Better Exploiting Latent Variables in Text Modeling

Canasai Kruengkrai
Yahoo Japan Corporation
`ckruengk@yahoo-corp.jp`

Abstract

We show that sampling latent variables multiple times at a gradient step helps in improving a variational autoencoder and propose a simple and effective method to better exploit these latent variables through hidden state averaging. Consistent gains in performance on two different datasets, Penn Treebank and Yahoo, indicate the generalizability of our method.[1]

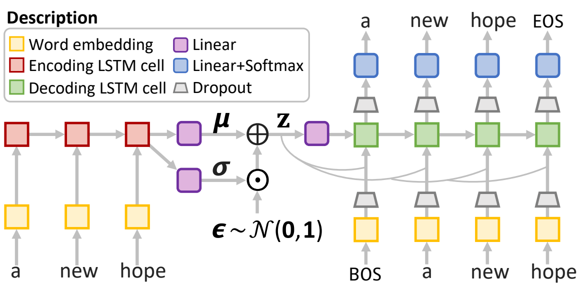

Figure 1: Baseline LSTM-VAE model.

1 Introduction

Introducing latent variables to neural language models would help in generating plausible sentences that reflect sentential semantics (Bowman et al., 2016). The success of learning latent variables is also beneficial to various natural language processing (NLP) tasks such as sentence compression (Miao and Blunsom, 2016) and text style transfer (Shen et al., 2017). One of the widely-used latent variable models is the variational autoencoder (VAE) (Kingma and Welling, 2014; Rezende et al., 2014). When applying the VAE to text data, recurrent neural networks are typically utilized for both the encoder and the decoder. Training the VAE with a high-capacity decoder such as a long short-term memory (LSTM) network (Hochreiter and Schmidhuber, 1997) can be challenging. The LSTM is powerful enough to model the underlying data distribution without the use of latent variables.

In this paper, we take a closer look at one of the components in an LSTM-VAE model, namely the latent variable sampling scheme. Fig. 1 illustrates our baseline LSTM-VAE model built upon Bowman et al. (2016)'s model. At each gradient step (i.e., a minibatch run), most previous work pairs an input sentence with a single latent variable denoted by $\mathbf{z}$. This would be sufficient in some tasks but not necessarily effective in text modeling. At the beginning of training, the latent variable $\mathbf{z}$ contains a small amount of information about the input sentence. Many latent units of $\mathbf{z}$ are pulled towards the prior early to optimize an objective function before they capture useful information (Hoffman et al., 2013; Sønderby et al., 2016). Without a cost annealing strategy or a constraint on the decoder (Bowman et al., 2016; Chen et al., 2017; Yang et al., 2017), $\mathbf{z}$ would be entirely ignored for the remaining training steps. In our work, we aim at developing a simple variant of the LSTM-VAE model to address this common training issue. We observe that pairing the input sentence with multiple latent variables improves latent variable usage. In addition, we present a method that leverages multiple latent variables to further boost the performance of the baseline LSTM-VAE model.

Our contributions are as follows: We suggest sampling the latent variables multiple times at each gradient step. We propose a simple method to better exploit these latent variables through hidden state averaging. We evaluate the proposed method on two different datasets, Penn Treebank and Yahoo, and compare to the best results published in the literature. Our empirical results show that our method can effectively make use of the latent variables, leading to the state-of-the-art performance.

[1]The code for reproducibility is available at `https://research-lab.yahoo.co.jp/en/software`.

2 Related work

Bowman et al. (2016) first proposed an LSTM-VAE model for text. They observed the posterior-collapse problem in which the approximate posterior collapses to the prior, and the model ignores the latent variable. They suggested two techniques to alleviate this issue: cost annealing (called *warm-up* in (Sønderby et al., 2016)) and word dropout. Weakening the decoder with word dropout forces the latent variable to encode more information, but their LSTM-VAE model still underperforms against the standard LSTM language model. Yang et al. (2017) proposed to replace the LSTM decoder with a dilated convolutional neural network (CNN) (van den Oord et al., 2016) to control the contextual capacity. However, their positive results also came from initializing the encoder with a pre-trained LSTM language model. Guu et al. (2018) first proposed using the von Mises–Fisher (vMF) distribution to model the VAE instead of using the Gaussian distribution. However, the vMF distribution presupposes that all data are directional unit vectors. Other applications of the vMF distribution can be found in (Davidson et al., 2018; Xu and Durrett, 2018). Kim et al. (2018) presented a semi-amortized (SA) approach to training the VAE, while He et al. (2019) proposed an aggressive inference network training. However, their training algorithms are computationally expensive since they require backpropagating through the decoder or the encoder multiple times. Our method is simpler and easy to implement. In practice, we just place a loop before reparameterization and do averaging.

3 Background

Let $\mathbf{x} = [w_1, w_2, \dots, w_T]$ be a sentence representation, where w_t is the t-th word. Assume that $\mathbf{x}$ is generated from a continuous latent variable $\mathbf{z}$ using a random process $\mathbf{x} \sim p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ parameterized by $\boldsymbol{\theta}$. By applying the standard language model (Bengio et al., 2003), we get:

$$p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) = \prod_{t=1}^{T} p_{\boldsymbol{\theta}}(w_t|w_{1:t-1}, \mathbf{z}). \tag{1}$$

Given a dataset $\mathbf{X} = \{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(N)}\}$, we typically fit the model by maximizing the average log-marginal likelihood $\frac{1}{N}\sum_1^N \log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)})$. We can express an individual log-marginal likelihood by $\log p_{\boldsymbol{\theta}}(\mathbf{x}) = \log \int_{\mathbf{z}} p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})p(\mathbf{z})\mathrm{d}\mathbf{z}$, where $p(\mathbf{z})$ is the prior on $\mathbf{z}$. Unfortunately, the integral over $\mathbf{z}$ is intractable (Hoffman et al., 2013). Alternatively, we would sample $\mathbf{z}$ directly from the posterior distribution $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$. However, $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ is also intractable since $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) = p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})p(\mathbf{z})/p_{\boldsymbol{\theta}}(\mathbf{x})$.

Variational inference approximates the posterior distribution $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ with a variational family of distributions $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ parameterized by $\boldsymbol{\phi}$. We wish that $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ is close to $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$. We measure this closeness by the Kullback–Leibler (KL) divergence: $\mathrm{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})||p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}))$. Instead of maximizing the true log-marginal likelihood, we maximize its lower bound:

$$\begin{aligned}\log p_{\boldsymbol{\theta}}(\mathbf{x}) &\geq \log p_{\boldsymbol{\theta}}(\mathbf{x}) - \mathrm{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})||p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})) \\ &= \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})] \\ &\qquad - \mathrm{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})||p(\mathbf{z})).\end{aligned} \tag{2}$$

The above equation is typically referred to as the evidence lower bound (ELBO) (Hoffman and Johnson, 2016). The ELBO consists of two terms: the expected reconstruction term and the KL-divergence term. We can solve the KL-divergence term analytically given that both the prior $p(\mathbf{z})$ and the variational posterior $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ are Gaussian (see Kingma and Welling (2014)'s Appendix B). We then need to rewrite the expected reconstruction term into some closed-form expression (detailed in §4) so that we can maximize it by applying stochastic optimization methods.

Optimizing the ELBO forms the VAE architecture in which $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ encodes $\mathbf{x}$ into a latent variable $\mathbf{z}$, and $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ decodes $\mathbf{z}$ to reconstruct $\mathbf{x}$. The gradient of the ELBO w.r.t. $\boldsymbol{\phi}$ can have low variance by applying the reparameterization trick (Kingma and Welling, 2014) that estimates $\mathbf{z} \sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ using $\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}$, where mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\sigma}^2$ are outputs of some neural networks, and $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$.

4 Proposed method

Having covered the technical background, we now describe our two extensions to improve the baseline LSTM-VAE model in Fig. 1. The baseline model approximates the expected reconstruction term by sampling one latent variable $\mathbf{z} \sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ at each gradient step (Bowman et al., 2016). Thus, $\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})] \approx \log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$.

Our first extension is to improve the sampling by using a Monte Carlo estimate of the expected

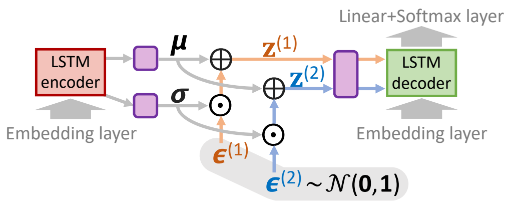

Figure 2: Example of sampling two latent variables using different random noise vectors drawn from the standard Gaussian distribution.

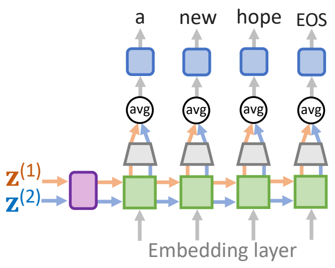

Figure 3: Example of averaging two hidden states at each decoding step.

reconstruction term (Kingma and Welling, 2014):

$$\mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})] \approx \frac{1}{L}\sum_{l=1}^{L}\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}^{(l)}), \quad (3)$$

where $\mathbf{z}^{(l)} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}^{(l)}$ and $\boldsymbol{\epsilon}^{(l)} \sim \mathcal{N}(\mathbf{0},\mathbf{1})$. Sampling latent variables multiple times at each gradient step should result in a better approximation of the expected reconstruction term. Fig. 2 shows an example of sampling two latent variables. Note that we use the same $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ for both latent variables. By using the language model from Eq. (1), we can decompose the reconstruction term as:

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}^{(l)}) = \sum_{t=1}^{T}\log p_{\boldsymbol{\theta}}(w_t|w_{1:t-1},\mathbf{z}^{(l)}). \quad (4)$$

Let $\mathcal{V}$ be a fixed size vocabulary of words in a dataset. Given the entire history of previous words $w_{1:t} = [w_1, \dots, w_t]$ and the latent variable $\mathbf{z}^{(l)}$, we compute the distribution over the possible corresponding values of w_{t+1} by applying a linear transformation to the decoder hidden state followed by a softmax:

$$\begin{aligned} p_{\boldsymbol{\theta}}(w_{t+1}|w_{1:t},\mathbf{z}^{(l)}) &= \mathrm{softmax}(\mathbf{h}_t^{(l)}\mathbf{M}_1), \\ \mathbf{h}_t^{(l)} &= \mathrm{dec}(\mathbf{h}_{t-1}^{(l)},\mathbf{w}_t), \\ \mathbf{h}_0^{(l)} &= \mathbf{M}_2\mathbf{z}^{(l)}, \end{aligned} \quad (5)$$

where $\mathbf{M}_1 \in \mathbb{R}^{m\times|\mathcal{V}|}$ and $\mathbf{M}_2 \in \mathbb{R}^{m\times n}$ are the trainable weight matrices, $\mathbf{h}_t^{(l)} \in \mathbb{R}^m$ is the decoder hidden state, $\mathbf{z}^{(l)} \in \mathbb{R}^n$ is the latent variable at each sampling step l, and $\mathbf{w}_t \in \mathbb{R}^d$ is the embedding vector of the word w_t. We compute $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}^2$ used in the reparameterization trick by:

$$\begin{aligned} \boldsymbol{\mu} &= \mathbf{M}_3\mathbf{s}_T, \\ \log\boldsymbol{\sigma}^2 &= \mathbf{M}_4\mathbf{s}_T, \\ \mathbf{s}_t &= \mathrm{enc}(\mathbf{s}_{t-1},\mathbf{w}_t), t = 1,\dots,T \\ \mathbf{s}_0 &= \mathbf{0}, \end{aligned} \quad (6)$$

where $\mathbf{M}_3, \mathbf{M}_4 \in \mathbb{R}^{n\times m}$ are the trainable weight matrices and $\mathbf{s}_T \in \mathbb{R}^m$ is the last encoder hidden state.

Our second extension is to exploit multiple latent variables to directly improve the expressiveness of the decoder. Instead of computing the separate reconstruction terms and taking the average of them as in Eq. (3), we combine the decoder hidden states at each time step t:

$$\tilde{\mathbf{h}}_t = \frac{1}{L}\sum_{l=1}^{L}\mathbf{h}_t^{(l)}, \quad (7)$$

where each hidden state is initialized with a different latent variable $\mathbf{z}^{(l)}$. Fig. 3 shows an example of averaging two hidden states at each decoding step. Thus our distribution of w_{t+1} becomes:

$$p_{\boldsymbol{\theta}}(w_{t+1}|w_{1:t},\mathbf{z}) = \mathrm{softmax}(\tilde{\mathbf{h}}_t\mathbf{M}_1). \quad (8)$$

Here we drop the superscript (l) since all hidden states $\mathbf{h}_t^{(l)}$ are averaged into $\tilde{\mathbf{h}}_t$.

5 Experiments

5.1 Datasets and training details

We experiment on two datasets: Penn Treebank (PTB) (Marcus et al., 1993) and Yahoo (Zhang et al., 2015). Training/validation/test sets are identical to (Bowman et al., 2016; Xu and Durrett, 2018) for PTB and (Yang et al., 2017; Kim et al., 2018) for Yahoo. We use single-layer unidirectional LSTMs as an encoder and a decoder. Configurations of our baseline model (**LSTM-VAE**, Fig. 1) are identical to (Xu and Durrett, 2018) for PTB and (Yang et al., 2017; Kim et al., 2018) for Yahoo. When the LSTM encoder is not applied, our model falls back to a vanilla language model (**LSTM-LM**). Table 1 summarizes data statistics and our model configurations.

We use the last hidden state (not the cell state) of the LSTM encoder and feed it through linear transformations to get the mean $\boldsymbol{\mu}$ and the variance $\boldsymbol{\sigma}^2$.

	PTB	Yahoo
Training	42068	100000
Validation	3370	10000
Test	3761	10000
$\lvert\mathcal{V}\rvert$	10000	20000
d	100	512
m	400	1024
n	32	32

Table 1: Data statistics and model configurations. $|\mathcal{V}|$ = vocabulary size; d = dimensionality of word embeddings; m = number of LSTM hidden units; n = dimensionality of latent variables.

We sample $\mathbf{z}$ using the reparameterization trick and feed it through a linear transformation to get the initial hidden state of the LSTM decoder while setting the initial cell state to zero. We concatenate $\mathbf{z}$ with the word embedding at each decoding step. We use dropout (Hinton et al., 2012) with probability 0.5 on the input-to-hidden layers and the hidden-to-softmax layers.

We initialize all model parameters and word embeddings by sampling from $\mathcal{U}(-0.1, 0.1)$. We train all models using stochastic gradient descent (SGD) with the batch size of 32, the learning rate of 1.0, and the gradient clipping at 5. The learning rate decays by halves if the validation perplexity does not improve. We train for 30 epochs or until the validation perplexity has not improved for 3 times. All models are trained on NVIDIA Tesla P40 GPUs.

Following previous work (Bowman et al., 2016; Sønderby et al., 2016), we apply KL cost annealing to all **LSTM-VAE** models. The multiplier on the KL term is increased linearly from 0 to 1 during the first 10 epochs of training.

We also try word dropout (Bowman et al., 2016) during development but find that it is not effective when combined with standard dropout. Our finding conforms to (Kim et al., 2018). So we do not apply word dropout to our models.

5.2 Main results

We report the upper bounds (i.e., the negative ELBO in Eq. (2)) on NLL/PPL. We vary the number of latent variables L in the variational models to assess their impact on performance. **LSTM-VAE-AVG** indicates the averaging of hidden states at each decoding step in Eq. (8). We also report the results of the inputless setting (Bowman et al., 2016), which corresponds to dropping all ground truth words during decoding.

Table 2 shows the results of various models. The **LSTM-VAE-AVG** models with multiple latent variables provide the best improvements in terms of NLL/PPL. The **LSTM-VAE** models trained with more latent variables offer slight improvements over the baseline version (i.e., using one latent variable) for the standard setting.

The baseline **LSTM-VAE** models have low KL values and underperform against **LSTM-LM** for the standard setting. Incorporating multiple latent variables consistently helps in increasing the KL values. Note that a high KL term does not necessarily imply a better upper bound. Generally, we do not expect the KL term to approach zero. When $\mathrm{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) = 0$, it indicates that $\mathbf{z}$ and $\mathbf{x}$ are independent (i.e., $q_\phi(\mathbf{z}|\mathbf{x}) = q_\phi(\mathbf{z}) = p(\mathbf{z})$). In other words, $\mathbf{z}$ learns nothing from $\mathbf{x}$.

The **LSTM-VAE-AVG** models have relatively high KL values (except the inputless setting on Yahoo), while still maintaining better upper bounds on NLL/PPL. These results suggest that our models with expressive decoders can effectively make use of the latent variables.

5.3 Discussion

On PTB, **LSTM-VAE-AVG** ($L = 10$) achieves the best results compared to previous work (Bowman et al., 2016; Xu and Durrett, 2018). On Yahoo, **LSTM-VAE-AVG** ($L = 5$) slightly outperforms Kim et al. (2018)'s SA-VAE. Our model can provide similar improvements while being simpler. We also observe that our vanilla **LSTM-LM** model and that of Kim et al. (2018) have better results than Yang et al. (2017)'s models. One plausible explanation is that Yang et al. (2017) trained their models with Adam (Kingma and Ba, 2015), while we used SGD. For text modeling, researchers have shown that SGD performs better than other adaptive optimization methods such as Adam (Wilson et al., 2017; Keskar and Socher, 2017).

The ELBO has been commonly used to evaluate the variational models (Bowman et al., 2016; Yang et al., 2017; Xu and Durrett, 2018; Kim et al., 2018). There also exists a line of work that uses importance sampling to estimate the true log-marginal likelihood (Rezende et al., 2014; Burda et al., 2016; Tomczak and Welling, 2018; He et al., 2019). We further conduct experiments by computing the importance sampling estimates with 500 samples and comparing to He et al. (2019)'s

Model		Standard			Inputless		
		NLL	KL	PPL	NLL	KL	PPL
Bowman et al. (2016)	LSTM-LM	100	–	116	135	–	600
	LSTM-VAE	101	2	119	125	15	380
Xu and Durrett (2018)	LSTM-LM	100	–	114	134	–	596
	LSTM-VAE	99	4.4	109	125	6.3	379
	LSTM-vMF-VAE	96	5.7	98	117	18.6	262
This work	LSTM-LM	100.8±0.2	–	99.4±0.7	139.9±0.0	–	592.3±0.5
	LSTM-VAE	102.5±0.2	1.5±0.3	107.5±1.0	134.8±0.4	3.8±0.5	469.3±7.6
	LSTM-VAE ($L = 5$)	100.7±0.3	2.1±0.4	98.8±1.2	134.7±0.9	3.9±0.9	468.1±19.4
	LSTM-VAE ($L = 10$)	100.4±0.2	2.2±0.4	97.7±0.9	134.8±0.8	3.5±1.0	468.2±16.2
	LSTM-VAE-AVG ($L = 5$)	97.3±0.6	7.6±0.9	84.6±2.4	118.8±0.5	10.6±0.3	225.8±5.6
	LSTM-VAE-AVG ($L = 10$)	**94.3**±0.4	8.1±0.2	**73.8**±1.5	**113.8**±1.0	9.6±0.5	**179.7**±8.3

(a) PTB

Model		Standard			Inputless		
		NLL	KL	PPL	NLL	KL	PPL
Yang et al. (2017)	CNN-LM	335.4	–	66.6	–	–	–
	CNN-VAE + init	332.1	10.0	63.9	–	–	–
Kim et al. (2018)	LSTM-LM	329.1	–	61.6	–	–	–
	SA-VAE	327.5	7.2	60.4	–	–	–
This work	LSTM-LM	328.4±0.2	–	61.1±0.2	507.4±0.0	–	574.0±0.0
	LSTM-VAE	330.4±0.4	1.5±0.5	62.6±0.3	467.5±0.3	18.5±0.4	348.5±1.5
	LSTM-VAE ($L = 5$)	328.8±0.1	2.6±0.5	61.4±0.1	464.3±1.2	22.2±1.7	334.8±4.9
	LSTM-VAE ($L = 10$)	329.1±0.1	2.8±0.7	61.6±0.1	464.3±1.3	22.8±1.7	334.8±5.5
	LSTM-VAE-AVG ($L = 5$)	**327.3**±0.5	12.2±0.4	**60.3**±0.4	446.4±0.1	19.7±0.2	267.5±0.4
	LSTM-VAE-AVG ($L = 10$)	328.5±1.3	10.8±1.0	61.2±1.0	**441.4**±0.5	16.8±0.2	**251.2**±1.5

(b) Yahoo

Table 2: Results on (a) PTB and (b) Yahoo test sets. For LSTM-LM, we show the exact negative log likelihood (NLL) and perplexity (PPL). For the variational models, we show the upper bounds (i.e., the negative ELBO) on NLL/PPL. The KL portion of the ELBO is given in the column alongside NLL. NLL/KL values are averaged across examples. L indicates the number of latent variables at each gradient step. We report mean and standard deviation computed across five training/test runs from different random initial starting points.

		PTB		Yahoo	
		$\text{NLL}_{\text{-ELBO}}$	NLL_{IW}	$\text{NLL}_{\text{-ELBO}}$	NLL_{IW}
He et al. (2019)	LSTM-VAE-AIN + anneal	–	–	328.4±0.2	326.7±0.1
This work	LSTM-VAE	102.5±0.2	102.1±0.2	330.4±0.4	329.6±0.2
	LSTM-VAE-AVG ($L = 5$)	97.3±0.6	95.1±0.8	327.3±0.5	**324.0**±0.5
	LSTM-VAE-AVG ($L = 10$)	94.3±0.4	**91.7**±0.5	328.5±1.3	324.9±1.3

Table 3: Comparison of different NLL estimates on PTB and Yahoo test sets. $\text{NLL}_{\text{-ELBO}}$ = the upper bounds taken from Table 2; NLL_{IW} = the importance sampling estimates of NLL with 500 samples. We report mean and standard deviation computed across five training/test runs from different random initial starting points.

aggressive inference network (AIN) training. Table 3 shows a comparison of different NLL estimates. Our results are consistent with those of (He et al., 2019) in which the importance sampling yields the tighter bounds than the ELBO.

6 Conclusion

We have shown that using multiple latent variables at each gradient step can improve the performance of the baseline LSTM-VAE model. The empirical results indicate that our models combined with expressive decoders can successfully make use of the latent variables, resulting in higher KL values and better NLL/PPL results. Our proposed method is simple and can serve as a strong baseline for latent variable text modeling.

Acknowledgments

We thank the anonymous reviewers for their insightful comments.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Józefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *Proceedings of CoNLL*.

Yuri Burda, Roger B. Grosse, and Ruslan Salakhutdinov. 2016. Importance weighted autoencoders. In *Proceedings of ICLR*.

Xi Chen, Diederik P. Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2017. Variational lossy autoencoder. In *Proceedings of ICLR*.

Tim R. Davidson, Luca Falorsi, Nicola De Cao, Thomas Kipf, and Jakub M. Tomczak. 2018. Hyperspherical variational auto-encoders. In *Proceedings of UAI*.

Kelvin Guu, Tatsunori B. Hashimoto, Yonatan Oren, and Percy Liang. 2018. Generating sentences by editing prototypes. *Transactions of the Association for Computational Linguistics (TACL)*.

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

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8).

Matthew D. Hoffman, David M. Blei, Chong Wang, and John W. Paisley. 2013. Stochastic variational inference. *Journal of Machine Learning Research*, 14(1).

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

Nitish Shirish Keskar and Richard Socher. 2017. Improving generalization performance by switching from adam to SGD. *CoRR*, abs/1712.07628.

Yoon Kim, Sam Wiseman, Andrew Miller, David Sontag, and Alexander Rush. 2018. Semi-amortized variational autoencoders. In *Proceedings of ICML*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR*.

Diederik P. Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *Proceedings of ICLR*.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Comput. Linguist.*, 19(2).

Yishu Miao and Phil Blunsom. 2016. Language as a latent variable: Discrete generative models for sentence compression. In *Proceedings of EMNLP*.

Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. 2016. Wavenet: A generative model for raw audio. *CoRR*, abs/1609.03499.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of ICML*.

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

Casper Kaae Sønderby, Tapani Raiko, Lars Maalø e, Søren Kaae Sø nderby, and Ole Winther. 2016. Ladder variational autoencoders. In *Proceedings of NIPS*.

Jakub M. Tomczak and Max Welling. 2018. VAE with a VampPrior. In *Proceedings of AISTATS*.

Ashia C. Wilson, Rebecca Roelofs, Mitchell Stern, Nati Srebro, and Benjamin Recht. 2017. The marginal value of adaptive gradient methods in machine learning. In *Proceedings of NIPS*.

Jiacheng Xu and Greg Durrett. 2018. Spherical latent spaces for stable variational autoencoders. In *Proceedings of EMNLP*.

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

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proceedings of NIPS*.