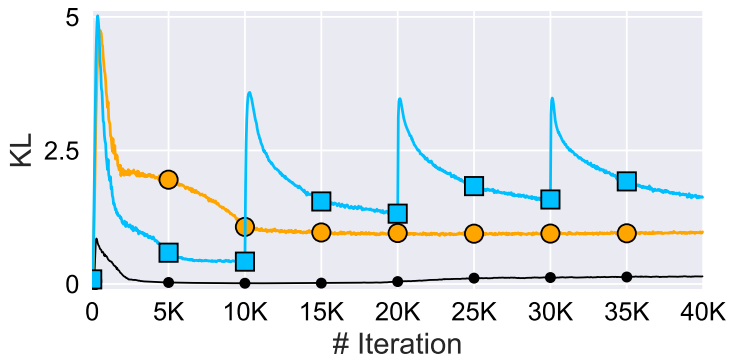
We visualize the resulting division of the latent space for different training steps in Figure 2.4, where each color corresponds to $\boldsymbol{z} \sim q(\boldsymbol{z}|n)$, for $n = 1, \cdots, 10$. We observe that the constant schedule produces heavily mixed latent codes $\boldsymbol{z}$ for different sequences throughout the entire training process. The monotonic schedule starts with a mixed $\boldsymbol{z}$, but soon divides the space into a mixture of 10 cluttered Gaussians in the annealing process (the division remains cluttered in the rest of training). The cyclical schedule behaves similarly to the monotonic schedule in the first 10K steps (the first cycle). But, starting from the 2nd cycle, much more divided clusters are shown when learning on top of the 1st cycle results. However, $\beta < 1$ leads to some holes between different clusters, making $q(\boldsymbol{z})$ violate the constraint of $p(\boldsymbol{z})$. This is alleviated at the end of the 2nd cycle, as the model is trained with $\beta = 1$. As the process repeats, we see clearer patterns in the 4th cycle than the 2nd cycle for both $\beta < 0$ and $\beta = 1$. It shows that more structured information is captured in $\boldsymbol{z}$ using the cyclical schedule, which is beneficial in downstream applications as shown in the experiments.



(a) ELBO



(b) Reconstruction Error



(c) KL term

Figure 2.3: Comparison of the learning curves for the three schedules on an illustrative problem.

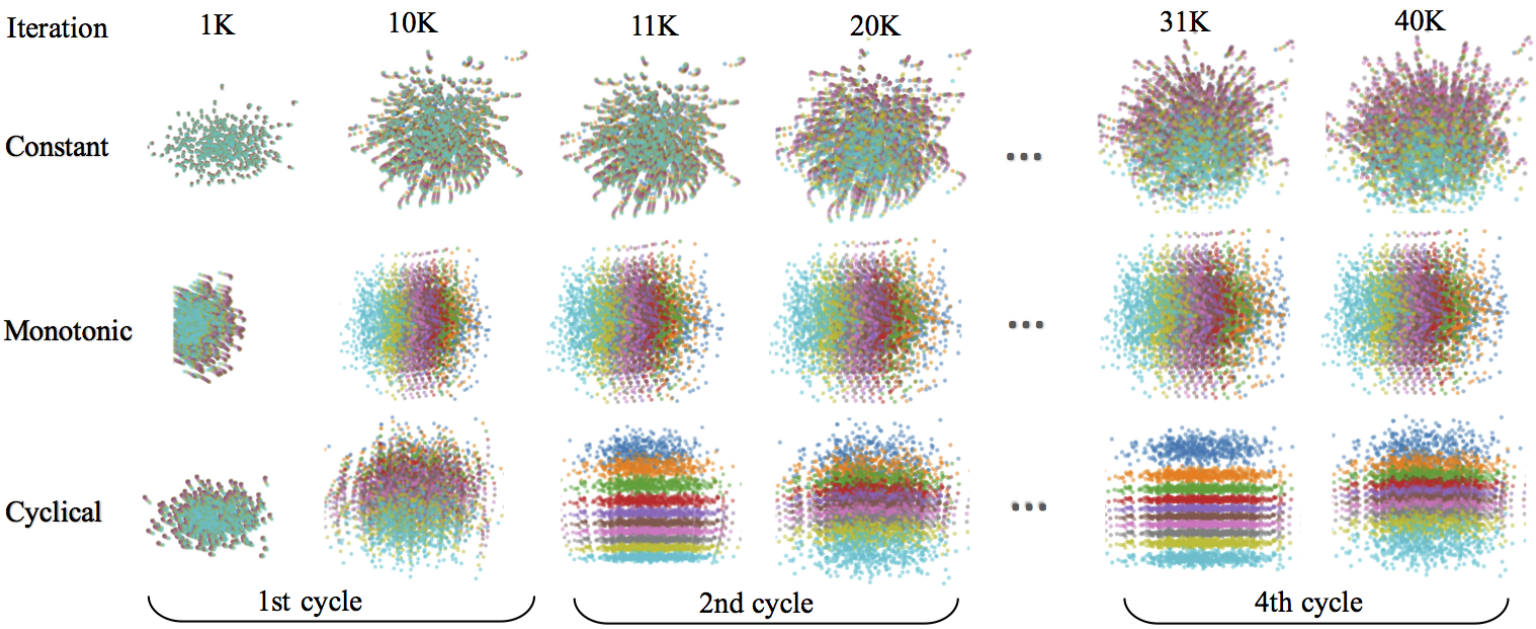


Figure 2.4: Visualization of the latent space along the learning dynamics on an illustrative problem.

2.5 Related Work

Solutions to KL vanishing Several techniques have been proposed to mitigate the KL vanishing issue. The proposed method is most closely related to the monotonic KL annealing technique in [37]. In addition to introducing a specific algorithm, we have comprehensively studied the impact of β and its scheduling schemes. Our explanations can be used to interpret other techniques, which can be broadly categorized into two classes.

The first category attempts to weaken Path B, and force the decoder to use Path A. Word drop decoding [37] sets a certain percentage of the target words to zero. It has shown that it may degrade the performance when the drop rate is too high. The dilated CNN was considered in [46] as a new type of decoder to replace the LSTM. By changing the decoder's dilation architecture, one can control Path B: the effective context from $x_{<t}$.

The second category of techniques improves the dependency in Path A, so that the decoder uses latent codes more easily. Skip connections were developed in [47] to shorten the paths from $\boldsymbol{z}$ to $\boldsymbol{x}$ in the decoder. **(author?)** [41] introduced an auxiliary loss that requires the decoder to predict the bag-of-words in the dialog response [41]. The decoder is thus forced to capture global information about the target response. Zhao et al. [60] enhanced Path A via mutual information. Concurrent with our work, **(author?)** [61] proposed to update encoder multiple times to achieve better latent code before updating decoder. Semi-amortized training [48] was proposed to perform stochastic variational inference (SVI) [62] on top of the amortized inference in VAE. It shares a similar motivation with the proposed approach, in that better latent codes can reduce KL vanishing. However, the computational cost to run SVI is high, while our monotonic schedule does not require any additional compute overhead. The KL scheduling methods are complementary to these techniques. As shown in experiments, the proposed cyclical schedule can further improve them.

β-VAE The VAE has been extended to β-regularized versions in a growing body of work [63, 64]. Perhaps the seminal work is β-VAE [63], which was extended in [65, 56] to consider β on the refined terms in the KL decomposition. Their primary goal is to

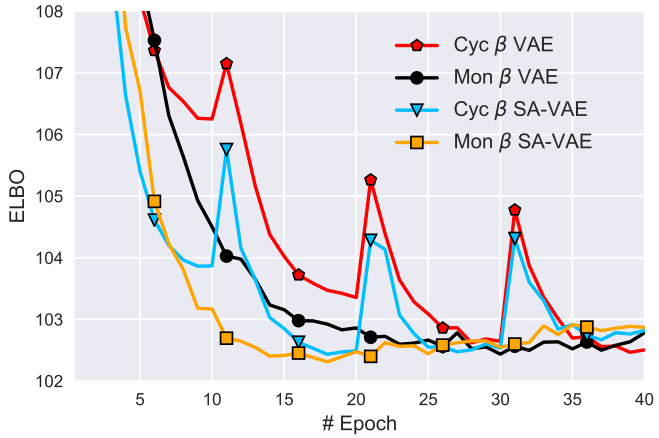
learn disentangled latent representations to explain the data, by setting $\beta > 1$. From an information-theoretic point of view, [64] suggests a simple method to set $\beta < 1$ to ensure that latent-variable models with powerful stochastic decoders do not ignore their latent code. However, $\beta \neq 1$ results in an improper statistical model. Further, β is static in their work; we consider dynamically scheduled β and find it more effective.

Cyclical schedules Warm-restart techniques are common in optimization to deal with multimodal functions. The cyclical schedule has been used to train deep neural networks [66], warm restart stochastic gradient descent [67], improve convergence rates [68], obtain model ensembles [69] and explore multimodal distributions in MCMC sampling [70]. All these works applied cyclical schedules to the learning rate. In contrast, this work represents the first to consider the cyclical schedule for β in VAE. Though the techniques seem simple and similar, our motivation is different: we use the cyclical schedule to re-open Path A in Figure 2.1(b) and provide the opportunity to train the decoder with high-quality $\boldsymbol{z}$.

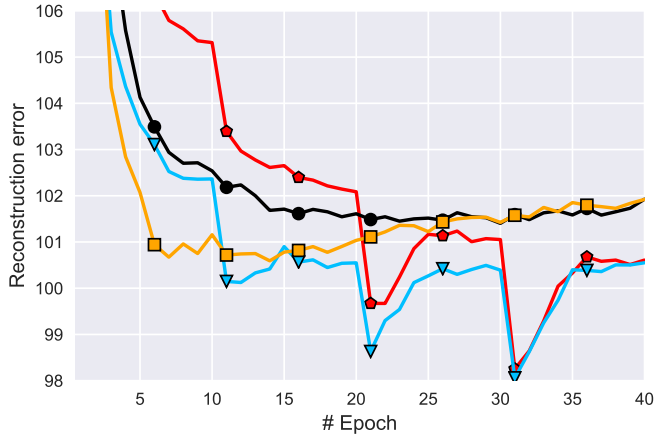
2.6 Experiments

The source code to reproduce the experimental results will be made publicly available on GitHub[3]. For a fair comparison, we follow the practical recipe described in Section 3.2, where the monotonic schedule is treated as a special case of cyclical schedule (while keeping all other settings the same). The default hyper-parameters of the cyclical schedule are used in all cases unless stated otherwise. We study the impact of hyper-parameters in the SM, and show that larger M can provide higher performance for various R. We show the major results in this section, and put more details in the SM. The monotonic and cyclical schedules are denoted as **M** and **C**, respectively.

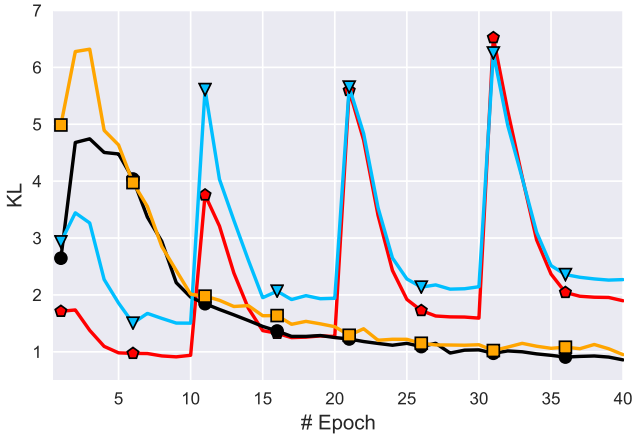
[3]https://github.com/haofuml/cyclical_annealing



(a) ELBO



(b) Reconstruction Error



(c) KL term

Figure 2.5: Learning curves of VAE and SA-VAE on PTB. Under similar ELBO, cyclical schedule provides lower reconstruction errors and higher KL.

Table 2.1: Comparison of language modeling on Penn Tree Bank (PTB). Since SA-VAE tends to overfit, we report its best results in row M*.

Schedule		Rec	KL	ELBO	PPL
VAE	**M**	101.73	0.907	-102.63	108.09
	C	**100.51**	**1.955**	-102.46	107.25
SA-VAE	**M***	100.75	1.796	-102.54	107.64
	M	101.83	1.053	-102.89	109.33
	C	**100.50**	**2.261**	-102.76	108.71
$\beta_m = 0.5$	**M**	97.36	9.605	-106.96	132.57
	C	95.22	9.484	-104.70	118.78

2.6.1 Language Modeling

We first consider language modeling on the Penn Tree Bank (PTB) dataset [71]. Language modeling with VAEs has been a challenging problem, and few approaches have been shown to produce rich generative models that do not collapse to standard language models. Ideally a deep generative model trained with variational inference would pursue higher ELBO, making use of the latent space (*i.e.,* maintain a nonzero KL term) while accurately modeling the underlying distribution (*i.e.,* lower reconstruction errors). We implemented different schedules based on the code[4] published by [48].

The latent variable is 32-dimensional, and 40 epochs are used. We compare the proposed cyclical annealing schedule with the monotonic schedule baseline that, following [37], anneals linearly from 0 to 1.0 over 10 epochs. We also compare with semi-amortized (SA) training [48], which is considered as the state-of-the-art technique in preventing KL vanishing. We set SVI steps to 10.

Results are shown in Table 2.1. The perplexity is reported in column PPL. The cyclical

[4]https://github.com/harvardnlp/sa-vae

Table 2.2: Generated dialog responses from the cyclical and monotonic schedules.

Context `Alice`: yeah you know it's interesting especially when my experience has always been at a public university.			
Topic: Choose a College	**Target** `Bob` (statement): yeah that's right		
	1. yes		1. i'm not sure
	2. oh really		2. and i'm not sure
C	3. and there's a lot of <unk> there's a lot of people	**M**	3. and i'm not sure
	4. yeah		4. i'm not sure
	5. and i think that's probably the biggest problem i've ever seen in the past		5. i'm not sure

schedule outperforms the monotonic schedule for both standard VAE and SA-VAE training. SA-VAE training can effectively reduce KL vanishing, it takes 472s per epoch. However, this is significantly more expensive than the standard VAE training which takes 30s per epoch. The proposed cyclical schedule adds almost zero cost.

We show the learning curves for VAE and SA-VAE in Figure 2.5. Interestingly, the cyclical schedule exhibits periodical learning behaviours. The performance of the cyclical schedule gets better progressively, after each cycle. While ELBO and PPL ar similar, the cyclical schedule improves the reconstruction ability and KL values for both VAE and SA-VAE. We observe clear over-fitting issues for the SA-VAE with the monotonic schedule, while this issue is less severe for SA-VAE with the cyclical schedule.

Finally, we further investigate whether our improvements are from simply having a lower β, rather than from the cyclical schedule re-opening Path A for better learning. To test this, we use a monotonic schedule with maximum $\beta = 0.5$. We observe that the reconstruction and KL terms perform better individually, but the ELBO is substantially worse than $\beta = 1$, because $\beta = 0.5$ yields an improper model. Even so, the cyclical schedule improves its performance.

2.6.2 Conditional VAE for Dialog

We use a cyclical schedule to improve the latent codes in [41], which are key to diverse dialog-response generation. Following [41], Switchboard (SW) Corpus [72] is used, which has 2400 two-sided telephone conversations.

Table 2.3: Comparison on dialog response generation. Reconstruction perplexity (Rec-P) and BLEU (B) scores are used for evaluation.

Model	CVAE		CVAE+BoW	
Schedule	**M**	**C**	**M**	**C**
Rec-P ↓	36.16	**29.77**	18.44	**16.74**
KL Loss ↑	0.265	**4.104**	14.06	**15.55**
B4 prec	0.185	**0.234**	0.211	**0.219**
B4 recall	0.122	**0.220**	0.210	**0.219**
A-bow prec	0.957	**0.961**	0.958	**0.961**
A-bow recall	0.911	**0.941**	0.938	**0.940**
E-bow prec	0.867	0.833	0.830	0.828
E-bow recall	0.784	**0.808**	0.808	0.805

Two latent variable models are considered. The first one is the Conditional VAE (CVAE), which has been shown better than the encoder-decoder neural dialog [73]. The second is to augment VAE with a bag-of-word (BoW) loss to tackle the KL vanishing problem, as proposed in [41].

Table 2 shows the sample outputs generated from the two schedules using CVAE. Caller `Alice` begins with an open-ended statement on choosing a college, and the model learns to generate responses from Caller `Bob`. The cyclical schedule generated highly diverse answers that cover multiple plausible dialog acts. On the contrary, the responses from the monotonic schedule are limited to repeat plain responses, *i.e.,* "*i'm not sure*".

Quantitative results are shown in Table 2.3, using the evaluation metrics from [41]. (*i*) Smoothed Sentence-level BLEU [74]: BLEU is a popular metric that measures the geometric mean of modified n-gram precision with a length penalty. We use BLEU-1 to 4 as our lexical similarity metric and normalize the score to 0 to 1 scale. (*ii*) Cosine Distance of Bag-of-word Embedding [75]: a simple method to obtain sentence embeddings is to

take the average or extreme of all the word embeddings in the sentences. We used Glove embedding and denote the average method as $A-bow$ and extreme method as $E-bow$. The score is normalized to $[0, 1]$. Higher values indicate more plausible responses.

The BoW indeed reduces the KL vanishing issue, as indicated by the increased KL and decreased reconstruction perplexity. When applying the proposed cyclical schedule to CVAE, we also see a reduced KL vanishing issue. Interestingly, it also yields the highest BLEU scores. This suggests that the cyclical schedule can generate dialog responses of higher fidelity with lower cost, as the auxiliary BoW loss is not necessary. Further, BoW can be improved when integrated with the cyclical schedule, as shown in the last column of Table 2.3.

2.6.3 Unsupervised Language Pre-training

We consider the Yelp dataset, as pre-processed in [76] for unsupervised language pre-training. Text features are extracted as the latent codes $\boldsymbol{z}$ of VAE models, pre-trained with monotonic and cyclical schedules. The AE is used as the baseline. A good VAE can learn to cluster data into meaningful groups [36], indicating that well-structured $\boldsymbol{z}$ are highly informative features, which usually leads to higher classification performance. To clearly compare the quality of $\boldsymbol{z}$, we build a simple one-layer classifier on $\boldsymbol{z}$, and fine-tune the model on different proportions of labelled data [77].

The results are shown in Figure 2.6. The cyclical schedule consistently yields the highest accuracy relative to other methods. We visualize the tSNE embeddings [78] of $\boldsymbol{z}$ in Figure 2.9 of the SM, and observe that the cyclical schedule exhibits clearer clustered patterns.

2.6.4 Ablation Study

To enhance the performance, we propose to apply the cyclical schedule to the learning rate η on real tasks. It ensures that the optimizer has the same length of optimization trajectory for each β cycle (so that each cycle can fully converge). To investigate the

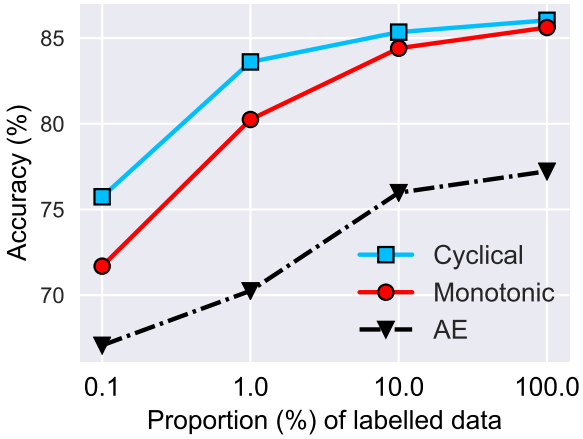


Figure 2.6: Accuracy of fine-tuning on the unsupervised pre-trained models on the Yelp dataset.

Table 2.4: Comparison of cyclical schedules on β and η, tested with language modeling on PTB.

Schedule	**Rec**	**KL**	**ELBO**
Cyc β + Const η	**101.30**	**1.457**	-102.76
Mon β + Const η	101.93	0.858	-102.78
Cyc β + Cyc η	**100.61**	**1.897**	-102.51
Mon β + Cyc η	101.74	0.748	-102.49

impact of cyclical on η, we perform two more ablation experiments: (i) We make only β cyclical, keep η constant. (ii) We make only η cyclical, keep β monotonic. The last epoch numbers are shown in Table 2.4, and the learning curves on shown in Figure 2.10 in SM. Compared with the baseline, we see that it is the cyclical β rather than cyclical η that contributes to the improved performance.

2.7 Conclusions

We provide a novel two-path interpretation to explain the KL vanishing issue, and identify its source as a lack of good latent codes at the beginning of decoder training. This provides an understanding of various β scheduling schemes, and motivates the proposed cyclical schedule. By re-opening the path at $\beta = 0$, the cyclical schedule can progressively improve the performance, by leveraging good latent codes learned in the previous cycles as warm re-starts. We demonstrate the effectiveness of the proposed approach on three NLP tasks, and show that it is superior to or complementary to other techniques.

2.8 Appendix

2.8.1 Comparison of different schedules

We compare the two different scheduling schemes in Figure 2.7. The three widely used monotonic schedules are shown in the top row, including linear, sigmoid and cosine. We can easily turn them into their corresponding cyclical versions, shown in the bottom row.

2.8.2 Proofs on the β and MI

When scheduled with β, the training objective over the dataset can be written as:

$$\mathcal{F} = -\mathcal{F}_E + \beta \mathcal{F}_R \tag{2.12}$$

We proceed the proof by re-writing each term separately.

Bound on $\mathcal{F}_E$ Following [58], on the support of $(\boldsymbol{x}, \boldsymbol{z})$, we denote q as the encoder probability measure, and p as the decoder probability measure. Note that the reconstruction loss for $\boldsymbol{z}$ can be writen as its negative log likelihood form as:

$$\mathcal{F}_E = -\mathbb{E}_{\boldsymbol{x}\sim q(\boldsymbol{x}), \boldsymbol{z}\sim q(\boldsymbol{z}|\boldsymbol{x})}[\log p(\boldsymbol{x}|\boldsymbol{z})]. \tag{2.13}$$

Lemma 1. *For random variables $\boldsymbol{x}$ and $\boldsymbol{z}$ with two different probability measures, $p(\boldsymbol{x}, \boldsymbol{z})$ and $q(\boldsymbol{x}, \boldsymbol{z})$, we have*

$$\begin{aligned}
&H_p(\boldsymbol{z}|\boldsymbol{x}) \\
&= -\mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z}), x\sim p(\boldsymbol{x}|\boldsymbol{z})}[\log p(\boldsymbol{z}|\boldsymbol{x})] \\
&= -\mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z}), \boldsymbol{x}\sim p(\boldsymbol{x}|\boldsymbol{z})}[\log q(\boldsymbol{z}|\boldsymbol{x})] \\
&- \mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z}), x\sim p(\boldsymbol{x}|\boldsymbol{z})}\big[\log p(\boldsymbol{z}|\boldsymbol{x}) - \log q(\boldsymbol{z}|\boldsymbol{x})\big] \\
&= -\mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z}), \boldsymbol{x}\sim p(\boldsymbol{x}|\boldsymbol{z})}[\log q(\boldsymbol{z}|\boldsymbol{x})] \\
&- \mathbb{E}_{p(\boldsymbol{x})}(KL(p(\boldsymbol{z}|\boldsymbol{x})\|q(\boldsymbol{z}|\boldsymbol{x}))) \\
&\leq -\mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z}), \boldsymbol{x}\sim p(\boldsymbol{x}|\boldsymbol{z})}[\log q(\boldsymbol{z}|\boldsymbol{x})] \qquad (2.14)
\end{aligned}$$

where $H_p(\boldsymbol{z}|\boldsymbol{x})$ is the conditional entropy. Similarly, we can prove that

$$H_q(\boldsymbol{x}|\boldsymbol{z}) \leq -\mathbb{E}_{\boldsymbol{x}\sim q(\boldsymbol{x}), \boldsymbol{z}\sim q(\boldsymbol{z}|\boldsymbol{x})}[\log p(\boldsymbol{x}|\boldsymbol{z})] \qquad (2.15)$$

From lemma 1, we have

Corollary 1. *For random variables $\boldsymbol{x}$ and $\boldsymbol{z}$ with probability measure $p(\boldsymbol{x}, \boldsymbol{z})$, the mutual information between $\boldsymbol{x}$ and $\boldsymbol{z}$ can be written as*

$$\begin{aligned}
I_q(\boldsymbol{x}, \boldsymbol{z}) = H_q(\boldsymbol{x}) - H_q(\boldsymbol{x}|\boldsymbol{z}) &\geq H_q(\boldsymbol{x}) \\
&+ \mathbb{E}_{\boldsymbol{x}\sim q(\boldsymbol{x}), \boldsymbol{z}\sim q(\boldsymbol{z}|\boldsymbol{x})}[\log p(\boldsymbol{x}|\boldsymbol{z})] \\
&= H_q(\boldsymbol{x}) + \mathcal{F}_E \qquad (2.16)
\end{aligned}$$

Decomposition of $\mathcal{F}_R$ The KL term in (3.5) can be decomposed into two refined terms [57]:

$$
\begin{aligned}
&\mathcal{F}_R \\
&= \mathbb{E}_{q(\boldsymbol{x})}[\mathrm{KL}(q(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z}))] \\
&= \mathbb{E}_{q(\boldsymbol{z},\boldsymbol{x})}[(\log q(\boldsymbol{z}|\boldsymbol{x}) - \log p(\boldsymbol{z}))] \\
&= \mathbb{E}_{q(\boldsymbol{z},\boldsymbol{x})}[(\log q(\boldsymbol{z}|\boldsymbol{x}) - \log q(\boldsymbol{z})] \\
&\qquad + \mathbb{E}_{q(\boldsymbol{z},\boldsymbol{x})}[\log q(\boldsymbol{z}) - \log p(\boldsymbol{z}))] \\
&= \mathbb{E}_{q(\boldsymbol{z},\boldsymbol{x})}[(\log q(\boldsymbol{z},\boldsymbol{x}) - \log q(\boldsymbol{x}) - \log q(\boldsymbol{z})] \\
&\qquad + \mathbb{E}_{q(\boldsymbol{z},\boldsymbol{x})}[\log q(\boldsymbol{z}) - \log p(\boldsymbol{z}))] \\
&= \underbrace{I_q(\boldsymbol{z},\boldsymbol{x})}_{\mathcal{F}_1\text{: Mutual Info.}} + \underbrace{\mathrm{KL}(q(\boldsymbol{z})||p(\boldsymbol{z}))}_{\mathcal{F}_2\text{: Marginal KL}}
\end{aligned} \tag{2.17}
$$

2.8.3 Model Description

Conditional VAE for dialog Each conversation can be represented via three random variables: the dialog context $\boldsymbol{c}$ composed of the dialog history, the response utterance $\boldsymbol{x}$, and a latent variable $\boldsymbol{z}$, which is used to capture the latent distribution over the valid responses($\beta = 1$) [41]. The ELBO can be written as:

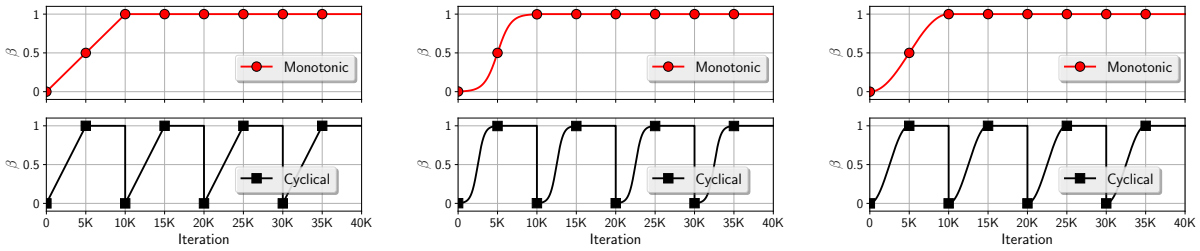
$$
\begin{aligned}
\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{c}) &\geq \mathcal{L}_{\mathrm{ELBO}} \\
&= \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{c})}\big[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z},\boldsymbol{c})\big] \\
&\quad - \beta \mathrm{KL}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{c})||p(\boldsymbol{z}|\boldsymbol{c}))
\end{aligned} \tag{2.18}
$$

Semi-supervised learning with VAE We use a simple factorization to derive the ELBO for semi-supervised learning. α is introduced to regularize the strength of classification loss.

$$
\log p_{\boldsymbol{\theta}}(\boldsymbol{y},\boldsymbol{x}) \geq \mathcal{L}_{\mathrm{ELBO}} \tag{2.19}
$$

$$
\begin{aligned}
&= \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\big[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) + \alpha \log p_{\boldsymbol{\psi}}(\boldsymbol{y}|\boldsymbol{z})\big] \\
&- \beta \mathrm{KL}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z}))
\end{aligned} \tag{2.20}
$$

where $\boldsymbol{\psi}$ is the parameters for the classifier.



(a) Linear (b) Sigmoid (c) Cosine

Figure 2.7: Comparison between traditional monotonic and proposed cyclical annealing schedules.

Good latent codes $\boldsymbol{z}$ are crucial for the the classification performance, especially when simple classifiers are employed, or less labelled data is available.

2.8.4 More Experimental Results

CVAE for Dialog Response Generation

Code & Dataset We implemented different schedules based on the code[5] published by **(author?)** [41]. In the SW dataset, there are 70 available topics. We randomly split the data into 2316/60/62 dialogs for train/validate/test.

Results The results on full BLEU scores are shown in Table 2.5. The cyclical schedule outperforms the monotonic schedule in both settings. The learning curves are shown in Figure 2.8. Under similar ELBO results, the cyclical schedule provide lower reconstruction errors, higher KL values, and higher BLEU values than the monotonic schedule. Interestingly, the monotonic schedule tends to overfit, while the cyclical schedule does not, particularly on reconstruction errors. It means the monotonic schedule can learn better latent codes for VAEs, thus preventing overfitting.

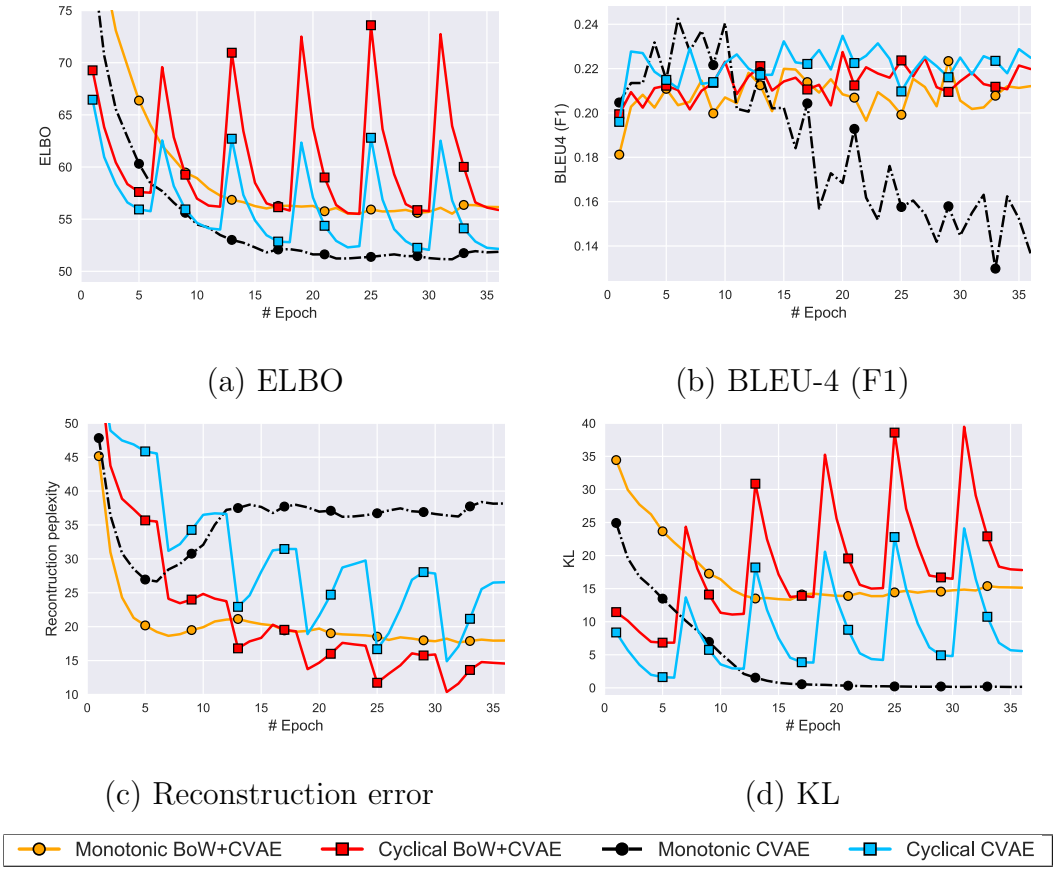
Semi-supervised Text Classification

[5]https://github.com/snakeztc/NeuralDialog-CVAE

Dataset Yelp restaurant reviews dataset utilizes user ratings associated with each review. Reviews with rating above three are considered positive, and those below three are considered negative. Hence, this is a binary classification problem. The pre-processing in [76] allows sentiment analysis on sentence level. It further filters the sentences by eliminating those that exceed 15 words. The resulting dataset has 250K negative sentences, and 350K positive ones. The vocabulary size is 10K after replacing words occurring less than 5 times with the "<unk>" token.

Results The tSNE embeddings are visualized in Figure 2.9. We see that cyclical β provides much more separated latent structures than the other two methods.

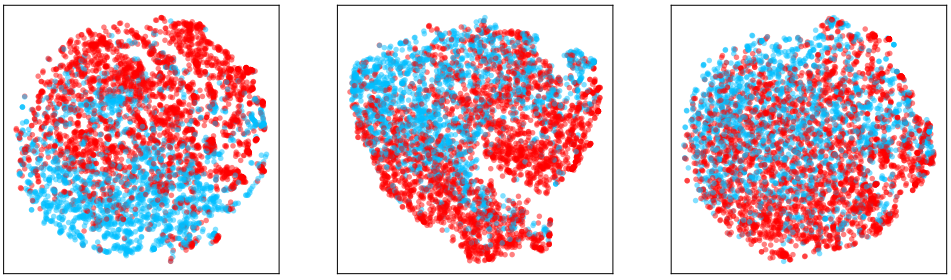
Hyper-parameter tuning The cyclical schedule has two hyper-parameters M and R. We provide the full results on M and R in Figure 2.11 and Figure 2.12, respectively. A larger number of cycles M can provide higher performance for various proportion value R.



(a) ELBO (b) BLEU-4 (F1)

(c) Reconstruction error (d) KL

Figure 2.8: Results of CVAE and BoW+CVAE on SW. Under similar ELBO, cyclical schedule provides lower rec errors, higher KL, and higher BLEU.

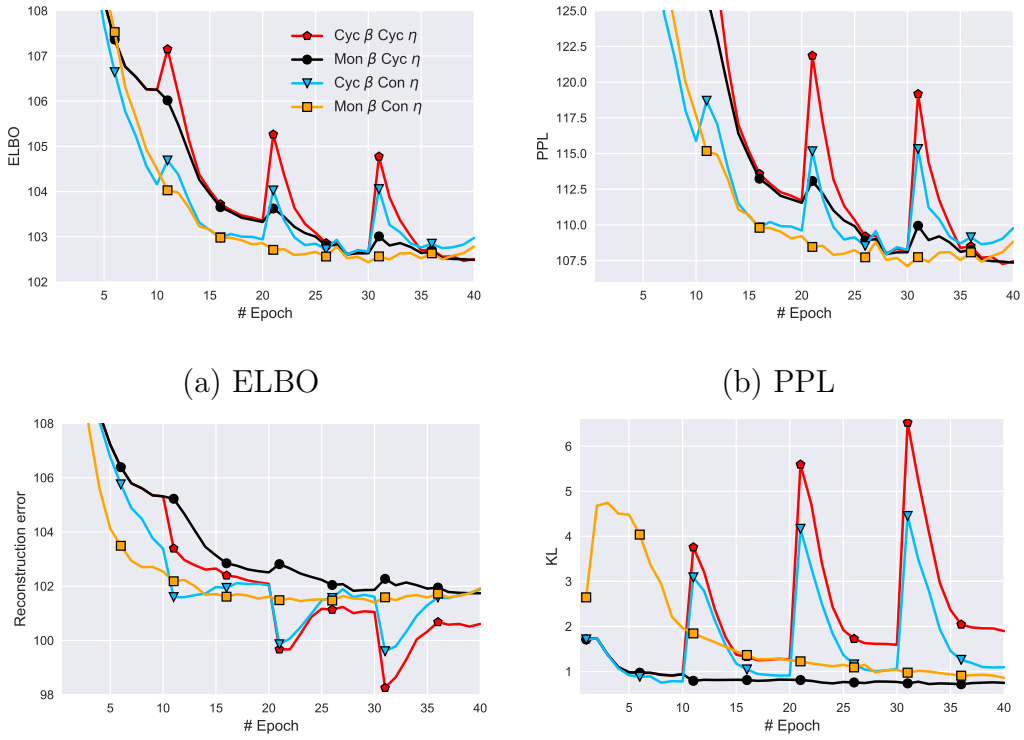


(a) Cyclical β VAE (b) Monotonic β VAE (c) AE

Figure 2.9: Comparison of tSNE embeddings on Yelp dataset. More structured latent patterns usually lead to better classification performance.

Table 2.5: Comparison on dialog response generation. Monotonic (**M**) and Cyclical (**C**) schedules are tested on two models.

Model	CVAE		CVAE+BoW	
Schedule	**M**	**C**	**M**	**C**
B1 prec	0.326	**0.423**	0.384	**0.397**
B1 recall	0.214	**0.391**	0.376	**0.387**
B2 prec	0.278	**0.354**	0.320	**0.331**
B2 recall	0.180	**0.327**	0.312	**0.323**
B3 prec	0.237	**0.299**	0.269	**0.279**
B3 recall	0.153	**0.278**	0.265	**0.275**
B4 prec	0.185	**0.234**	0.211	**0.219**
B4 recall	0.122	**0.220**	0.210	**0.219**



(a) ELBO

(b) PPL

(c) Reconstruction error

(d) KL

Figure 2.10: Ablation study on cyclical schedules on β and η. It is the cyclical β rather than cyclical η that contributes to the improved performance

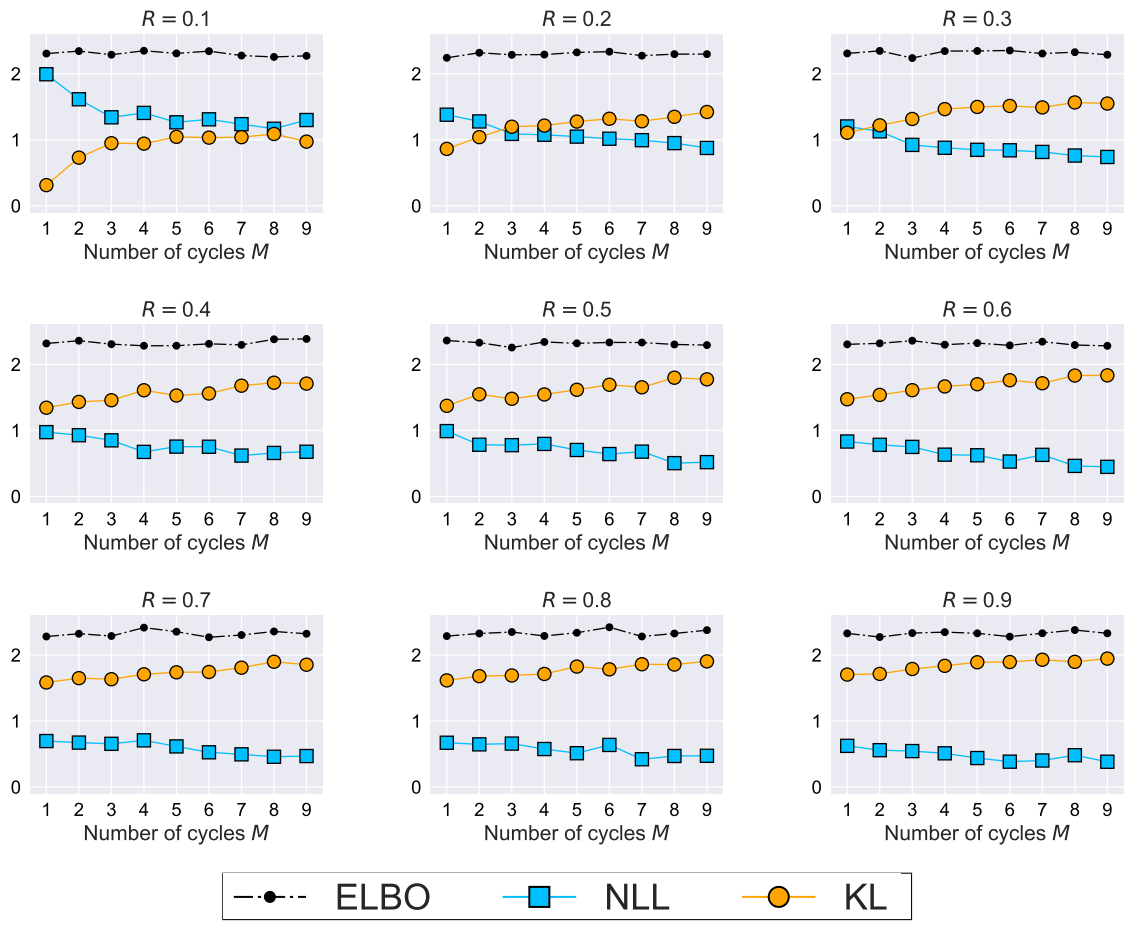


Figure 2.11: The impact of hyper-parameter M: number of cycles. A larger number of cycles lead to better performance.

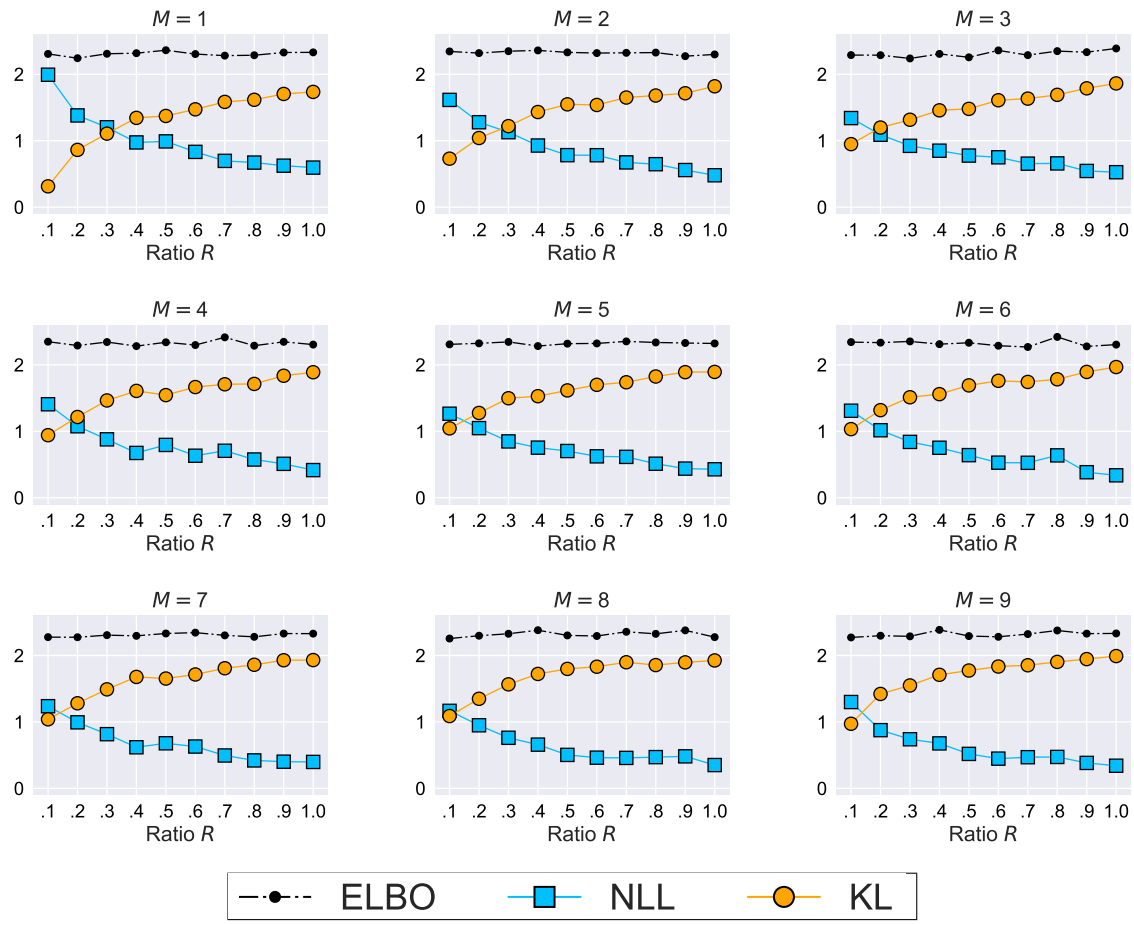


Figure 2.12: The impact of hyper-parameter R: proportion for the annealing stage. A larger R leads to better performance for various M.

Chapter 3

Text VAE with Flexible Latent Distribution

3.1 Introduction

Deep latent variable models such as variational autoencoders (VAEs) [36, 39] are becoming increasingly popular in natural language processing (NLP). They have contributed to fundamental advances in many NLP tasks, such as language modeling [37, 40], dialog response generation [41, 42]. Compared with the traditional neural language models [49], VAE is designed to explicitly capture the global features of natural languages in a continuous latent variable. It provides an effective mechanism for reasoning about the observed languages directly in the abstract level, enabling one step further towards controllable text generation [44].

While VAE models provide a rich, flexible framework for modeling many NLP tasks, difficulties exist: deep parameterizations of conditional likelihoods usually make posterior inference intractable, and latent variable objectives often complicate backpropagation by introducing points of non-differentiability [79]. To perform approximate posterior inference, the Gaussian form is usually assumed for the approximate distribution to facilitate efficient learning in variational inference [36, 39].

This Gaussian approximate is a simple and effective approach for image analysis, where multi-layer perceptron (MLP) and convolutional neural networks (CNN) [50] are often used to parameterize the decoder, and the entire image sample is generated from the latent space in one shot. However, due to the sequential nature of text, an auto-regressive decoder is typically employed in the text VAE. This decoder often parameterized with a recurrent neural network (RNN) such as the long short-term memory (LSTM) [17]. The word token of a sentence is generated in a sequential manner, where the ground-truth previous word

tokens are fed into the decoder in conjunction with the latent feature. This introduces one notorious issue when a VAE is trained using Gaussian approximates: the learned Gaussian-shaped representations for all observations collapse into a single uninformative prior Gaussian distribution, yielding what is termed the *KL vanishing* problem [37].

Various attempts have been made to ameliorate this issue from different perspectives, including weakening the dependency on the ground-truth previous word tokens [37, 46], augmenting the dependency on latent codes [47], reducing the amortized inference errors [48], modifying the training objective [41], and rescheduling the training dynamics [37, 61, 80]. Among them, it is recently concluded in [61, 80, 48] that feeding high quality latent codes into the decoder at its early training stage can mitigate the KL vanishing issue. While their proposed techniques are effective, the best latent codes they can feed to the decoder still inherit the Gaussian form, yielding a gap from the target distributions. This implies that the model may still suffer from KL vanishing, leading to sub-optimal representations.

In this work, we investigate the more flexible variational approximate in learning text representations, and make the following contributions:

- We develop a semi-implicit (SI) approximate for the posterior inference. It can be viewed as Gaussian mixtures, but can easily represent an infinite number of Gaussian components without adding extra inference network weights.

- The SI representation enjoys an attractive property to enable the trade-off between expressive power and computational cost, controlled by an hyper-parameter K. This allows us to derive an efficient inference procedure to mitigate the KL issue with SI, while maintaining low cost.

- We demonstrate that the proposed SI representation improves performance on two NLP tasks, including language modeling and dialog response generation.

3.2 Background

3.2.1 VAE Model Description

For a text sequence of length T, $\boldsymbol{x} = [x_1, \cdots, x_T]$, neural language models [49] generate the t-th token x_t conditioned on the previously generated tokens:

$$p(\boldsymbol{x}) = \prod_{t=1}^{T} p(x_t|x_{<t}),$$

where $x_{<t}$ indicates all tokens before t.

However, language models lack an efficient inference mechanism, which prevents them from reasoning about data at an abstract level. For instance, language models don't allow the sort of neural sentence manipulations showcased in [37]. The VAE was introduced to fill the gap, simultaneously learning generative models with higher-quality sentence samples while learning an efficient inference network [36, 39, 37]. The *generative model* (*decoder*) draws a continuous latent vector $\boldsymbol{z}$ from prior $p(\boldsymbol{z})$, and generates the text sequence $\boldsymbol{x}$ from a conditional distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$; $p(\boldsymbol{z})$ is typically assumed a multivariate Gaussian, and $\boldsymbol{\theta}$ represents the network parameters. The following auto-regressive decoding process is usually used:

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) = \prod_{t=1}^{T} p_{\boldsymbol{\theta}}(x_t|x_{<t}, \boldsymbol{z}). \tag{3.1}$$

Parameter learning of $\boldsymbol{\theta}$ typically employs maximizing the marginal log likelihood $\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \log \int p(\boldsymbol{z}) p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) \mathrm{d}\boldsymbol{z}$. However, this marginal term is intractable to compute for many decoder choices. Thus, variational inference is considered, and the true posterior $p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}) \propto p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})$ is approximated via the variational distribution $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$. This distribution is often produced with an *inference model* or *encoder*, implemented via a $\boldsymbol{\phi}$-parameterized neural network. It yields the *evidence lower bound* (ELBO) as an objective:

$$\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) \geq \mathcal{L} = \tag{3.2}$$
$$\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\big[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})\big] - \mathrm{KL}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z}))$$

In summary, the encoder $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ provides an efficient way to infer the latent global representations $\boldsymbol{z}$ for the observation $\boldsymbol{x}$, while the decoder $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ aims to generate the sentence samples, controlled by $\boldsymbol{z}$.

3.2.2 VAE Training and Failure Modes

The ELBO can be viewed as a regularized version of the autoencoder (AE) [50]. It is thus natural to extend the negative of $\mathcal{L}_{\text{ELBO}}$ in (3.2) by introducing a hyper-parameter β to control the strength of regularization:

$$\mathcal{L}_\beta = \mathcal{E} - \beta\mathcal{R}, \quad \text{with} \tag{3.3}$$

$$\mathcal{E} = \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\big[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})\big] \tag{3.4}$$

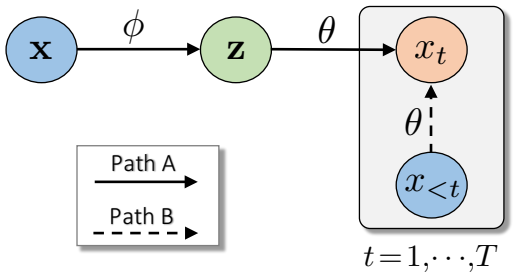
$$\mathcal{R} = \text{KL}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})) \tag{3.5}$$

where $\mathcal{E}$ is the log-likelihood or the negative reconstruction error term, and $\mathcal{R}$ is a KL regularization term.

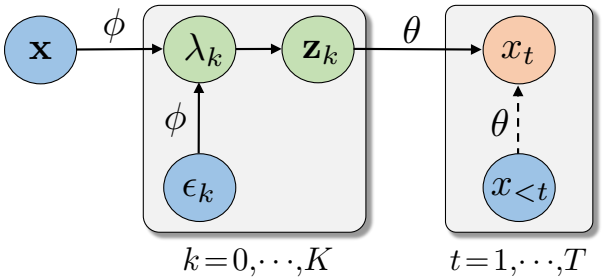
It is beneficial to understand VAEs from the cost function $\mathcal{L}_\beta$. $\beta = 1$ yields the standard VAE in (3.2), while $\beta = 0$ yields the AE objective (when $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ is defined as a delta distribution). One may need to carefully train VAEs with $\mathcal{L}_\beta$ to avoid the following two failing modes.

Over-fitting Issues When the KL regularization term $\mathcal{R}$ is weak (as in the AE), only a point-estimate represents the text sequence's latent feature, as the learned representations are not effectively regularized toward a prior distribution. In practice, it has been found that learning with an AE is prone to over-fitting [37], or generating plain dialog responses [41]. Hence, it is desirable to retain meaningful posteriors in real NLP applications.

KL Vanishing Issues The VAE objective in (3.2) suggests keeping $\beta = 1$ *constant* during the entire training procedure. Unfortunately, instability on language modeling has been witnessed, in that the KL term $\mathcal{R}$ learns faster than the NLL term $\mathcal{E}$, and $\mathcal{R}$ quickly



(a) Vanilla VAE with an auto-regressive decoder



(b) SI-VAE with an auto-regressive decoder

Figure 3.1: Illustration of learning bottleneck in two methods: (a) Vanilla VAE and (b) Semi-implicit VAE.

becomes vanishingly small during training [37]. This is known as the *KL vanishing* issue, which causes undesirable outcomes: the encoder produces posteriors almost identical to the Gaussian prior, for all observations (rather than a more informative posterior); the decoder ignores the $\boldsymbol{z}$, and the VAE model reduces to a simpler language model. To address the issue, *monotonic annealing schedule* was proposed to increase β from 0 to 1 during the early stage of VAE learning [37].

3.3 Semi-Implicit Text Representations

3.3.1 Diagnostics of KL Vanishing

It has been recently shown that the auto-regressive decoder in VAE has two paths working together to generate text sequences, the KL vanishing issue happens due to the destruc-

tive competition between the two paths for the information flows [80], illustrated in Figure 3.1(a). The circles with blue, green and red indicate observed, latent, reconstructed variables, respectively. Path A consists of $\{\boldsymbol{\phi}, \boldsymbol{\theta}\}$ passing through $\boldsymbol{z}$. It firstly extracts the global representation $\boldsymbol{z}$, then generates $\boldsymbol{x}$ directly. Path B employs the partial ground-truth information of $\boldsymbol{x}$ at every time step of the sequential decoding, it generates x_t conditioned on $x_{<t}$. Hence, both paths compete to generate the observed text; the KL vanishing issue would appear when Path B dominates Path A.

However, it is unknown why Path A tends to be weak in this competition. In this work, we hypothesize that there is a *bottleneck* on Path A that may weaken the information flow from $\boldsymbol{x}$ to x_t. Let's examine the bottleneck by reviewing the the assumption made in variational approximate. Typically, $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ is modeled as a fully factorized Gaussian distribution for its simplicity and efficiency:

$$q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{\mu}, \text{diag}(\boldsymbol{\sigma}^2)), \text{ with } \boldsymbol{\mu}, \boldsymbol{\sigma}^2 = f_{\boldsymbol{\phi}}(\boldsymbol{x}) \tag{3.6}$$

where $f_{\boldsymbol{\phi}}$ is a $\boldsymbol{\phi}$-parameterized inference network, which outputs the mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\sigma}^2$ of the Gaussian distribution.

The learning of VAE is highly scalable to large datasets because the inference network shares the parameters to amortize inferential cost. This amortized approach contrasts with traditional variational techniques that have separate local variational parameters for every data point. The inference network aims to find the variational parameters for all sentences in the space of Gaussian distribution. The performance is determined by two factors [81]: (*i*) the capacity of the Gaussian distribution to match the true posterior; and (*ii*) the ability of the inference network to produce good variational parameters for each sentence.

Specifically, the inference gap of a VAE, as described in [48], is defined as the KL divergence between inference network posterior $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ and true posterior distribution $p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$. The inference gap can be decomposed into two distinct terms (the approximation

gap and the amortization gap) as follows:

$$\mathrm{KL}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})||p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})) = \underbrace{\mathrm{KL}(q_{\boldsymbol{\lambda}^*}(\boldsymbol{z})||p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}))}_{\text{Approximation Gap}} \tag{3.7}$$

$$+ \underbrace{\mathrm{KL}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})||p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})) - \mathrm{KL}(q_{\boldsymbol{\lambda}^*}(\boldsymbol{z})||p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}))}_{\text{Amortization Gap}} \tag{3.8}$$

Here $\boldsymbol{\lambda} = [\boldsymbol{\mu}, \boldsymbol{\sigma}^2]$ represents the mean and variance for Gaussian distribution. The approximation gap is the difference between the true posterior and the best possible variational posterior within the family of distributions $\mathcal{Q}$. This gap arises due to the limited expressiveness of the variational family, such as the multivariate Gaussian used in vanilla VAE. The amortization gap, on the other hand, is the difference between the inference network posterior and the best possible posterior. This gap arises because the shared parameters from the inference network are used for each data point. Both the approximation and amortization gaps have an impact on the learned generative model.

In practice, the inference networks may always have a capacity limit, which can result in imperfect variational parameters. In certain extreme cases, the limited capacity may produce all variational Gaussian parameters that share a similar configuration such as the prior, *i.e.,* KL vanishing. Alternatively, we may relax the Gaussian approximation to more expressive forms. Recent studies show that the KL vanishing can be alleviated if the high-quality latent codes are provided at the beginning of decoder training [48, 61, 80]. It is thus promising to reduce the issue by feeding $\boldsymbol{z}$ samples that are beyond Gaussian, and close to the target distributions.

3.3.2 From GMM to Semi-Imipclit Posteriors

One natural idea to extend the Gaussian posterior in (3.6) to more flexible distribution is the Gaussian Mixture Models (GMM) [82]:

$$q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{\mu}_k, \mathrm{diag}(\boldsymbol{\sigma}_k^2)),$$
$$\text{with } \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\sigma}_k^2\}_{k=1}^{K} = f_{\boldsymbol{\phi}}(\boldsymbol{x}), \tag{3.9}$$

where K is the total number of Gaussian components, $\boldsymbol{\mu}_k$ and $\boldsymbol{\sigma}_k^2$ is the mean and variance of the k-th component, and $\boldsymbol{\pi} = [\pi_1, \cdots, \pi_K]$ is the mixture coefficient. The traditional GMM is restrictive in that (i) K needs to be pre-defined, and (ii) the size of network $\boldsymbol{\phi}$ for amortized inference will increase as K increases.

To bypass these issues, we propose the following *semi-implicit* (SI) form of GMM [83]:

$$q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) = \int_{\boldsymbol{\lambda}} q(\boldsymbol{z}|\boldsymbol{\lambda}) q_{\boldsymbol{\phi}}(\boldsymbol{\lambda}|\boldsymbol{x}) \tag{3.10}$$

where $\boldsymbol{\lambda} = [\boldsymbol{\mu}, \boldsymbol{\sigma}^2]$ is the mean and variance for Gaussian distribution $q(\boldsymbol{z}|\boldsymbol{\lambda}) = \mathcal{N}(\boldsymbol{z}; \boldsymbol{\mu}, \boldsymbol{\sigma}^2)$, whose distribution parameters are drawn from an *implicit* distribution $\boldsymbol{\lambda} \sim q_{\boldsymbol{\phi}}(\boldsymbol{\lambda}|\boldsymbol{x})$. Specifically, the sampling process is:

$$\boldsymbol{\lambda} = f_{\boldsymbol{\phi}}(\boldsymbol{x}, \boldsymbol{\epsilon}), \text{ with } \boldsymbol{\epsilon} \sim q_0(\boldsymbol{\epsilon}) \tag{3.11}$$

where $q_0(\boldsymbol{\epsilon})$ can be any easy-to-sample distribution, $f_{\boldsymbol{\phi}}$ is a $\boldsymbol{\phi}$-parameterized network with input $\{\boldsymbol{x}, \boldsymbol{\epsilon}\}$.

Note that (3.10) can be viewed as GMM with an infinite number of components. As once $\boldsymbol{\phi}$ is well trained, one can draw an arbitrary number of samples via (3.11), each of which corresponds to a new Gaussian component. In contrast to (3.9), this sampling process only requires adding a small constant number of network parameters in $\boldsymbol{\phi}$.

We show the probabilistic graphical model of SI in Figure 3.1 (b). Path A is now composed of $\{\boldsymbol{x}, \boldsymbol{\epsilon}\} \to \boldsymbol{\lambda} \to \boldsymbol{z} \to x_t$, where $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ has a much more flexible distribution family to choose from than a Gaussian form. It allows $\boldsymbol{z}$ to encode $\boldsymbol{x}$ more easily. The enriched $\boldsymbol{z}$ can strength path A, and keep it stay informative in the two-path competition.

3.3.3 Learning and Inference with SI

Though flexible, the SI representation in (3.10) introduces additional difficulty in learning: the evaluation of the KL term in (3.5) becomes inefficient because its closed form disappears.

Fortunately, there is a lower bound for the KL term:

$$\begin{aligned}\mathcal{R} &= -\operatorname{KL}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})) && (3.12)\\ &= -\operatorname{KL}(\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{\lambda}|\boldsymbol{x})} q(\boldsymbol{z}|\boldsymbol{\lambda})||p(\boldsymbol{z}))\\ &\geq -\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{\lambda}|\boldsymbol{x})}\big[\operatorname{KL}(q(\boldsymbol{z}|\boldsymbol{\lambda})||p(\boldsymbol{z}))\big] \triangleq \underline{\mathcal{R}}. && (3.13)\end{aligned}$$

Note that (3.13) is efficient to evaluate due to the Gaussian form of $q(\boldsymbol{z}|\boldsymbol{\lambda})$. Therefore, we can maximize the lower bound of ELBO:

$$\mathcal{L} = \mathcal{E} + \mathcal{R} \geq \mathcal{E} + \underline{\mathcal{R}} \triangleq \underline{\mathcal{L}} \tag{3.14}$$

However, directly optimizing (3.14) causes a *degeneracy* issue that $q_{\boldsymbol{\phi}}(\boldsymbol{\lambda}|\boldsymbol{x})$ could converge to a point mass density, making $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ reduce to the vanilla Gaussian form. To prevent degeneracy, a repulsive term is added to regularize $\underline{\mathcal{L}}$ [83]:

$$\begin{aligned}\mathcal{B}_K =& \mathbb{E}_{\boldsymbol{\lambda},\boldsymbol{\lambda}^{(1)},\ldots,\boldsymbol{\lambda}^{(K)}\sim q_{\boldsymbol{\phi}}(\boldsymbol{\lambda}|\boldsymbol{x})} \operatorname{KL}(q(\boldsymbol{z}|\boldsymbol{\lambda})||\tilde{q}_K(\boldsymbol{z}|\boldsymbol{\lambda})),\\ \text{where}\quad \tilde{q}_K(\boldsymbol{z}|\boldsymbol{\lambda}) &\triangleq \frac{q(\boldsymbol{z}|\boldsymbol{\lambda}) + \sum_{k=1}^{K} q(\boldsymbol{z}|\boldsymbol{\lambda}^{(k)})}{K+1}. \end{aligned} \tag{3.15}$$

It leads to $\underline{\mathcal{L}}_K \triangleq \underline{\mathcal{L}} + \mathcal{B}_K$. Note that maximizing $\underline{\mathcal{L}}_K$ with $K \geq 1$ would encourage positive $\mathcal{B}_K$ and drive $q_{\boldsymbol{\phi}}(\boldsymbol{\lambda}|\boldsymbol{x})$ away from degeneracy. In another word, $\mathcal{B}_K$ guarantees the SI in (3.10) to spread out as an infinite number of Gaussian mixtures, rather than collapsing into one single Gaussian.

3.4 Mitigating KL Vanishing with SI

The KL vanishing issue happens due to a lack of good latent codes $\boldsymbol{z}$ in the initial stages of decoder training [61, 80], and the decoder passively chooses to use the previous word tokens $x_{<t}$ alone to generate the next word x_t. Based on this observation, two algorithms are proposed independently:

- **Aggressive encoder training**: He et al. [61] propose to aggressively optimize the encoder T_a times ($T_a > 0$) before performing each model update.

- **Cyclical β-annealing schedule**: Fu et al. [80] propose simply repeats the monotonic β annealing procedure M times ($M > 1$). Within each cycle, only a proportion $R \in (0, 1)$ is used to increase β from 0 to 1, and while the $1 - R$ for fixing $\beta = 1$. A linear annealing scheme is:

$$\beta_t = \begin{cases} \tau/R, & \tau \leq R \\ 1, & \tau > R \end{cases} \quad \text{with} \quad \tau = \frac{\text{mod}(t-1, \lceil T/M \rceil)}{T/M} \ . \tag{3.16}$$

Both work share a similar motivation: improving the quality of $\boldsymbol{z}$ before it is used for decoder training. However, they optimize the latent codes in the space of Gaussian distribution, due to the variational assumption.

In this work, we propose to further mitigate the KL issue through improving the latent codes using SI. The update of SI is based on the tractable bound:

$$\begin{aligned} \underline{\mathcal{L}}_K =& \underline{\mathcal{L}} + \mathcal{B}_K = \mathcal{E} + \underline{\mathcal{R}} + \mathcal{B}_K & (3.17) \\ =& \mathbb{E}_{\boldsymbol{z} \sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) + p(\boldsymbol{z}) - \tilde{q}_K(\boldsymbol{z}|\boldsymbol{\lambda})]. \\ \approx& \frac{1}{J} \sum_{j=1}^{J} \Big[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}_j) + \log p(\boldsymbol{z}_j) & (3.18) \\ & - \log \frac{1}{K+1} [\sum_{k=1}^{K} q(\boldsymbol{z}_j|\boldsymbol{\lambda}_k) + q(\boldsymbol{z}_j|\boldsymbol{\lambda}_j)] \Big], \end{aligned}$$

where the expectation in (3.17) is estimated via J samples in (3.18), with the j-th sample drawn as:

$$\boldsymbol{z}_j \sim q(\boldsymbol{z}|\boldsymbol{\lambda}_j), \quad \boldsymbol{\lambda}_j \sim q_{\boldsymbol{\phi}}(\boldsymbol{\lambda}|\boldsymbol{x}) \tag{3.19}$$

On the impact of K The representational power of SI relies on K, the number of samples used to approximate (3.10). Each sample is used to construct the variational parameters for one Gaussian mixture, thus larger K indicates more Gaussian mixtures. It has been shown in [83] that the regularized lower bound $\underline{\mathcal{L}}_K$ is an asymptotically exact ELBO that satisfies $\underline{\mathcal{L}}_0 = \underline{\mathcal{L}}$ and $\lim_{K \to \infty} \underline{\mathcal{L}}_K = \mathcal{L}$.

Table 3.1: Comparisons of vanilla VAE and SI. The SI representation provides a trade-off between expressive power and computational cost.

	VAE	SI
K	$K = 0$	$K \geq 1$
Speed	Fast	Larger K yields higher compute cost
Flexibility	Limited	Larger K yields higher flexibility

We summarize the connection/comparison of vanilla VAE and SI in Table 3.1. Note that vanilla VAE can be considered as a special case of SI-VAE when $K = 0$. SI can balance the trade-off between the posterior flexibility and the computational cost using K. To leverage both advantages, we suggest an *annealing training* procedure via scheduling K. It performs in two stages: (*i*) A small K is first used. We can leverage the speed advantage to push the variational distribution towards the target distribution at a fast pace. (*ii*) We gradually increase K to a pre-defined $K_{\max}$. It refines the distribution learned in the first stage into highly flexible SI representations.

This annealing training procedure can be integrated into the aggressive encoder training [61] or the cyclical schedule [80] to get better performance, while maintaining low computational cost. Note the key to reduce KL vanishing is to feed high-quality $\boldsymbol{z}$ to decode in the initial stage of decode learning. Here, this initial stage means the time after each aggressive training loop ends in [61], or each cyclical period ends in [80]. Hence, the SI annealing training periods can be synchronized with them. By the time the next aggressive loop or cyclical period start, we can make sure the flexible SI latent codes are fed into the decoder.

We summarize the full training in Algorithm 1. K_t is iteration-dependent; it is reserved to implement the stacked training. Note that when $M = 1$ and $R < 0.5$, we have the standard monotonical annealing scheme. When $T_a = 0$ we have the standard encoder-decoder update.

Algorithm 1: SI Training

Input: encoder $\boldsymbol{\phi}$, decoder $\boldsymbol{\theta}$, K_t, T_a, M, R

Initialize: $\boldsymbol{\phi}$, $\boldsymbol{\theta}$;

for $t = 1, 2, \ldots, T$ **do**

  % β annealing schedule

  Evaluate β_t using (3.16) ;

  % Aggressive encoder training

  for $t_a = 1, 2, \ldots, T_a$ **do**

    Sample a minibatch $\mathbf{X}$;

    Compute gradient $g_{\boldsymbol{\phi}} = \nabla_{\boldsymbol{\phi}} \underline{\mathcal{L}}_{\beta_t, K_t}(\mathbf{X})$;

    Update $\boldsymbol{\phi}$ using gradient $g_{\boldsymbol{\phi}}$;

  end

  % SI update

  Sample a minibatch $\mathbf{X}$;

  Sample $\boldsymbol{\lambda}_k$ using (3.11) for $k = 1, \cdots, K_t$;

  Sample $\boldsymbol{z}_j$ using (3.19) for $j = 1, \cdots, J$;

  Compute gradient $g_{\boldsymbol{\phi},\boldsymbol{\theta}} = \nabla_{\boldsymbol{\phi},\theta} \underline{\mathcal{L}}_{\beta_t, K_t}(\mathbf{X})$;

  Update $\boldsymbol{\theta}$, $\boldsymbol{\phi}$ using gradient $g_{\boldsymbol{\phi},\boldsymbol{\theta}}$;

end

3.5 Related Work

Good latent representations are key for a wide range of NLP tasks. We recommend [79] for a comprehensive study for deep latent variable models in natural languages. Several VAE model variants have been developed to reduce the approximate errors in Gaussian proposals via more flexible posteriors, including normalizing flows (NFs) [84], adversarial variational Bayes (AVB) [85], particle-based methods [86]. Our method is based on on [83]. It bypasses the strict invertibility requirement of NFs, the instability of adversarial training in AVB,

and prohibitive computation in particle-based methods. Nevertheless, all these methods can provide more accurate posterior approximate than the Gaussian variants when carefully tuned. However, to the best of our knowledge, they are only evaluated on image datasets with MLP/CNN decoders. In the text domain, the use of auto-regressive decoders in VAE brings additional difficulties in learning representations. Our work presents the first work to investigate the flexible posterior learning for auto-regressive decoders. We gain the insights that the KL vanishing issue is less severe than previously thought when trained with more flexible posteriors such as SI.

3.6 Experimental Results

We implement SI in Pytorch. The source code to reproduce the experimental results will be made publicly available on GitHub[1]. The **M** and **C** indicate training with monotonic and cyclical β schedule, respectively; **A** indicates the results to combine with aggressive encoder training.

3.6.1 Visualization of Latent Spaces

To illustrate the learned $\boldsymbol{z}$ of SI and VAE, we visualize the latent space of a synthetic dataset. Consider a dataset consisting of 10 sequences, each of which is a 10-dimensional one-hot vector with the value 1 appearing in different positions. A 2-dimensional latent space is used for the convenience of visualization. A 1-layer LSTM with hidden units is employed for the decoder in analog with the sentence decoding process. The encoder is implemented using a 2-layer MLP with 64 hidden. We use $T = 4$K total iterations, $K = J = 100$. Three β-scheduling schemes are considered [80].

We visualize the resulting latent space division of the last training step in Figure 3.2, where each color corresponds to $\boldsymbol{z} \sim q(\boldsymbol{z}|n)$, for $n = 1, \cdots, 10$. The Gaussian VAE results are in the top row. The constant schedule produces heavily mixed latent codes $\boldsymbol{z}$ for

[1] https://github.com/

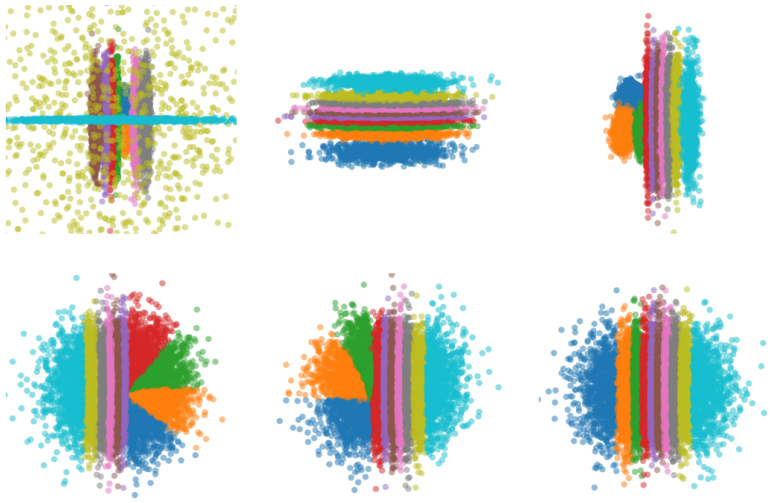
different sequences. The monotonic schedule divides the space into a mixture of 10 thin and long cluttered Gaussians. The cyclical schedule behaves similarly but with more circled distributions. It shows that Gaussian VAE results heavily depend on the β-scheduling schemes. This is perhaps because the learning process is conducted in the restrictive space of Gaussian variational forms, carefully-crafted searching schedules play an important role. However, the proposed SI method consistently produces well-divided latent representations for all three schedules. Importantly, its posteriors $q(\boldsymbol{z}|n)$ have non-Gaussian forms such as slice-shaped distributions.

We refer $q(\boldsymbol{z}) = \sum_{n=1}^{N} q(\boldsymbol{z}|n)q(n)$ as the aggregated posterior [55]. This marginal distribution characterizes the aggregated $\boldsymbol{z}$ after embedding the entire dataset into the latent space. Good latent representations should have low $\mathrm{KL}(q(\boldsymbol{z})||p(\boldsymbol{z}))$. Ideally, it means the overall shape of $\boldsymbol{z}$ samples is close to $p(\boldsymbol{z}) = \mathcal{N}(0, 1)$. The flexibility of SI allows the learned posteriors to adapt arbitrarily well to fit the prior, thus produces high-quality $q(\boldsymbol{z})$. In contrast, the limitation of Gaussian assumptions in vanilla VAE leads to some holes between $q(\boldsymbol{z}|n)$ and $p(\boldsymbol{z})$, making $q(\boldsymbol{z})$ violate the constraint of $p(\boldsymbol{z})$. The SI representations capture more clear patterns and structured information is captured in $\boldsymbol{z}$, which is beneficial in downstream applications below.

3.6.2 Language Modeling

Dataset and Setup We first consider to apply VAEs to the language modeling task on the Penn Tree Bank (PTB) dataset [71]. Following [48, 61], we set the word embedding dimension as 512. The encoder and decoder is an LSTM with 1024 hidden units, respectively. $J=5$, K is linearly increased to $K_{\max} = 50$.

Evaluation In addition to generating high-quality sentences as in the traditional neural language models that only, VAEs also aim to learn a good posterior distribution in the latent space. The language modeling performance is evaluated with ELBO, perplexity (PPL) and importance weighted perplexity (IWP) [61]. To compute IWP, Importance weighted nega-



(a) Constant (b) Monotonic (c) Cyclical

Figure 3.2: Comparison of VAE (top row) and SI (bottom row) in terms of the learned latent spaces for the three schedules.

tive log likelihood (IWNLL) is first calculated according to $-\log \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}([(p_\theta(\boldsymbol{x}|\boldsymbol{z}))p(\boldsymbol{z})/q_\phi(\boldsymbol{z}|\boldsymbol{x})]$. The exponential of the mean of the IWNLL for each word gives IWP. Note that IWP provides a tighter bound to $\log p(\boldsymbol{x})$ than PPL. Higher ELBO and lower PPL/IWP indicate the model fits the observed sentences better. More importantly, we are interested in the learned $\boldsymbol{z}$, which is evaluated using the following three metrics:

- **AU**: The total number of active units in $\boldsymbol{z}$, defined as $A_z = \text{Cov}_{\boldsymbol{x}}(\mathbb{E}_{z\sim q(z|\boldsymbol{x})}[z]) > 0.01$ [87];
- **MI**: The mutual information $I(\boldsymbol{x}, \boldsymbol{z})$;
- **KL**: The posterior-prior KL divergence, *i.e.,* $\mathcal{R}$ defined in (3.5). The KL for SI is reported as its bound $\underline{\mathcal{R}}_K + \mathcal{B}_K$.

The results are reported in Table 3.2. We implement SI based on the Github repository [2]. We first compare SI with VAE trained with monotonic and cyclical β schedule,

[2]https://github.com/jxhe/vae-lagging-encoder

Table 3.2: Results on PTB. Blue numbers indicate SI performs than VAE. **Bold** means the highest compared with all other methods.

Methods		AU	MI	KL	PPL	IWP
M	VAE	1	0.56	0.68	94.6	93.1
	SA	-	-	1.05	109.2	-
	SI	7	2.73	4.83	105.6	96.73
C	VAE	4	0.95	1.26	95.74	96.23
	SA	-	-	2.26	108.7	-
	SI	5	**4.08**	2.63	106.3	96.87
M A	VAE	14	3.33	9.02	101.9	92.26
	SI	**32**	3.73	**9.20**	100.3	**88.51**

respectively. We see that SI achieves higher AU, MI, and KL than VAE for both schedules. It means SI can provide more informative latent codes than VAE. We also compare with the semi-amortized (SA) training [48]. SI outperforms SA in terms of KL while maintaining the same PPL. It implies that learning with more flexible distribution forms can indeed lead to more informative latent codes. When training VAE and SI with the aggressive encoder training, both methods get improved. SI still achieves substantially better results than VAE in terms of AU. Interestingly, it activates all 32 latent units. We argue that the aggressive encoder training can be particularly important for the proposed SI, as it allows SI getting fully optimized to release its representational power.

3.6.3 Dialogue Response Generation

Dataset and Setup The conditional VAE (CVAE) is often used in dialogue response generation, the learned representations are crucial in generating relevant and diverse responses. We evaluate our model on Switchboard dataset [88], which contains 2,400 two-way

Table 3.3: Performance comparison on the SwitchBoard dataset. Blue numbers indicate SI performs than VAE. **Bold** means the highest.

Methods	BLEU			BOW Embedding			Intra Distinct		Inter Distinct		L
	R	P	F1	A	E	G	dist-1	dist-2	dist-1	dist-2	
HRED	0.262	0.262	0.262	0.820	0.537	0.832	0.813	0.452	0.081	0.045	12.1
CVAE	0.295	0.258	0.275	0.836	0.572	0.846	0.803	0.415	0.112	0.102	12.4
CVAE+BOW	0.298	0.272	0.284	0.828	0.555	0.840	0.819	0.493	0.107	0.099	12.5
CVAE+CO	0.299	0.269	0.283	0.839	0.557	0.855	0.863	0.581	0.111	0.110	10.3
DialogWAE	0.394	0.254	0.309	0.897	0.627	0.887	0.713	0.651	0.245	0.413	15.5
DialogWAE+GMP	0.420	0.258	0.319	0.925	0.661	0.894	0.713	0.671	0.333	0.555	15.2
SI + **M**	0.376	0.248	0.299	0.888	0.640	0.890	0.832	0.461	0.209	0.298	11.5
CVAE + **M**	0.360	0.245	0.292	0.884	0.634	0.884	0.826	0.459	0.204	0.282	11.2
SI + **C**	**0.441**	0.246	0.315	**0.937**	0.683	0.905	**0.888**	**0.782**	**0.499**	**0.77**	12.1
CVAE + **C**	0.440	0.244	0.314	0.937	0.688	0.906	0.876	0.741	0.461	0.72	12.0

telephone conversations under 70 specified topics. We use SI to replace the Gaussian posteriors in CVAE. Our model architecture and hyper-parameters are identical to [41] except that we concatenate the context and sentence embedding with 100-dim noise vector as the input of the SI inference network. $J=5$, K is linearly increased to $K_{\max} = 30$.

Evaluation We adopted the same metrics from [82] to quantify the performance, including *BLEU*, *BOW Embedding* and *Intra & Inter Distinct*. We detailed the model and evaluation description in Appendix.

Baselines We compare the performance of SI-CVAE with six recently-proposed baselines for dialogue modeling: (1) HRED: a generalized sequence-to-sequence model with hierarchical RNN encoder [73]; (2) CVAE: a conditional VAE model with KL-annealing [89] [41]; (3) CVAE-BOW: a conditional VAE model with a BOW loss [41]; (4) CVAE-CO: a collaborative conditional VAE model [90]; (5) DialogWAE: a conditional Wasserstein autoencoder model [82] and (6) DialogWAE-GMP: a conditional Wasserstein autoencoder

Table 3.4: Generated dialog responses from SI and CVAE methods.

Context mind of the jury is probably a better vehicle for sentencing than the judge himself because then there's the consensus thing rather than a single person	
Target well there is an argument in that a judge has seen a lot more cases and you know perhaps he sees one crime	
1. yeah that is true	1. yeah
SI 2. that's a very good idea i guess is not even the jury system	**CVAE** 2. that's right
3. i think the judges should be a little more like a man	3. um - hum
4. uh - huh	4. oh i can't believe that
5. right i should be able to get on the jury that's not necessarily the way	5. uh - huh yeah i think that's a good idea but i'm not sure about it
Context favorite type of music or musician or any	
Target i don't really have a favorite type i like a lot of different types of music	
1. i'm in classical music and i enjoy listening to music	1. what kind of music do you like i said i like
SI 2. so i just love the whole bunch of different kinds of music	**CVAE** to read a lot of music
3. i used to watch cartoons and watch cartoons	2. i like to watch tv and stuff like that
4. purple buddy in a lot of fun i like to	3. have you ever been to a concert or
5. do you have a favorite show on the show	4. uh - huh
	5. oh okay well i'm not sure what the name is on tv or

model with Gaussian mixture prior [82].

Table 3.3 summarizes the results for the various methods. All baseline results from [82]. We implement SI based on their Github repository [3], and report CAVE and SI results based on our runs. We compare SI with Gaussian CVAE for both monotonic and cyclical β schedules. SI improves CVAE in terms of all evaluation metrics for the monotonic schedule. When the cyclical schedule is used, SI provides higher BLEU scores than CVAE, indicating that SI is able to generate more relevant responses. It also implies that SI is able to learn more informative $\boldsymbol{z}$, and suffer less from the KL vanishing issue. When comparing with the state-of-the-art methods, SI provides the highest BLEU Recall and BOW Embedding (A). Meanwhile, SI achieves the best Intra/Inter Distinct values, this shows the evidence that SI can produce the most diverse response generation.

Table 4 shows examples of generated responses from the CVAE and SI. Given a context, five samples are drawn for each method. SI-CVAE can generate more coherent responses that cover multiple plausible aspects. In contrast, the responses generated by CVAE exhibits relatively limited variations or less informative contents.

3.7 Conclusion

We have introduced the semi-implicit approximate in the posterior learning of VAEs to infer text representations. The flexibility of SI helps reduce the representational bottleneck in the latent space and thus alleviate the KL vanishing issue. The effectiveness of SI is validated with the clear latent space division in the synthetic dataset and improved performance on two NLP tasks: providing more informative latent codes in VAE language modeling, and promoting the diversity in conditioned dialogue response generation.

[3] https://github.com/guxd/DialogWAE

3.8 Appendix

3.8.1 Conditional VAE for Dialog

Model Description Each conversation can be represented via three random variables: the dialog context $\boldsymbol{c}$ composed of the dialog history, the response utterance $\boldsymbol{x}$, and a latent variable $\boldsymbol{z}$, which is used to capture the latent distribution over the valid responses($\beta = 1$) [41]. The ELBO can be written as:

$$\begin{aligned} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{c}) &\geq \mathcal{L}_{\text{ELBO}} \\ &= \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{c})}\big[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z},\boldsymbol{c})\big] \\ &- \beta \text{KL}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{c})||p(\boldsymbol{z}|\boldsymbol{c})) \end{aligned} \tag{3.20}$$

Evaluation We adopted the same metrics from [82] to quantify the performance for dialog response generation:

- **BLEU**: BLEU measures how much a generated response contains n-gram overlaps with the reference.
- **BOW Embedding**: The cosine similarity of bag-of-words embeddings between the hypothesis and the reference. The Glove vectors [91] are used. For each test context, we report the maximum BOW embedding score among the 10 sampled responses. We use three metrics to compute the word embedding similarity: **A**verage, **E**xtrema and **G**reedy.
- **Intra & Inter Distinct**: dist-n is defined as the ratio of unique n-grams ($n = 1, 2$) over all n-grams in the generated responses. Note that intra-dist and and inter-dist are defined as the average of distinct values within each sampled response and among all sampled responses, respectively. Higher values indicate higher diversity.

Chapter 4

Neural Architecture Search for Vision OOD Detection

4.1 Introduction

The success of Convolutional Neural Network (CNN) achieve record-breaking performance under various real-world applications, such as medical image analysis [92, 93], autonomous driving [94, 95], and human pose estimation [96, 97]. Modern machine learning system requires that the models not only be accurate but also robust to distribution shift and outliers. While modern CNN architectures enjoy high performance on computer vision benchmarks thanks to continuous development and techniques from latest research, the robustness of these models in addressing data distribution shift is rarely studied. Thus, Out-of-Distribution (OOD) detection [38] becomes an essential part for safe deployment of CNN models.

Detecting OOD inputs has made significant progress in recent years. Most previous studies have focused on developing post-hoc techniques using confidence score [98, 99, 100] or methods based on deep generative models [101, 102]. Another branch of OOD detection methods makes use of auxiliary outlier samples with extra regularization during training [103, 104] to help models learn ID/OOD discrepancy. However, there is an orthogonal aspect from the intrinsic influence of structure to OOD detection which still remains unexplored. The significance of network architectures in OOD detection has emerged in several previous experiments. Liang et al. [105] find out different architectures with close classification accuracy on in-distribution data demonstrate clear OOD detection performance difference using MSP and ODIN method. According to Hendrycks et al. [106], pretrained transformers [19] are more effective in identifying OOD examples than other network structures. Koner et al. [107] proposed a new architecture OODformer that leverages the

contextualization capabilities of the transformer to improve OOD detection performance.

Recent development and evolution in CNN architectures [108, 109, 34] significantly boost the performance on image applications with chain-structured CNN architectures. For example, Neural Architecture Search (NAS) boosts ImageNet classification accuracy from 72% in MobileNetV2 to 80% in Once-for-all under mobile computation regime. Despite the significant impact of architecture on the performance, existing OOD research overlooks such advancements and instead, use conventional chain-like CNN architectures (e.g., ResNet [9]) as backbones to carry outlier detection, achieving modest AUROC gain compared to prior state-of-the-art baselines. Thus, a systematic approach that thoroughly studies the relationship between CNN architecture fabrications and the quality outlier detectors is beneficial to push the boundary of outlier detection.

Motivated by the strong advancements brought by Neural Architecture Search (NAS) on image classification, a natural idea is incorporating NAS into the search of CNN architectures dedicated for outlier detection. However, NAS has two major challenges in crafting optimal outlier detectors. First, image-based NAS approaches use hand-crafted design motifs (e.g., ResNet block [9]) as building block to craft CNN architectures, taking advantage of its strong performance on image classification. Yet, these design motifs may not prove effective on outlier detection, see Figure 4.1. This calls for a more flexible search space to cover more flexible CNN designs, i.e., dense connectivity of versatile convolution operators. Second, evaluating outlier detectors require precise techniques to distinguish the OOD performance of different NAS models. While existing OOD research [38] made efforts to enhance the quality of outlier detector, their evaluation metric suffers from high variance and thus, not feasible to be deployed in NAS process. This calls for a more stable and robust evaluation strategy to improve the ranking of different CNN models.

Given the above challenges and motivations, our solution is establishing a systematic framework to search for the dense connectivity of versatile convolution building operators for outlier detection. Thus, we focus on the following question in this paper, "*Can the search of dense connectivity in CNN architectures benefit outlier detectors?*"

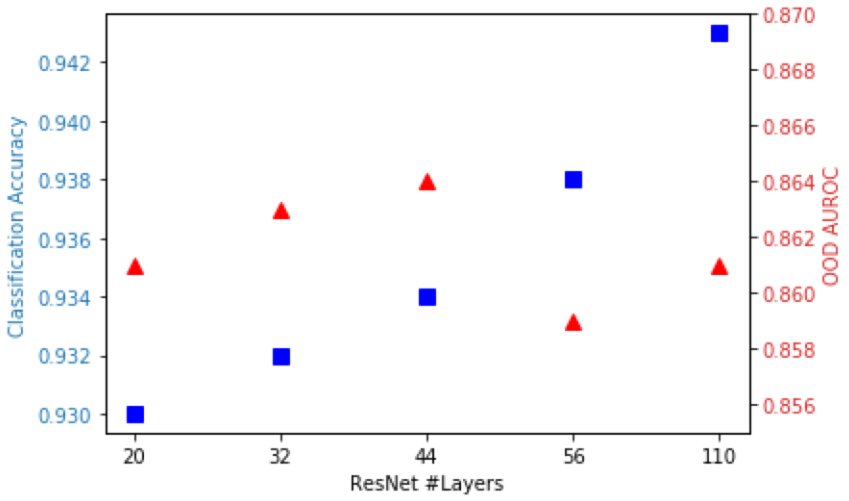


Figure 4.1: Classification Accuracy v.s. OOD AUROC of ResNets with different number of layers. Better accuracy does not indicate higher AUROC.

To fully answer the question, we propose a new paradigm, **D**ense **C**onnectivity **S**earch of **O**utlier **D**etector (DCSOD), that employ Neural Architecture Search (NAS) to automatically search for the optimal outlier detectors for out-of-distribution samples. We focus on the challenging near-OOD detection task, where the outliers exhibit semantic similarity to the in-distribution classes. To enable comprehensive exploration of dense connection patterns within neural networks, We introduce a flexible convolutional neural network (CNN) search space containing dense connectivity of versatile building operators. More specfically, we represent the hierarchical structure of CNN architectures as a composition of Directed Acyclic Graphs (DAGs). Each DAG represents a building cell in a particular hierarchy. The DAG representation captures the edge connections between heterogeneous building operators, including convolution and depthwise convolution with varying filter sizes. As such, our search space includes a wealth of potential candidates and encompasses a wide range of design motifs (e.g., ResNet [9], DenseNet [10]), allowing versatile dense connectivity between diverse computation operators with minimal structural constraints.

We obtain the optimal outlier detector from our search space by leveraging neural predictor [32]. We identify the challenge of high evaluation variance on OOD detection evalua-

tion based on multi-view feature learning: The subset of features that are acquired through cross entropy training exhibit differing levels of ability in recognizing out-of-distribution (OOD) data, resulting in a significant fluctuation in the efficacy of OOD detection. To tackle the challenge, we introduce a new outlier distillation training objective that offers more informative guidance and enables efficient multi-view learning for outlier detection, and propose an evolving distillation algorithm to mitigate the variance in OOD evaluation. The proposed algorithm searches for an evolving ensemble of teacher models, resulting in a progressively improved training loss landscape and providing better guidance on ranking candidate CNN architectures in predictor-based NAS.

We examine the proposed methods on CIFAR near-OOD benchmarks and achieve state-of-the-art (SOTA) performance over existing CNN-based structures and NAS methods. More specifically, DCSOD achieve 95.8% (95.1%) AUROC score on CIFAR-10 vs. CIFAR-100 (CIFAR-100 vs. CIFAR-10) detection task, improving previous state-of-the-art by 0.8% (1.1%). We also conduct extensive experiments to study the effects of hyper-parameters, loss landscape structures, and extract beneficial structural patterns for outlier detection. We summarize our contributions as follows:

- We propose DCSOD, a novel paradigm that leverages Neural Architecture Search (NAS) to explore optimal outlier detectors within a highly flexible search space, brewing CNN architectures with dense connectivity of versatile convolution operators.
- We provide an interpretation based on multi-view feature learning and propose an outlier distillation loss along with an evolving distillation algorithm to stabilize training and reduce variance in OOD detection evaluations.
- The best models crafted by DCSOD, achieve the state-of-the-art OOD AUROC on CIFAR near-OOD benchmarks.

4.2 Preliminaries

4.2.1 OOD Detection.

Most popular techniques for OOD detection involve designing a confidence score to distinguish outlier data. One of the most basic methods for OOD detection is to use the maximum softmax probability (MSP)[98] as the confidence score. Lee et al.[99] proposed fitting a class-conditional Gaussian distribution with feature embeddings and using the Mahalanobis distance (Maha) for OOD detection. Another line of work focus on modifying the training loss with usage of outlier examples. For instance, Outlier Exposure (OE)[103] trains the model to predict a uniform distribution over in-distribution classes. Liu et al. [100] proposed energy-bounded learning objective to explicitly shape the energy surface for differentiation between in- and out-of-distribution data.

The following provides a more detailed introduction to the methods mentioned above:

1. Maximum softmax probabilities (MSP) [98] is a simple yet strong baseline using information from output space. Assume that the neural network $\boldsymbol{f} = (f^1, ..., f^C)$ is trained to classify C classes. It takes the maximum over softmax output as confidence score for OOD detection.

$$S_{MSP}(\boldsymbol{x}) = \max_{i \in \{1,...,C\}} S^i_{soft}(\boldsymbol{x}) \tag{4.1}$$
$$\text{where } S^i_{soft}(\boldsymbol{x}) = \frac{exp(f^i(\boldsymbol{x}))}{\sum_{j=1}^{C} exp(f^j(\boldsymbol{x}))}$$

2. Mahalanobis distance (Maha) [99]: Mahalanobis is a distance-based detection method. Let N be the training size, C be the total class number, and $h(\boldsymbol{x})$ be the feature or embedding of input $\boldsymbol{x}$ obtained from certain hidden layer (usally the penultimate layer) of neural network. They fit a Gaussian distribution to the embeddings by computing per-class mean $\boldsymbol{\mu}_i$ and shared covariance matrix Σ. The score $S_{Maha}(\boldsymbol{x})$

can then be computed by taking the maximum of negative distances:

$$S_{Maha}(\boldsymbol{x}) = \max_{i\in\{1,\ldots,C\}} \{-(h(\boldsymbol{x}) - \boldsymbol{\mu}_i)^T \Sigma^{-1} (h(\boldsymbol{x}) - \boldsymbol{\mu}_i)\} \tag{4.2}$$

$$\text{where } \boldsymbol{\mu}_i = \frac{1}{N_i} \sum_{j:y_j=i} h(\boldsymbol{x}_j) \text{ and } \Sigma = \frac{1}{N} \sum_{i=1}^{C} \sum_{j:y_j=i} (h(\boldsymbol{x}_j) - \boldsymbol{\mu}_i)(h(\boldsymbol{x}_j) - \boldsymbol{\mu}_i)^T$$

3. Outlier exposure (OE) [103] is proposed to improve detection performance by utilizing the auxiliary out-distribution data $\mathcal{D}_{out}$. Specifically, they fine-tune the model by encouraging it to make uniform distribution on outlier data. The training loss is a combination of two cross-entropy losses, one for in-distribution data and another for out-of-distribution data, with λ as the controlling weight ($\mathcal{H}$ is is cross-entropy loss, $\mathcal{U}$ is uniform distribution.):

$$\mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}_{in}}[\mathcal{L}(f(\boldsymbol{x}), y)] + \lambda \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_{out}}[\mathcal{H}(\mathcal{U}; f(\boldsymbol{x}))] \tag{4.3}$$

4. Energy OOD (Energy): Instead of relying on softmax confidence score, Liu et al.[100] proposed using the energy score defined as: $E(\boldsymbol{x}) = -T \cdot \log \sum_{i=1}^{C} e^{f_i(x)/T}$ (where T is the temperature parameter, C is the total class number), to differentiate in- and out-of-distribution examples. Furthermore, they also propose an energy-bounded learning objective, where the neural network is fine-tuned to assign lower energies to the in-distribution data, and higher energies to the OOD data:

$$\mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}_{in}}[\mathcal{L}(f(\boldsymbol{x}), y)] + \lambda L_{energy} \tag{4.4}$$

$$\text{where } L_{energy} = \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}_{in}}(max(0, E(\boldsymbol{x}) - m_{in}))^2 + \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_{out}}(max(0, m_{out} - E(\boldsymbol{x})))^2 \tag{4.5}$$

4.2.2 Neural Architecture Search (NAS).

Neural Architecture Search (NAS) democratizes the development of Convolutional Neural Network (CNN) models on computer vision applications [28, 96, 110]. Reinforcement Learning [28, 29], differentiable-based search [30, 31], predictor-based [32], and one-shot

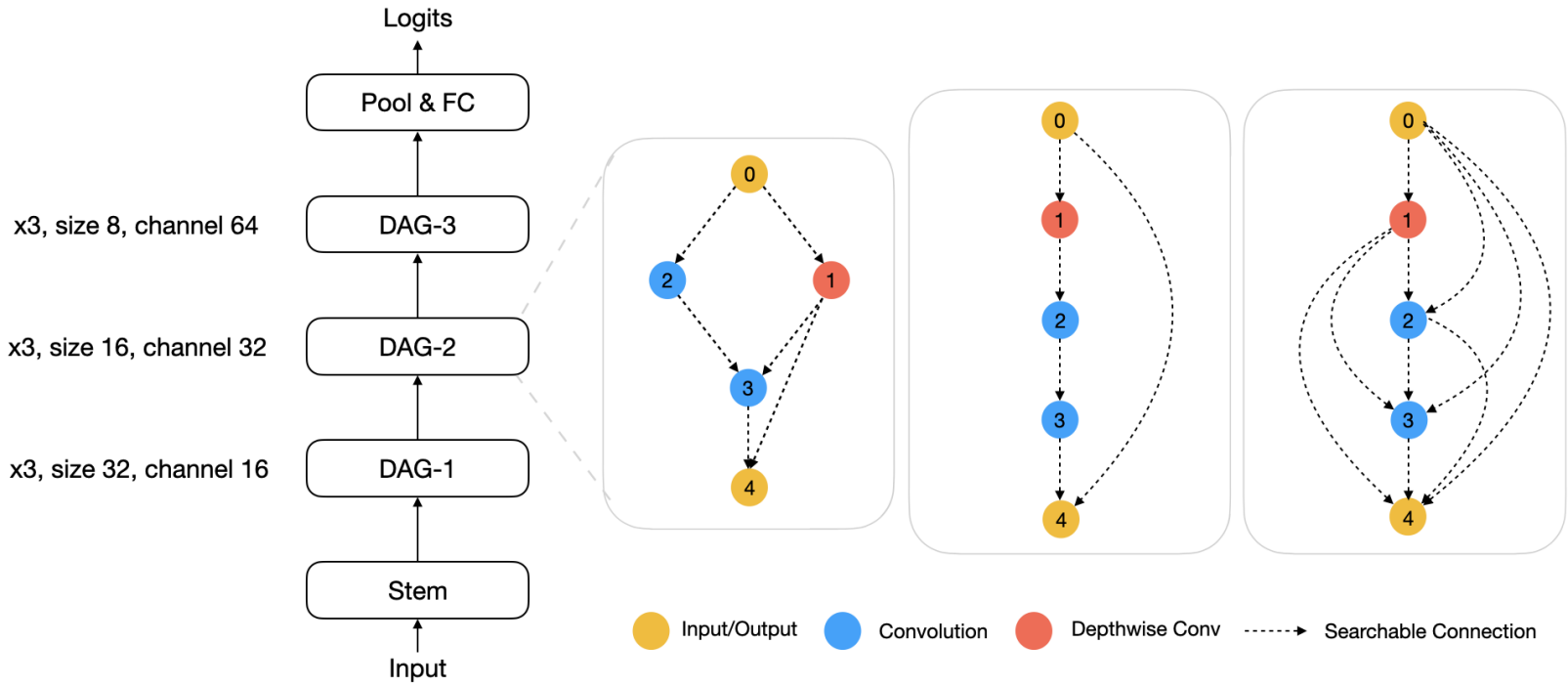


Figure 4.2: The dense connectivity search space. Standard structure such as ResNet and DenseNet are included.

search [34, 35] are popular search methods in delivering state-of-the-art vision models. However, most existing NAS methods utilizes design motifs as building blocks to brew a limited search space, with strong human priors on the quality of image perception. Yet, the same prior may not be optimal on outlier detection problems dealing with data distribution. A few NAS works attempt to explore the structural wiring of CNN architectures [111, 112, 113], yet end up with constraints (e.g., limitation of building operators between nodes) in search space and modest performance on vision benchmarks. Our approach extends the search space to enable dense connectivity between different convolutions with minimal constraints in search space, thus provides more opportunity in obtaining better outlier detectors.

4.3 Methodology

The relationship between CNN architectures and OOD evaluation quality is left unexplored in existing literature. In this work, we propose a novel paradigm, **D**ense **C**onnectivity **S**earch of **O**utlier **D**etector (DCSOD), to explore optimal outlier detectors using Neural Architecture Search. DCSOD searches for the optimal structural wiring of convolution operators within a dense connectivity search space, see Figure 4.2. The dense connectivity between different convolution operators allow information to be aggregated without structural constraints, thus extending the flexibility in CNN architecture designs compared to chain-structured counterparts (e.g., ResNet). DCSOD utilizes a neural predictor [32] to explore the design space, and introduces evolving distillation algorithm to enhance the ranking of CNN architectures. The evolving distillation algorithm enables efficient multi-view learning for outlier detection, thus obtains a more precise and robust OOD evaluation.

4.3.1 NAS Framework for OOD Detection

In this section, we introduce the dense connectivity space and predictor-based NAS.

Dense Connectivity Search Space. Modern CNN architectures are composed of a series

of building blocks (i.e. stages), where each stage is responsible for transforming features at the same input resolution. Our search space follows the same paradigm but focuses on structural design in terms of connectivity between operators inside each stage. As shown in Figure 4.2, the dense connectivity search space has three stages, each consisting of three identical cells. The cells from different stages are independent and can be different. We represent each building cell as a Directed Acyclic Graph (DAG) to capture the edge connections between diversified vertices (i.e., operators). Each vertex in the DAG is an atomic convolutional operator selected from {Convolution 1×1, 3×3, 5×5} and {Depthwise Convolution 3×3, 5×5}. This constitutes a large search space covering several standard architectures, such as ResNet, DenseNet, etc.

Specifically, among all vertices in a DAG from stage s defined as $\mathcal{G}^{(s)} = (\mathcal{V}^{(s)}, \mathcal{E}^{(s)})$, v_0 is the input that receives the output from the previous cell, and v_K is the output that concatenate all leaf vertices in the DAG. Each intermediate vertex $v_i \in \{v_1, v_2..., v_{K-1}\}$ can connect to its preceding vertices and produce the output X_{v_i} with the assigned operators $op^{(i)}$:

$$X_{v_i} = op^{(i)}(Concat_{\{v_j|j<i, e_{ji}\in\mathcal{E}^{(k)}\}}[X_{v_j}]) \tag{4.6}$$

Given the selected K vertex operators, the entire architecture $\mathcal{A}$ can then be represented as the union of independent DAGs, i.e. $\mathcal{A} = \text{Arch}(\mathcal{G}^{(1)}, \mathcal{G}^{(2)}, \mathcal{G}^{(3)}; op^{(1)}, ...op^{(K)})$ and the searching goal is to find the optimal dense connectivity inside each DAG as follows:

$$\mathcal{E}^* = \arg\max_{\mathcal{E}} \text{Val}(\text{Arch}(\mathcal{G}^{(1)}, \mathcal{G}^{(2)}, \mathcal{G}^{(3)}; op^{(1)}, ...op^{(K)})). \tag{4.7}$$

where $\text{Val}(\cdot)$ is the validation function. In practice, we set K exceed the number of operator choices to enable more versatile dense connection. The total search space contains $2^{\frac{3K(K-1)}{2}}$ unique architectures.

Predictor-based NAS Framework. We utilize predictor-based NAS[32] to systematically explore beneficial structure patterns for OOD detection. Compared to other popular NAS methods that use weight sharing techniques, predictor-based NAS uses a neural predictor as a surrogate for ground-truth performance. This approach avoids co-adaptation

issues and provides trustworthy predictive performance, demonstrated as follows:

1. Randomly sample N architecture from the search space, train and evaluate their outlier detection performance. Construct $\mathcal{D}_{predictor} = (arch_i, z_i)_{i=1}^{N}$ where $arch_i$ is the encoding of the architecture, and z_i is AUROC.
2. Train a neural predictor on $\mathcal{D}_{predictor}$. Use the trained predictor to explore the search space for top-performing CNN architectures.
3. Retrain and evaluate the optimal CNN architecture.

Searching with Neural Predictor A neural predictor is trained using a carefully curated high-quality dataset, whose construction details will be elaborated in the next section. We employ a multi-layer perceptron as the predictor backbone and the training loss is computed as the sum of the mean squared error and ranking loss. Can we say regularized evolution? We utilize a modified version of the evolutionary algorithm (EA)[114] in conjunction with our trained predictor to comprehensively explore the vast search space for dense connection. Our EA is equipped with two types of mutation steps:

- *Adding/removing an edge from a selected DAG.*
- *Re-sampling a new DAG to replace the old one.*

To overcome the tendency of EA to converge to a local optimal solution, we incorporate an accepting-rejection mechanism inspired by the Metropolis-Hasting algorithm [115]. This allows us to maintain the best structure while still considering weaker candidates based on the difference between their score and the current best score. The acceptance rate is defined as

$$ac = min(1, exp((score^* - score)/T)) \tag{4.8}$$

where $score^*$ and $score$ denote the score of the current best and candidate structure respectively, as evaluated by our predictor.

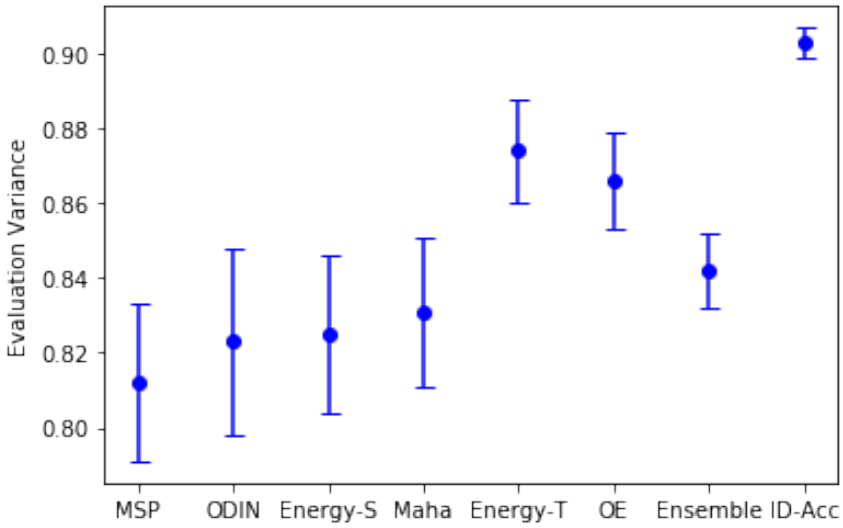


Figure 4.3: Evaluation variance of sampled networks from search space. Energy(S) and Energy(T) stands for energy scoring and training respectively.

4.3.2 Evaluation with Evolving Distillation

In this section, we identify the main challenges in OOD evaluation, and propose the solution to mitigate this issue.

Noisy Evaluation Issue in OOD Detection. When using predictor-based NAS, it is crucial to assess the accurate performance of each architecture to build a high-quality dataset for training the predictor. However, for OOD detection performance, the evaluation of neural architectures can be severely noisy. This can lead to inaccurate estimates of the true OOD performance of an architecture and bias the NAS search towards sub-optimal solutions.

We evaluate the noisy level of several post-training OOD detection methods - MSP, Mahalanobis distance (Maha), ODIN [105], energy score, and outlier training methods - OE, energy training. To do this, we use CIFAR-10 as our in-distribution dataset and CIFAR-100 as our out-distribution dataset. We randomly sample 10 architectures from our search space and initialize each one with 5 different seeds. We then train the models

for 20 epochs by minimizing the cross-entropy loss on the in-distribution data only. For outlier exposure methods, we further fine-tune 5 epochs by using 80M tiny images [116] as auxiliary outlier dataset. We assess the noise level of the trained models by calculating the variance of AUROC on CIFAR-10 versus CIFAR-100, while using in-distribution accuracy on CIFAR-10 as the baseline.

Based on our experiments, we find that the noise level of AUROC is significantly higher than that of ID-accuracy for both post-training and outlier exposure methods. This observation is consistent with our additional experiments, revealing poor Kendall's τ of the trained predictor for OOD detection. These results suggest that vanilla cross-entropy training with post-training OOD detection methods is strongly impacted by the noisy behavior resulting from the network's random initialization. However, we observe that a few epochs of fine-tuning with outlier exposure methods could substantially reduce the evaluation variance. To build a high-quality predictor, it is crucial to understand the root cause of the significant variance in OOD detection performance. This understanding will enable us to develop a more stable training procedure that can effectively reduce the evaluation noise for NAS.

Multi-view Learning with Outlier Distillation From a multi-view feature learning perspective [117], we can gain insight into the behavior of noisy OOD detection evaluation. Cross-entropy training with different initialization learns a subset of useful features for the classification task. Although different feature subsets may perform similarly on the classification task, they can have varied capabilities in identifying out-of-distribution data, leading to a high variance in OOD detection performance. To mitigate this problem, the outlier exposure method explicitly encourages the model to learn features that can distinguish between in-distribution and OOD samples. While the vanilla OE training aims to produce a uniform distribution on outlier sample, assigning equal probabilities to each class for outlier data can be misleading, particularly for near-OOD samples that are semantically close to the in-distribution data. This can hinder the model's ability to perform effective multi-view feature learning and may also hurt the performance of in-distribution accuracy.

We use Knowledge Distillation [118] to provide more accurate and informative supervision from both in-distribution and out-of-distribution training data. Our key idea is to encourage the model to imitate the behavior of a highly performing OOD detection model, enabling it to acquire beneficial multi-view features. Specifically, we have access to a labeled in-distribution training dataset $\mathcal{D}_{in}^{train}$ and an unlabeled outlier dataset $\mathcal{D}_{out}^{train}$. Assume that the neural network use a softmax layer to produces class probabilities on each class i:

$$P_i(x) = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)}, \tag{4.9}$$

where T is the temperature scaling parameter to affect the final output distribution. Let $P^t(x) = (P_1^t(x), ..., P_C^t(x))$ $(P^s(x))$ denotes the probability output over C classes from the teacher model (student model). The distillation loss is the distance between the teacher and student predictions, given by:

$$L^{\text{DS}}(x) = -\sum_i [P_i^t(x) \cdot \log P_i^s(x)]. \tag{4.10}$$

It should be noted that the distillation loss is added to both in-distribution and outlier training data, encouraging the model to imitate the teacher's predictive behavior on both data types. In addition, we also include the cross-entropy loss $L^{\text{CE}}(y; P^s(x))$ for in-distribution data and $L^{\text{CE}}(\mathcal{U}; P^s(x))$ for outlier data respectively, where $\mathcal{U}$ denotes the uniform distribution over C classes. We introduce hyper-parameters α and β to control the trade-off between the distillation loss and cross-entropy loss, and use λ to balance the contribution of the in-distribution and outlier losses. The final training loss can be expressed as follows:

$$\begin{aligned} L = {} & \mathbb{E}_{(x,y)\sim\mathcal{D}_{in}^{train}}[\alpha L^{\text{DS}}(x) + (1-\alpha)L^{\text{CE}}(y; P^s(x))] \\ & + \lambda\, \mathbb{E}_{x\sim\mathcal{D}_{out}^{train}}[\beta L^{\text{DS}}(x) + (1-\beta)L^{\text{CE}}(\mathcal{U}; P^s(x))] \end{aligned} \tag{4.11}$$

Evolving Distillation Algorithm. Obtaining a high-performing OOD detection teacher model is crucial in providing valuable knowledge for student learning. Rather than incurring a significant cost to train another model, we can obtain our teacher model by selecting from the sampled networks and using an ensemble of them, see algorithm 8. Specifically,

we first sample N architectures from our search space and train them using the vanilla outlier exposure loss. Next, we evaluate the OOD detection performance of the trained architectures on the validation set. Then in each iteration, we select the top-performing architectures and ensemble their outputs as the teacher's output. In practice, we find that a combination of top-performing and randomly selected models can regularize the training and lead to better exploration. We then retrain all architectures using equation 4.11 and re-evaluate their performance to prepare for the next iteration. This procedure can be repeated multiple times, resulting in an evolving ensemble of teacher models that provides a progressively improved training loss landscape. At the end of the process, we save N pairs of (arch, AUROC) for predictor training. The final teacher models from last iteration are also saved to provide supervision for re-training.

Algorithm 2: Evolving OOD Distillation

Sample N architecture, train on $\mathcal{D}_{in}^{train}$ and $\mathcal{D}_{out}^{train}$ with cross-entropy and outlier exposure.

Evaluate trained archs on $\mathcal{D}^{val}$, then add N pair of (arch, AUROC) to $\mathcal{H}$.

repeat

  Create teacher set $\mathcal{T}$ by a combination of top $K\%$ and randomly selected $k\%$ archs from $\mathcal{H}$.

  Train all N archs with ensemble teacher using equation 4.11

  Evaluate trained archs on $\mathcal{D}^{val}$, then update $\mathcal{H}$.

until *maximum iteration is reached*;

Use $\mathcal{H}$ for predictor training; Save teacher set $\mathcal{T}$ for later use.

4.4 Experiments

In this section, we empirically evaluate the performance of DCSOD on near-OOD detection task. First, we present the detailed configuration of architecture search, evaluation

Table 4.1: OOD AUROC and in-distribution classification accuracy on CIFAR. OT stands for Outlier Training, () indicates improvement over ERD.

Methods		CIFAR-10 vs CIFAR-100				CIFAR-100 vs CIFAR-10			
		MSP	Maha	OT	ID-Acc	MSP	Maha	OT	ID-Acc
Standard Arch	ResNet-110	86.4	86.3	92.1	94.2	74.1	73.8	76.2	77.4
	WideResNet-40-2	87.5	87.7	93.0	**95.2**	74.4	74.9	75.6	78.1
	DenseNet-100-12	87.9	88.1	92.7	94.6	75.2	75.5	76.8	77.1
NAS BL	Random Search	88.3	88.4	92.3	94.7	76.1	76.4	78.3	77.3
	Regularize EA	88.9	89.2	93.2	94.9	77.1	77.5	79.5	77.9
Predictor NAS	Cross Entropy	88.1	88.3	92.4	94.8	76.2	76.3	78.1	77.5
	DCSOD (Ours)	**90.6**	**91.0**	**95.8(+0.8)**	95.1	**89.2**	**89.9**	**95.1(+1.1)**	**78.3**

metrics and baselines for comparison. Next, we demonstrate the empirical results on CIFAR benchmarks, followed by a comprehensive ablation study to examine the effect of hyper-parameters. Finally, we visualize the loss landscape and optimal searched cell.

4.4.1 Experiment Setup

Dataset. We evaluate our proposed paradigm on both the CIFAR-10 versus CIFAR-100 and CIFAR-100 versus CIFAR-10 detection tasks, where the former is an in-distribution dataset and the latter is an out-of-distribution dataset. We consider two scenarios: with or without auxiliary outlier training data. ("OT" stands for outlier training, corresponding to the scenario with auxiliary outlier data.) We utilize a modified version of 80 Million Tiny Images as auxiliary outlier training data, wherein all examples that appear in the CIFAR dataset are removed.

Baselines. To focus our study on the impact of network architecture, we consider a few popular out-of-distribution (OOD) detection techniques. Specifically, for the case without auxiliary outlier data, we consider the MSP and Mahalanobis distance methods, while for the available auxiliary outlier cases, we consider the vanilla outlier exposure method. Our baseline models consist of three types: (1) widely used human-designed architectures,

including ResNet, Wide-ResNet [119], and DenseNet. (2) NAS baselines include random search and predictor-free regularized EA. (3) predictor-based NAS training using cross-entropy loss. We also compared our results with the current CNN state-of-the-art method ERD [120].

Search Configuration. To construct the neural predictor, we randomly sample 2000 architectures from our search space. Each candidate architecture was first pre-trained on 50% of the CIFAR dataset for 20 epochs, using SGD with an initial learning rate of 0.1 and batch size of 128 via a cosine learning rate schedule. The trained models were evaluated using the aforementioned OOD detection methods. To train for distillation, we select the top 10 performing and 1 randomly selected architectures as teachers. Following the advice from [118], we utilized a high temperature of $T = 20$ to facilitate more knowledge transfer from the teacher to the student model. The hyper-parameters chosen for the cases with (without) auxiliary outlier data are $(\lambda, \alpha, \beta) = 0.5, 0.5, 0.2$ $((\lambda, \alpha) = 0, 0.5)$, which were determined using validation data. We repeat the iterative process for three times.

The collected dataset, denoted as $\mathcal{D}_{predictor} = (arch_i, auroc_i)_{i=1}^{2000}$, is split into training data (85%) and testing data (15%). We use a multi-layer perceptron as the predictor backbone, and the training loss was computed as the sum of the mean squared error and ranking loss. We employed a modified version of the evolutionary algorithm (EA)[114] in conjunction with our trained predictor to comprehensively explore the vast search space for dense connections. Our EA was equipped with two types of mutation steps: either *adding/removing an edge from a selected DAG* or *re-sampling a new DAG to replace the old one*. Finally, we took the optimal searched architecture and re-trained (using previously saved teacher) on the full CIFAR dataset for 300 epochs and evaluate it on the test set.

Evaluation Metrics. We assess both the quality of neural predictors and the out-of-distribution (OOD) detection performance of each architecture. To measure the ranking quality provided by the predictor, we use Kendall's Tau and Pearson's Rho. For evaluating OOD detection performance, we use the area under the receiver operating characteristic curve (AUROC), as it exhibits less variance compared to other metrics such as FPR95.

Additionally, we report the in-distribution classification accuracy to ensure that the model can maintain its performance on the original task.

4.4.2 CIFAR Benchmark Results

As shown in Table 4.1, DCSOD consistently outperforms all baselines across various OOD detection methods and configurations. Specifically, our method has improved the previous state-of-the-art (SOTA) ERD on CIFAR by ∼1.0%. For NAS baseline, without auxiliary outlier data, DCSOD improves Regularized EA performance by an average of 2.6% (13.3%) for CIFAR-10 vs. CIFAR-100 (CIFAR-100 vs. CIFAR-10). When trained with auxiliary outlier data, the optimal architecture searched by DCSOD outperforms regularized EA by 3.4% for CIFAR-10 vs. CIFAR-100 (from 92.4% to 95.8%), and by 15.6% for CIFAR-100 vs. CIFAR-10 (from 79.5% to 95.1%). However, NAS predictor with cross-entropy training only achieves similar performance as random NAS search due to its low Kendall's Tau, as shown in Table 4.2. These results suggest that the neural predictor obtained from DCSOD can more efficiently explore the vast search space for optimal architectures. Additionally, the distillation training loss, constructed by teachers from the evolving distillation algorithm, enhances OOD detection performance and helps maintain classification accuracy.

Table 4.2: Evaluation of neural predictor on CIFAR-10 vs CIFAR-100. High-quality predictors should exhibit high Kendall's τ and Pearson's ρ.

Predictors	**Kendall's τ**	**Pearson's ρ**
In-dist Acc	0.77	0.86
Cross Entropy-MSP	0.44	0.57
Cross Entropy-OT	0.51	0.62
DCSOD-MSP	0.72	0.81
DCSOD-OT	0.74	0.83

4.4.3 Ablation Study

Number of iteration. The trade-off between the number of iterations and sample size in the evolving distillation algorithm plays a central role in determining the quality of the neural predictor. To find the appropriate trade-off under a given computational budget, we experimented with multiple iteration-sample size combinations with similar computational costs. For instance, 3000-0 indicates no iteration for teacher selection and distillation training. We conducted experiments on the CIFAR-10 vs. CIFAR-100 task with access to auxiliary outlier data. The results are shown in Figure 4.4: vanilla outlier exposure training suffers from a high evaluation variance issue, and the poor quality predictor cannot effectively guide architecture search. With only one iteration of evolving distillation, there is a sharp improvement in both predictor Kendall's τ and final AUROC performance. From the 1800-1 to 900-3 combinations, a fair improvement can still be witnessed on AUROC, even though Kendall's τ became saturated. We hypothesize that this is because a better distillation loss term can enhance the final performance of OOD detection. A better version of teacher also reduces the need for a large sample size to train the predictor. However, if we keep reducing the sample size, the predictor will become under-trained without seeing enough examples. We also considered another setup by fixing the sample size to 2000 and increasing the number of iterations. The results show that increasing iterations can consistently increase predictor quality and OOD detection performance, and the improvement starts to saturate after 3 iterations.

Hyper-parameters in Outlier Distillation. We conducted a study on the effect of hyper-parameters λ, α, and β for outlier distillation training, see Table 4.3. We began with vanilla cross-entropy training without any distillation loss term. As expected, this setup suffered from high OOD evaluation issues and gave poor performance. After adding the distillation loss on in-distribution data, we observed a significant increase in Kendall's τ. This suggests that imitating the out-performing teacher's behavior only on in-distribution data can facilitate multi-view feature learning and stabilize the OOD detection performance.

Next, we used auxiliary outlier data, but with only $L^{\text{CE}}(\mathcal{U}; P^s(x))$, and observed a no-

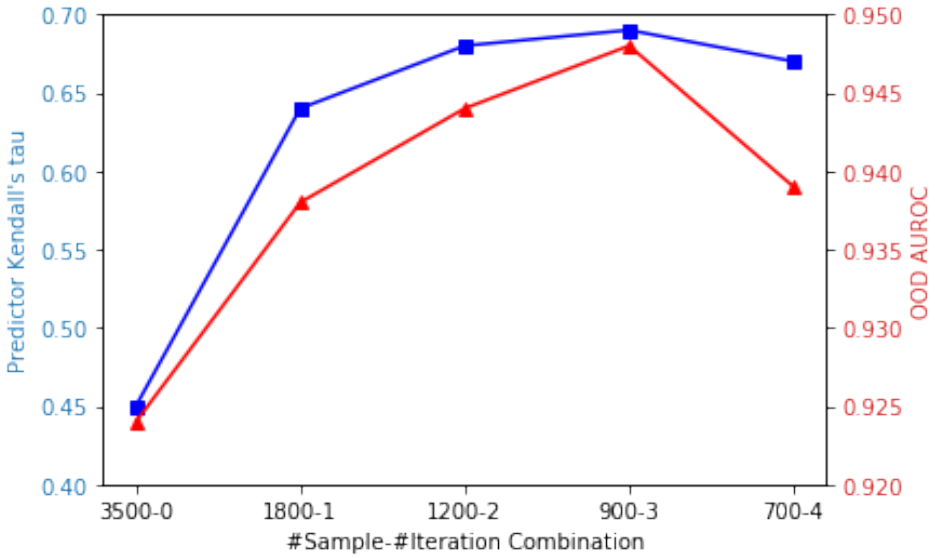


Figure 4.4: Comparison of out-of-distribution (OOD) AUROC and predictor Kendall's τ for different sample-iteration trade-offs.

ticeable decrease in in-distribution classification accuracy. Finally, we added the distillation loss on outlier data, and the accuracy increased to the level before. This indicates that the outlier distillation term is beneficial to both OOD detection and the original classification task.

Table 4.3: Ablation study of hyper-parameter λ, α, β.

Hyper-parameters (λ, α, β)	**Kendall's** τ	**AUROC**	**ID-Acc**
(0, 0, -)	0.44	88.1	94.8
(0, 0.5, -)	0.67	89.4	95.1
(0.5, 0.5, 0)	0.71	93.7	94.3
(0.5, 0.5, 0.2)	0.73	95.8	95.1

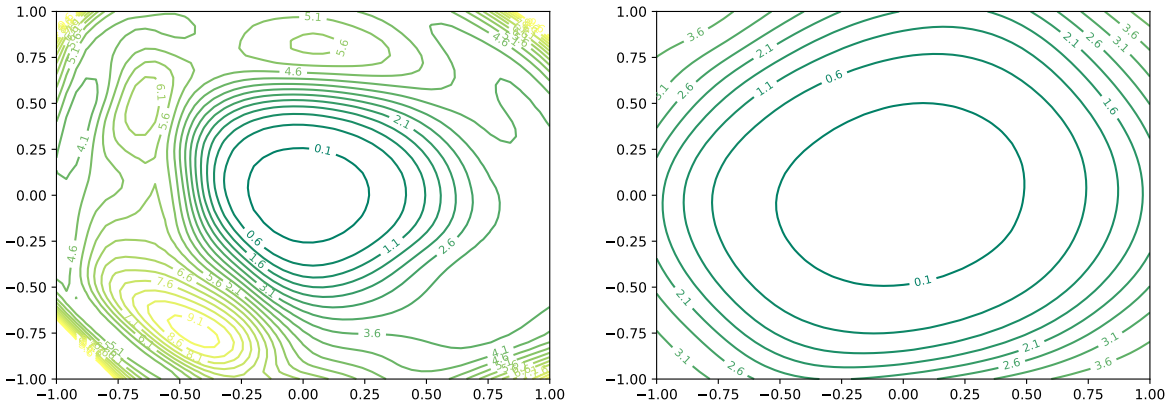


Figure 4.5: Visualization of the loss landscape. The outlier distillation loss had smoother and wider minima with a convex-like contour.

4.4.4 Visualization

We utilized the "filter normalization" technique [121] to visualize the landscape of the training loss function searched by the evolving distillation algorithm. We took the architecture searched by DCSOD and compared the loss landscape for the vanilla cross-entropy and outlier distillation loss. Specifically, we plotted the loss near each minimizer separately using two random filter-normalized directions. The outlier distillation loss had smoother and wider minima with a convex-like contour and fewer local minima. This suggests that the outlier distillation loss is easier to train and leads to stable evaluation performance on out-of-distribution (OOD) detection.

Last, we visualize several representative cell from top-performing architectures searched by DCSOD, see Fig 4.6. The discovered models contain versatile structural wirings that effectively aggregate and communicate information within different parts of the CNN architecture, showing superior accuracy/AUROC over hand-crafted arts and NAS-crafted arts. Thus, we answer the question that, allowing dense connectivity within CNN architectures help outlier detection evaluation, and boost the performance of outlier detectors.

4.5 Conclusion

In this work, we pioneer the study on the relationship between CNN architectures and their performance on outlier detection. We propose a novel paradigm, **D**ense **C**onnectivity **S**earch of **O**utlier **D**etector, that leverages Neural Architecture Search (NAS) to explore the structural wiring of CNN architectures containing dense connectivity of versatile convolution operators. DCSOD utilizes a neural predictor to guide the search, and improves the evaluation of candidate outlier detectors by proposing evolving distillation algorithm. Evaluation results show that our NAS-crafted models achieve the state-of-the-art performance on CIFAR benchmarks, distinguishing out-of-distribution examples from CIFAR-10/CIFAR-100 benchmarks with higher accuracy and AUROC, answering the question that dense connectivity is beneficial for the construction of outlier detectors.

4.6 Appendix

4.6.1 Evolving Distillation with Smaller Overhead

The main concept behind the evolution OOD distillation algorithm is to continually improve the teacher and utilize it to guide the training of sampled architectures. In the original algorithm, all the initially sampled architectures were iteratively trained and evaluated with the evolving teachers, resulting in a significant computational overhead. However, a more efficient approach is to first sample a small batch of architectures and progressively increase the sample size in each iteration. This can be achieved by sampling new architectures and training them with the teachers selected in the current iteration. The newly trained and evaluated architectures can then be added to the pool $\mathcal{P}$ for teacher selection in the next iteration. By the final iteration, all previously sampled architectures are retrained and evaluated to update $\mathcal{H}$ and the teacher set $\mathcal{T}$. This version of the algorithm allows for a larger dataset to be used in predictor training, without compromising the OOD evaluation quality.

Algorithm 3: Evolving OOD Distillation

Sample P archs, train on $\mathcal{D}_{in}^{train}$ and $\mathcal{D}_{out}^{train}$ with cross-entropy and outlier exposure.

Evaluate trained archs on $\mathcal{D}^{val}$, then add N pair of (arch, AUROC) to $\mathcal{P}$ and $\mathcal{H}$.

repeat

  Create teacher set $\mathcal{T}$ by a combination of top $K\%$ and randomly selected $k\%$ archs from $\mathcal{P}$.

  Sample q new archs, train new archs with ensemble teachers using equation 4.11.

  Evaluate on $\mathcal{D}^{val}$, update $\mathcal{P}$ by adding q newly evaluated archs and removing 'dead' archs; also add q new archs to $\mathcal{H}$.

until *maximum iteration is reached*;

Retrain and evaluate all archs from $\mathcal{H}$. Select teacher set $\mathcal{T}$ for later use.

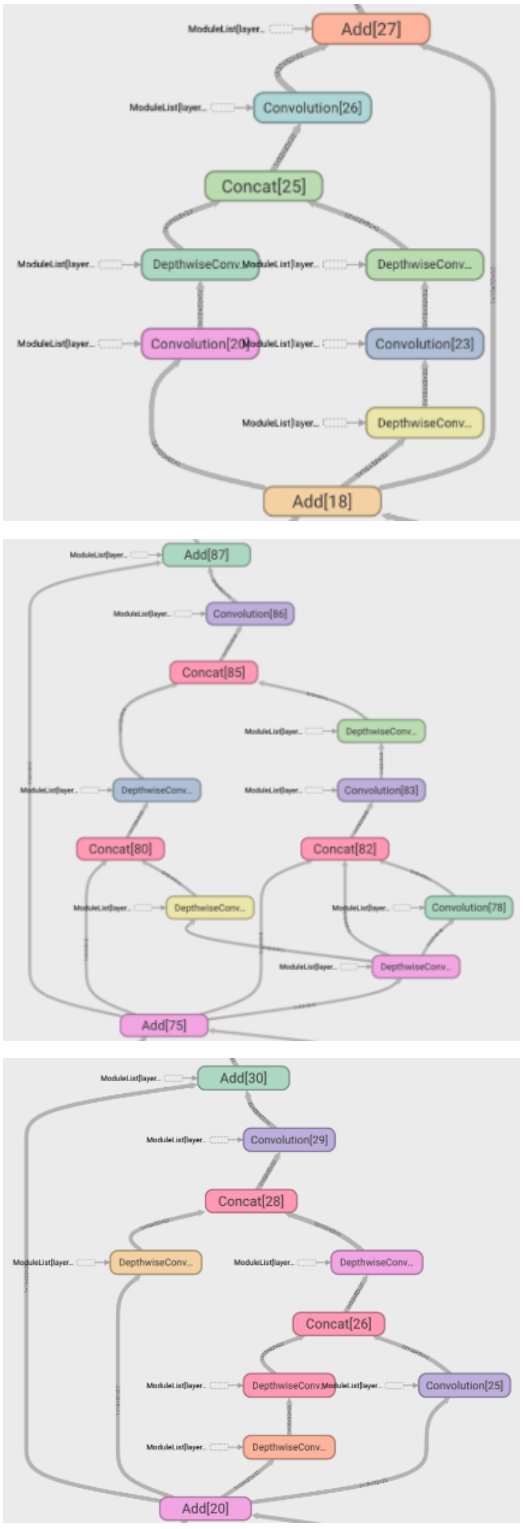


Figure 4.6: Visualization of representative cell from DCSOD. The discovered models contain versatile structural wirings between various CNN operators.

Chapter 5

Conclusion

This dissertation focuses on the architecutal influence of neural network for training process and model performance and robustness. In the field of NLP, we address the training deficiency of text VAEs with auto-regressive decoders through two approaches. First, we provide a novel two-path interpretation to explain the KL vanishing issue, and identify its source as a lack of good latent codes at the beginning of decoder training. The cyclical schedule can progressively improve the performance, by leveraging good latent codes learned in the previous cycles as warm re-starts. Second, we have introduced the semi-implicit approximate in the posterior learning of VAEs to infer text representations. The flexibility of SI helps reduce the representational bottleneck in the latent space and thus alleviate the KL vanishing issue. These two approaches together contributes to addressing the training deficiency, enabling controllable text generation and abstract reasoning with text. In the field of computer vision, we pioneer the study on the relationship between CNN architectures and their performance on outlier detection. We propose a novel paradigm, **D**ense **C**onnectivity **S**earch of **O**utlier **D**etector, that leverages Neural Architecture Search (NAS) to explore the structural wiring of CNN architectures containing dense connectivity of versatile convolution operators. DCSOD utilizes a neural predictor to guide the search, and improves the evaluation of candidate outlier detectors by proposing evolving distillation algorithm. Evaluation results show that our NAS-crafted models achieve the state-of-the-art performance on CIFAR benchmarks, distinguishing out-of-distribution examples from CIFAR-10/CIFAR-100 benchmarks with higher accuracy and AUROC, answering the question that dense connectivity is beneficial for the construction of outlier detectors.

We have identified several potential avenues for future research, including: (1) Exploring the effectiveness of pre-trained transformer-based generative models on large text corpora. By leveraging the smooth latent space learned by the pre-trained text generative model, we

may be able to achieve better generalization especially for language tasks under low-resource regime. (2) Building upon our investigation of the impact of dense connectivity in CNNs on confidence-score based OOD detection, we suggest two potential extensions: Firstly, expanding the search space to include transformer-based models; Secondly, exploring the architectural influence on generative model-based OOD detection. These extensions could help to complete our understanding of the structural factors affecting OOD detection and enhance the robustness of models in more challenging scenarios.

Bibliography

[1] Athanasios Voulodimos, Nikolaos Doulamis, Anastasios Doulamis, Eftychios Protopapadakis, et al. Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience*, 2018, 2018.

[2] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. Recent trends in deep learning based natural language processing. *ieee Computational intelligenCe magazine*, 13(3):55–75, 2018.

[3] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. Deep learning based recommender system: A survey and new perspectives. *ACM computing surveys (CSUR)*, 52(1):1–38, 2019.

[4] Sorin Grigorescu, Bogdan Trasnea, Tiberiu Cocias, and Gigel Macesanu. A survey of deep learning techniques for autonomous driving. *Journal of Field Robotics*, 37(3):362–386, 2020.

[5] Riccardo Miotto, Fei Wang, Shuang Wang, Xiaoqian Jiang, and Joel T Dudley. Deep learning for healthcare: review, opportunities and challenges. *Briefings in bioinformatics*, 19(6):1236–1246, 2018.

[6] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

[7] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[8] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

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

[10] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[11] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[12] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

[13] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[14] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

[15] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[16] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[17] Sepp Hochreiter and Jurgen Schmidhuber. Long short-term memory. *Neural computation*, 1997.

[18] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681, 1997.

[19] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[20] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[21] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[22] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[23] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9,

2019.

[24] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[25] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

[26] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

[27] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *The Journal of Machine Learning Research*, 20(1):1997–2017, 2019.

[28] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc 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.

[29] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameters sharing. In *International conference on machine learning*, pages 4095–4104. PMLR, 2018.

[30] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

[31] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10734–10742, 2019.

[32] Wei Wen, Hanxiao Liu, Yiran Chen, Hai Li, Gabriel Bender, and Pieter-Jan Kindermans. Neural predictor for neural architecture search. In *European Conference on Computer Vision*, pages 660–676. Springer, 2020.

[33] Colin White, Willie Neiswanger, and Yash Savani. Bananas: Bayesian optimization with neural architectures for neural architecture search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 10293–10301, 2021.

[34] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. *arXiv preprint arXiv:1908.09791*, 2019.

[35] Jiahui Yu, Pengchong Jin, Hanxiao Liu, Gabriel Bender, Pieter-Jan Kindermans, Mingxing Tan, Thomas Huang, Xiaodan Song, Ruoming Pang, and Quoc Le. Bignas: Scaling up neural architecture search with big single-stage models. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VII 16*, pages 702–717. Springer, 2020.

[36] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *ICLR*, 2013.

[37] Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*, 2015.

[38] Jingkang Yang, Kaiyang Zhou, Yixuan Li, and Ziwei Liu. Generalized out-of-distribution detection: A survey. *arXiv preprint arXiv:2110.11334*, 2021.

[39] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic back-propagation and approximate inference in deep generative models. *ICML*, 2014.

[40] Yishu Miao, Lei Yu, and Phil Blunsom. Neural variational inference for text processing. In *ICML*, 2016.

[41] Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. *ACL*, 2017.

[42] Tsung-Hsien Wen, Yishu Miao, Phil Blunsom, and Steve Young. Latent intention dialogue models. *ICML*, 2017.

[43] Weidi Xu, Haoze Sun, Chao Deng, and Ying Tan. Variational autoencoder for semi-supervised text classification. In *AAAI*, 2017.

[44] Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. Toward controlled generation of text. *ICML*, 2017.

[45] Yishu Miao and Phil Blunsom. Language as a latent variable: Discrete generative models for sentence compression. *EMNLP*, 2016.

[46] Zichao Yang, Zhiting Hu, Ruslan Salakhutdinov, and Taylor Berg-Kirkpatrick. Improved variational autoencoders for text modeling using dilated convolutions. *ICML*, 2017.

[47] Adji B Dieng, Yoon Kim, Alexander M Rush, and David M Blei. Avoiding latent

variable collapse with generative skip models. *arXiv preprint arXiv:1807.04863*, 2018.

[48] Yoon Kim, Sam Wiseman, Andrew C Miller, David Sontag, and Alexander M Rush. Semi-amortized variational autoencoders. *ICML*, 2018.

[49] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[50] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[51] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *ICLR*, 2017.

[52] Marco Fraccaro, Søren Kaae Sønderby, Ulrich Paquet, and Ole Winther. Sequential neural models with stochastic layers. In *NIPS*, 2016.

[53] Anirudh Goyal Alias Parth Goyal, Alessandro Sordoni, Marc-Alexandre Côté, Nan Rosemary Ke, and Yoshua Bengio. Z-forcing: Training stochastic recurrent networks. In *NIPS*, 2017.

[54] Guokun Lai, Bohan Li, Guoqing Zheng, and Yiming Yang. Stochastic wavenet: A generative latent variable model for sequential data. *ICML workshop*, 2018.

[55] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *ICLR workshop*, 2016.

[56] Ricky TQ Chen, Xuechen Li, Roger Grosse, and David Duvenaud. Isolating sources of disentanglement in VAEs. *NIPS*, 2018.

[57] Matthew D Hoffman and Matthew J Johnson. Elbo surgery: yet another way to carve up the variational evidence lower bound. In *Workshop in Advances in Approximate Bayesian Inference, NIPS*, 2016.

[58] Chunyuan Li, Hao Liu, Changyou Chen, Yuchen Pu, Liqun Chen, Ricardo Henao, and Lawrence Carin. ALICE: Towards understanding adversarial learning for joint distribution matching. In *NIPS*, 2017.

[59] Rui Shu, Hung H Bui, Shengjia Zhao, Mykel J Kochenderfer, and Stefano Ermon. Amortized inference regularization. *NIPS*, 2018.

[60] Shengjia Zhao, Jiaming Song, and Stefano Ermon. InfoVAE: Information maximizing variational autoencoders. *AAAI*, 2019.

[61] Junxian He, Daniel Spokoyny, Graham Neubig, and Taylor Berg-Kirkpatrick. Lagging inference networks and posterior collapse in variational autoencoders. *ICLR*, 2019.

[62] Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic variational inference. *The Journal of Machine Learning Research*, 2013.

[63] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. *ICLR*, 2017.

[64] Alexander Alemi, Ben Poole, Ian Fischer, Joshua Dillon, Rif A Saurous, and Kevin Murphy. Fixing a broken ELBO. In *ICML*, 2018.

[65] Hyunjik Kim and Andriy Mnih. Disentangling by factorising. *ICML*, 2018.

[66] Leslie N Smith. Cyclical learning rates for training neural networks. In *WACV*. IEEE, 2017.

[67] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *ICLR*, 2017.

[68] Leslie N Smith and Nicholay Topin. Super-convergence: Very fast training of residual networks using large learning rates. *arXiv preprint arXiv:1708.07120*, 2017.

[69] Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E Hopcroft, and Kilian Q Weinberger. Snapshot ensembles: Train 1, get m for free. *ICLR*, 2017.

[70] Ruqi Zhang, Chunyuan Li, Jianyi Zhang, Changyou Chen, and Andrew Gordon Wilson. Cyclical stochastic gradient mcmc for bayesian deep learning. *arXiv preprint arXiv:1902.03932*, 2019.

[71] Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 1993.

[72] J Godfrey and E Holliman. Switchboard-1 release 2: Linguistic data consortium. *SWITCHBOARD: A User's Manual*, 1997.

[73] Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C Courville, and Joelle Pineau. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*, 2016.

[74] Boxing Chen and Colin Cherry. A systematic comparison of smoothing techniques for sentence-level bleu. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 362–367, 2014.

[75] Chia-Wei Liu, Ryan Lowe, Iulian V Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. *arXiv preprint arXiv:1603.08023*, 2016.

[76] Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. Style transfer from non-parallel text by cross-alignment. In *NIPS*, 2017.

[77] Yizhe Zhang, Dinghan Shen, Guoyin Wang, Zhe Gan, Ricardo Henao, and Lawrence Carin. Deconvolutional paragraph representation learning. In *NIPS*, 2017.

[78] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[79] Yoon Kim, Sam Wiseman, and Alexander M Rush. A tutorial on deep latent variable models of natural language. *arXiv preprint arXiv:1812.06834*, 2018.

[80] Hao Fu, Chunyuan Li, Xiaodong Liu, Jianfeng Gao, Asli Celikyilmaz, and Lawrence Carin. Cyclical annealing schedule: A simple approach to mitigating KL vanishing. *NAACL*, 2019.

[81] Chris Cremer, Xuechen Li, and David Duvenaud. Inference suboptimality in variational autoencoders. *ICML*, 2018.

[82] Xiaodong Gu, Kyunghyun Cho, Jungwoo Ha, and Sunghun Kim. DialogWAE: Multimodal response generation with conditional Wasserstein auto-encoder. *ICLR*, 2019.

[83] Mingzhang Yin and Mingyuan Zhou. Semi-implicit variational inference. *ICML*, 2018.

[84] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.

[85] Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks. In *ICML*, 2017.

[86] Yihao Feng, Dilin Wang, and Qiang Liu. Learning to draw samples with amortized stein variational gradient descent. *UAI*, 2017.

[87] Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*, 2015.

[88] John J Godfrey and Edward Holliman. Switchboard-1 release 2. *Linguistic Data Consortium, Philadelphia, 926:927*, 1997.

[89] Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[90] Xiaoyu Shen, Hui Su, Shuzi Niu, and Vera Demberg. Improving variational encoder-decoders in dialogue generation. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[91] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *EMNLP*, 2014.

[92] Yu Weng, Tianbao Zhou, Yujie Li, and Xiaoyu Qiu. Nas-unet: Neural architecture search for medical image segmentation. *IEEE access*, 7:44247–44257, 2019.

[93] Hu Cao, Yueyue Wang, Joy Chen, Dongsheng Jiang, Xiaopeng Zhang, Qi Tian, and Manning Wang. Swin-unet: Unet-like pure transformer for medical image segmentation. In *Computer Vision–ECCV 2022 Workshops: Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part III*, pages 205–218. Springer, 2023.

[94] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[95] Haotian Tang, Zhijian Liu, Shengyu Zhao, Yujun Lin, Ji Lin, Hanrui Wang, and Song Han. Searching efficient 3d architectures with sparse point-voxel convolution. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVIII*, pages 685–702. Springer, 2020.

[96] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5693–5703, 2019.

[97] Yuanhao Cai, Zhicheng Wang, Zhengxiong Luo, Binyi Yin, Angang Du, Haoqian Wang, Xiangyu Zhang, Xinyu Zhou, Erjin Zhou, and Jian Sun. Learning delicate local representations for multi-person pose estimation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, pages 455–472. Springer, 2020.

[98] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016.

[99] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. *Advances in neural information processing systems*, 31, 2018.

[100] Weitang Liu, Xiaoyun Wang, John Owens, and Yixuan Li. Energy-based out-of-distribution detection. *Advances in neural information processing systems*, 33:21464–21475, 2020.

[101] Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Gorur, and Balaji Lakshminarayanan. Hybrid models with deep and invertible features. In *International Conference on Machine Learning*, pages 4723–4732. PMLR, 2019.

[102] Hongjie Zhang, Ang Li, Jie Guo, and Yanwen Guo. Hybrid models for open set recognition. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, pages 102–117. Springer, 2020.

[103] Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. Deep anomaly detection with outlier exposure. *arXiv preprint arXiv:1812.04606*, 2018.

[104] Aristotelis-Angelos Papadopoulos, Mohammad Reza Rajati, Nazim Shaikh, and Jiamian Wang. Outlier exposure with confidence control for out-of-distribution detection. *Neurocomputing*, 441:138–150, 2021.

[105] Shiyu Liang, Yixuan Li, and Rayadurgam Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. *arXiv preprint arXiv:1706.02690*, 2017.

[106] Dan Hendrycks, Xiaoyuan Liu, Eric Wallace, Adam Dziedzic, Rishabh Krishnan, and Dawn Song. Pretrained transformers improve out-of-distribution robustness. *arXiv preprint arXiv:2004.06100*, 2020.

[107] Rajat Koner, Poulami Sinhamahapatra, Karsten Roscher, Stephan Günnemann, and Volker Tresp. Oodformer: Out-of-distribution detection transformer. *arXiv preprint arXiv:2107.08976*, 2021.

[108] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[109] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2820–2828, 2019.

[110] Chenxi Liu, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, Wei Hua, Alan L Yuille, and Li Fei-Fei. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 82–92, 2019.

[111] Saining Xie, Alexander Kirillov, Ross Girshick, and Kaiming He. Exploring randomly wired neural networks for image recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1284–1293, 2019.

[112] Renqian Luo, Fei Tian, Tao Qin, Enhong Chen, and Tie-Yan Liu. Neural architecture optimization. *Advances in neural information processing systems*, 31, 2018.

[113] Hanxiao Liu, Karen Simonyan, Oriol Vinyals, Chrisantha Fernando, and Koray Kavukcuoglu. Hierarchical representations for efficient architecture search. *arXiv preprint arXiv:1711.00436*, 2017.

[114] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pages 4780–4789, 2019.

[115] Siddhartha Chib and Edward Greenberg. Understanding the metropolis-hastings algorithm. *The american statistician*, 49(4):327–335, 1995.

[116] Antonio Torralba, Rob Fergus, and William T Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 30(11):1958–1970, 2008.

[117] Zeyuan Allen-Zhu and Yuanzhi Li. Towards understanding ensemble, knowledge distillation and self-distillation in deep learning. *arXiv preprint arXiv:2012.09816*, 2020.

[118] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[119] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[120] Alexandru Tifrea, Eric Stavarache, and Fanny Yang. Semi-supervised novelty detection using ensembles with regularized disagreement. In *Uncertainty in Artificial Intelligence*, pages 1939–1948. PMLR, 2022.

[121] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. *Advances in neural information processing systems*, 31, 2018.

Biography

Hao Fu is currently a PhD student at Duke University in the Electrical and Computer Engineering department, advised by Professor Yiran Chen. He obtained his Bachelor's degree from Tsinghua University in Beijing, China and his Master's degree from Columbia University in New York, USA. His research interests are primarily focused on the application of deep learning and neural architecture search in computer vision and natural language processing. His work has been published in several top-tier conferences such as NeurIPS, EMNLP, NAACL, ICASSP, and Interspeech, and has received 1300+ citations (as of May 2023).