Flexible Text Modeling with Semi-Implicit Latent Representations

Hao Fu
Duke University
hao.fu@duke.edu

Chunyuan Li
Microsoft Research
chunyl@microsoft.com

Ke Bai
Duke University
ke.bai@duke.edu

Jianfeng Gao
Microsoft Research
jfgao@microsoft.com

Lawrence Carin
Duke University
lcarin@duke.edu

Abstract

Variational autoencoders (VAEs) have been used recently to learn probabilistic representations in many natural language processing (NLP) applications. Training deficiency has been witnessed when an auto-regressive decoder is used: the learned latent codes becomes almost identical to the prior distribution (termed "posterior collapse," resulting in a vanishing of the Kullback-Leibler term in the variational expression). We hypothesize that the source of deficiency is due partially to the approximate Gaussian posterior often used in variational inference. We use *semi-implicit* (SI) representations for the latent distributions of the natural languages. It extends the commonly used Gaussian distribution family, by mixing the variational parameter with a flexible implicit distribution. The increasing representational power of SI is demonstrated on two NLP tasks, showing that it provides more informative latent codes in language modeling, and generates more diverse dialog responses.

1 Introduction

Deep latent-variable models such as variational autoencoders (VAEs) [13, 21] are becoming increasingly popular in natural language processing (NLP). They have contributed to fundamental advances in many NLP tasks, such as language modeling [1, 17] and dialog response generation [27, 25]. For a text sequence of length T, $\boldsymbol{x} = [x_1, \cdots, x_T]$, neural language models [18] 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, preventing them from reasoning about data at an abstract level. For instance, language models don't allow the sort of neural sentence manipulations showcased in [1].

The VAE was introduced to fill the gap, simultaneously learning generative models with higher-quality sentence samples while learning an efficient inference network [13, 21, 1]. The *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:

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

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 *encoder*, implemented via a $\boldsymbol{\phi}$-parameterized neural

4th workshop on Bayesian Deep Learning (NeurIPS 2019), Vancouver, Canada.

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

Figure 1: Illustration of learning bottleneck in the two different methods. For the VAE model with an auto-regressive decoder, two paths are considered from x for its reconstruction: Path A via $\{a\ ;\ g\}$ and Path B via $\cdot$. The circles with blue, green and red indicate observed, latent, reconstructed variables, respectively. (a) Vanilla VAE with Gaussian posteriors represents the observation x as Gaussian latent code z, which encodes all information flow in Path A to decode its reconstruction. (b) Semi-implicit VAE represents the latent code of observation x as the mixture of Gaussian $\{z_k\}_{k=0}^K$, whose variational parameter λ_k is generated using x and auxiliary noise samples ϵ_k.

network. It yields the evidence lower bound (ELBO) as an objective:

$$\log p(x) \quad \mathcal{L} = E \quad R\ ;\ \text{with} \tag{2}$$

$$E = E_{q(z|x)} \log p(x|z) \tag{3}$$

$$R = KL(q(z|x)||p(z)) \tag{4}$$

where E is the log-likelihood term, and R is a Kullback-Leibler (KL) regularization term.

KL Vanishing Issues The ELBO can be viewed as a regularized version of the autoencoder (AE) [7]. It is thus natural to extend the negative ELBO in (2) by introducing a hyper-parameter $\beta \geq 0$ to control the strength of regularization [10]: $\mathcal{L} = E \quad \beta R$. When $\beta = 1$ (constant) during the training procedure, R quickly becomes vanishingly small during training [1]. [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; the decoder ignores z, and the VAE model reduces to a simpler language model. To address this issue, a monotonic annealing schedule was proposed to increase β from 0 to 1 during the early stage of VAE learning [1]. The cyclical annealing schedule further proposed to repeat this annealing process multiple time [5] during training.

2 Semi-Implicit Text Representations

2.1 Diagnostics of KL Vanishing

It has been shown recently that the auto-regressive decoder in the VAE has two paths working together to generate text sequences, and the KL vanishing issue happens due to the destructive competition between the two paths for the information flow [5], [illustrated in Figure 1(a). Path A consists of $\{a\ ;\ g\}$ passing through z. It first extracts the global representation z, then generates x directly. Path B employs the partial ground-truth information $x_{<t}$ of x at every time step of the sequential decoding, and it generates x_t conditioned on $x_{<t}$. Hence, both paths compete to generate the observed text; the KL vanishing issue appears when Path B dominates Path A.

However, it is unknown why Path A tends to be weak in this competition. In this paper, we hypothesize that there is a bottleneck on Path A that may weaken the information flow from x to x_t. Typically, $q(z|x)$ is modeled as a fully factorized Gaussian distribution:

$$q(z|x) = \mathcal{N}(\ ; \text{diag}(\ ^2)); \quad ; \quad ^2 = f\ (x) \tag{5}$$

where f is a $$-parameterized inference network, which outputs the mean and variance ^{2} of the Gaussian distribution.

The inference network aims to find the variational parameters for all sentences in the space of a Gaussian distribution. The performance is determined by two factors: (i) the capacity of the Gaussian form to match the true posterior; and (ii) the ability of the inference network to produce good variational parameters for each sentence. 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 con guration such as the prior, i.e., KL vanishing. It is thus promising to relax the Gaussian approximation to more expressive forms.

2.2 From GMM to Semi-Implicit Posteriors

We seek to extend the Gaussian posterior (5) to more exible distribution via Gaussian Mixture Models (GMM) [8]:

$$q_\phi(z|x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mu_k, \mathrm{diag}(\sigma_k^2)), \quad \text{with } \{\pi_k, \mu_k, \sigma_k^2\}_{k=1}^{K} = f_\phi(x), \tag{6}$$

where K is the number of Gaussian components, and μ_k, σ_k^2 is the mean and variance of the k-th component, and $\pi = [\pi_1, \ldots, \pi_K]$ is the mixture coef cient. The traditional GMM is restrictive in that (i) K needs to be pre-de ned, and (ii) the size of network ϕ for amortized inference will increase as K increases.

To bypass these issues, we propose the following semi-implicit (SI) form of GMM [26]:

$$q_\phi(z|x) = \int q(z|\psi) q_\phi(\psi|x) \tag{7}$$

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

$$\psi = f_\phi(x, \epsilon), \text{ with } \epsilon \sim q_0(\epsilon) \tag{8}$$

where $q_0(\epsilon)$ is an easy-to-sample distribution, f_ϕ is a ϕ-parameterized network with input x and ϵ.

Note that (7) can be viewed as a GMM with an in nite number of components: once ϕ is well trained, one can draw an arbitrary number of samples of ψ via (8), each of which corresponds to a new Gaussian component. In contrast to (6), this sampling process only requires adding a small constant number of network parameters in ϕ.

We show the probabilistic graphical model of SI in Figure 1(b). Path A is now composed of $\{x, \epsilon\} \to \psi \to z \to x_t$, where $q_\phi(z|x)$ has a much more exible distribution family to choose from than a Gaussian form. It allows z to encode x more easily. The enriched z can strength path A, and help it stay informative in the competition.

2.3 Learning and Inference with SI

Though exible, the SI representation (7) introduces additional dif culty in learning: the evaluation of the KL term in (2) becomes inef cient because its closed form disappears. Fortunately, there is a upper bound for the KL term:

$$\begin{aligned} R &= \mathrm{KL}(q_\phi(z|x) \| p(z)) \qquad (9) \\ &= \mathrm{KL}(\mathbb{E}_{q_\phi(\psi|x)} q(z|\psi) \| p(z)) \\ &\le \mathbb{E}_{q_\phi(\psi|x)} \mathrm{KL}(q(z|\psi) \| p(z)) \triangleq \underline{R}. \qquad (10) \end{aligned}$$

Note that (10) is ef cient to evaluate due to the Gaussian form of $q(z|\psi)$. Therefore, we can maximize the lower bound of the ELBO:

$$\mathcal{L} = \mathbb{E}[R] - \mathbb{E}[R] \le \mathbb{E}[R] - \underline{R} \triangleq \underline{\mathcal{L}} \tag{11}$$

However, directly optimizing (11) causes a degeneracy issue, characterized by $q_\phi(\psi|x)$ converging to a point mass density, making $q_\phi(z|x)$ reduce to the vanilla Gaussian form. To prevent degeneracy, a repulsive term is added to regularize [26]:

$$\begin{aligned} B_K &= \mathbb{E}_{\epsilon, \psi^{(1)}, \ldots, \psi^{(K)} \sim q_\phi(\psi|x)} \mathrm{KL}(q(z|\psi) \| q_K(z|\psi)) \\ \text{where } q_K(z|\psi) &\triangleq \frac{q(z|\psi) + \sum_{k=1}^{K} q(z|\psi^{(k)})}{K+1} \end{aligned} \tag{12}$$

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

Figure 2: Comparison of VAE (top row) and SI (bottom row) in the learned latent spaces for the three schedules.

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

Table 1: Performance comparison on the SwitchBoard dataset (P: precision, R: recall, L: average length). Blue numbers indicate SI performs than VAE. Bold means the highest compared with all other methods.

This leads to $\underline{L}_K \;,\; \underline{L} + B_K$. Note that maximizing $\underline{L}_K$ with $K \quad 1$ would encourage positive B_K and drive $q(\;|x)$ away from degeneracy. In other words, B_K guarantees the SI in (7) to spread out as an in nite number of Gaussian mixtures, rather than collapsing into one single Gaussian.

In the Appendix A, we describe how SI mitigates the KL issue, with various training techniques, including schedules (Monotonic and Cyclical), as well as Aggressive encoder training [9]. We also discuss related work in the Appendix B.

3 Experiments

We implement SI in Pytorch. We provide a detailed description of the settings, datasets, evaluation metrics in Appendix.

3.1 Visualization of Latent Spaces

To illustrate the learned of SI and VAE, in Figure 2 we visualize the latent space of a synthetic dataset of 10 sequences, where each color corresponds to $q(z|n)$, for $n = 1; \quad ;10$. The Gaussian VAE results are in the top row. The constant schedule produces heavily mixed latent codes for 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. This shows that the Gaussian VAE results heavily depend on the scheduling schemes. This is perhaps because the learning process is conducted in the restrictive space of Gaussian variational forms, and hence 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 $q(z|n)$ have non-Gaussian forms such as slice-shaped distributions.

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

Table 2: Results on PTB. The SA-VAE results are from [15]. Blue numbers indicate SI performs than VAE. Bold means the highest compared with all other methods.

3.2 Language Modeling

The results on Penn Tree Bank (PTB) dataset [5] are reported in Table 2. We first compare SI with the VAE trained with a monotonic and cyclical schedule. We see that SI achieves higher AU, MI, and KL than VAE for both schedules. This means SI can provide more informative latent codes than VAE. We compare with the semi-amortized (SA) training [10]. SI outperforms SA in terms of KL while maintaining the same PPL. This implies that learning with more flexible distribution forms yields more informative latent codes. When training VAE and SI with the aggressive encoder training, both methods are 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 reach its representational power.

3.3 Dialogue Response Generation

Table 1 summarizes the results for the various methods. All baseline results are from [8]. We compare SI with Gaussian CVAE for both monotonic and cyclical 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. This also implies that SI is able to learn more informative z, and suffers 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, showing evidence that SI can produce the most diverse response generation.

Table 4 in Appendix 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.

4 Conclusion

We have introduced a semi-implicit approximation 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 alleviates 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.

References

[1] Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. *ACL*, 2016.

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

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

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

[5] 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.

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

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

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

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

[10] 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.

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

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

[13] Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. ICLR, 2013.

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

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

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

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

[18] Tomáš Mikolov, Martin Kara át, Lukáš Burget, Jan Cernocky, and Sanjeev Khudanpur. Recurrent neural network based language model. Eleventh Annual Conference of the International Speech Communication Association, 2010.

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

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

[21] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. ICML, 2014.

[22] 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.

[23] 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 Arti cial Intelligence, 2017.

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

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

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

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

Table 3: Comparisons of vanilla VAE and SI.

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

A Mitigating KL Vanishing with SI

The KL vanishing issue happens due to a lack of good latent codes in the initial stages of decoder training [9, 5], and the decoder passively chooses to use the previous word token as clue to generate the next word x_t. Based on this observation, two algorithms are proposed independently:

> Aggressive encoder training He et al.[9] propose to aggressively optimize the encoder T_a times ($T_a > 0$) before performing each model update.
> Cyclical β-annealing schedule Fu et al.[5] propose simply repeats the monotonic 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 rest $1 - R$ for fixing $\beta = 1$. A linear annealing scheme is:
>
> $$\beta_t = \begin{cases} \tau/R, & \tau \leq R \\ 1, & \tau > R \end{cases} \quad \text{with} \quad \tau = \frac{\text{mod}(t-1, \lceil T/M \rceil)}{T/M}. \tag{13}$$

Both works share a similar motivation: improving the quality of 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 paper, 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:

$$\underline{\mathcal{L}}_K = \underline{\mathcal{L}} + \mathcal{B}_K = \mathcal{E} + \underline{\mathcal{R}} + \mathcal{B}_K \tag{14}$$

$$= \mathbb{E}_{z \sim q_\phi(z|x)}[\log p_\theta(x|z) + p(z) - q_{\mathcal{K}}(z|\cdot)]:$$

$$\frac{1}{J}\sum_{j=1}^{J}\Big[\log p_\theta(x|z_j) + \log p(z_j) - \log \frac{1}{K+1}\sum_{k=1}^{K}[q(z_j|\cdot_k) + q(z_j|\cdot_j)]\Big]; \tag{15}$$

where the expectation in (14) is estimated via J samples in (15), with the j-th sample drawn as:

$$z_j \sim q(z|\cdot_j), \quad \cdot_j \sim q_\phi(\cdot|x) \tag{16}$$

On the impact of K The representational power of SI relies on K, the number of samples used to approximate (7). 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 [26] that the regularized lower bound $\underline{\mathcal{L}}_K$ is an asymptotically exact ELBO that satisfies $\underline{\mathcal{L}}_0 = \underline{\mathcal{L}}$ and $\lim_{K\to\infty} \underline{\mathcal{L}}_K = \mathcal{L}$.

We summarize the connection/comparison of vanilla VAE and SI in Table 3. 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 [9] or the cyclical schedule [5] to get better performance, while maintaining low computational cost. Note the key to reduce KL vanishing is to feed high-quality z to decode in the initial stage of decode learning. Here, this initial stage means the time after each aggressive training loop ends, or in each cyclical period ends in [5]. 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 , decoder , K_t, T_a, M , R

Initialize: , ;

for t = 1; 2; : : : ; T do

  % annealing schedule

  Evaluate $_t$ using (13);

  % Aggressive encoder training

  for t_a = 1; 2; : : : ; T_a do

    Sample a minibatch X ;

    Compute gradient g = r $\underline{L}_{t;K_t}(X)$;

    Update using gradient g ;

  end

  % SI update

  Sample a minibatch X ;

  Sample $_k$ using (8) for k = 1; ; K_t ;

  Sample z_j using (16) for j = 1; ; J ;

  Compute gradient $g_{;}$ = r $_{;}$ $\underline{L}_{t;K_t}(X)$;

  Update , using gradient $g_{;}$;

end

B Related Work

Good latent representations are key for a wide range of NLP tasks. We recommend [12] 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 exible posteriors, including normalizing ows (NFs) [20], adversarial variational Bayes (AVB) [16], particle-based methods [4]. Our method is based on one [26]. 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 dif culties in learning representations. Our paper presents the rst work to investigate the exible 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 exible posteriors such as SI.

C Visualization

Each sequence 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 -scheduling schemes are considered [5].

We refer $q(z) = \sum_{n=1}^{N} q(z|n)q(n)$ as the aggregated posterior [14]. This marginal distribution characterizes the aggregated z after embedding the entire dataset into the latent space. Good latent representations should have low $KL(q(z)||p(z))$. Ideally, it means the overall shape of z samples is close to $p(z) = N(0; 1)$. The exibility of SI allows the learned posteriors to adapt arbitrarily well to t the prior, thus produces high-quality $q(z)$. In contrast, the limitation of Gaussian assumptions in vanilla VAE leads to some holes between $q(z|n)$ and $p(z)$, making $q(z)$ violate the constraint of $p(z)$. The SI representations capture more clear patterns and structured information is captured in z, which is bene cial in downstream applications below.

D Language Modeling

Dataset and Setup We rst consider applying VAEs to the language modeling task on the Penn Tree Bank (PTB) dataset [15]. Following [11, 9], we set the word embedding dimension as 512. The encoder and decoder are LSTMs with 1024 hidden units, respectively. , K is linearly increased to K_{max} = 50.