Implementation of Trigram Kneser-Ney Language Modeling

Qing Wei
A53309131
University of California San Diego
Computer Science and Engineering
qiwei@eng.ucsd.edu

Abstract

This report includes a brief introduction of both RNN, its variants, and VAE language models. In the report, we will show the validity of ELBO as the lower bound of log likelihood, as well as some empirical investigation of how latent variables and recurrent neural networks can improve language modelling performance.

1 Introduction

N-gram language models make a Markov assumption and assumes that data is distributed evenly, which applies a rather strong restriction on the language model. With the following described models, we are able to build language models that can predict the next word based on the complete history of the sentence, and that can handle heterogeneous data.

2 Recurrent Neural Network Language Model

With no assumption to the dataset, we apply the chain rules to calculate the probability of data $\mathbf{x} = \{x_1, x_2, \dots, x_n\}$:

$$p(x_1, \dots, x_n) = \prod_i p(x_i | x_1, \dots, x_{i-1})$$

In each step of calculation of RNN language models, instead of taking all the prior words in a sentence as the input, only takes a output of the last word inference h_{i-1} and the current word x_i. It uses the same parameters for all positions of the sentence, and generates the new prediction $h_i = \sigma(W^T h_{i-1} + U^T x_i)$. In other words, RNN treats h_t as a feature representation of the history up to word x_t. Each neuron of RNN is a logistic regression classifier to predict the conditional probability of the next word $p(x_{i+1}|x_1, \dots, x_i)$.

A RNN language model has many advantages over the feed-forward deep language models. It can process any length of inputs, the computation could use information from any steps back as well, and model size doesn't increase for longer input.

2.1 Variations

In theory, RNN can keep track of any information from earlier steps. However, in practice, the back propagation process of RNN often suffers from the vanishing gradient or exploding gradient problems. Vanishing gradient problem is often observed in earlier layers. When the gradient value becomes extremely close to zero, the updates on corresponding parameters would be small enough to be ignored. In consequence, the RNN will only has a short-term memory. To solve this problem, many variants of RNN have been, such as Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU).

LSTM's and GRU's were created as the solution to short-term memory (Hochreiter and Schmidhuber, 1997; Cho et al., 2014). They have internal mechanisms called gates that can regulate the flow of information.

2.1.1 Long Short-Term Memory

Unlike standard feedforward neural networks, LSTM has feedback connections. Unlike basic RNN neuron modules which only concatenate h_{t-1} and x_t, and apply a sigmoid function to get the output, A LSTM unit is composed of a forget gate, an input gate, an output gate and the cell state.

The forget gate decides what information to be kept. It takes h_{t-1} and x_t as input, uses sigmoid as activation function, and gets a value between 0 and 1. The closer to 0 means to forget, and the closer to 1 means to keep.

The input gate applied a sigmoid function and a tanh function to h_{t-1} and x_t. The output of sigmoid function decides the importance of the values of the tanh output. The multiplication of the outputs of these two functions is the final output of the input gate.

The cell state c_t, in theory, can carry relevant information throughout the processing of the sequence. c_t can be calculated by first point-wise multiplying the cell state from the last unit c_{t-1} and the output of forget gate, then adding the output of input gate to the result.

The output gate outputs h_t in the current stage, which contains information on previous inputs and is used for predictions. h_t is a multiplication of $sigmoid([h_{t-1}, x_t])$ and $\tanh(c_t)$.

2.1.2 Gated Recurrent Units

Instead of keeping a cell state like LSTM, GRU chooses to still use the hidden state h_t to transfer information. It only has two gates, a reset gate and an update gate. The update gate has the same function of the forget and input gate of an LSTM, deciding what information to throw away and what new information to add. The reset gate also is used to decide how much past information to forget.

3 Variational Autoencoder

In order to handle heterogeneous data, a new variable $\mathbf{z}$ is introduced to capture clusters of dialect, style, sentiment in data. If we could identify the category $\mathbf{z}$ for the sentence, then we could generate $p_\theta(\mathbf{x}|\mathbf{z}) = \prod_t p_\theta(x_t|x_1, x_2, \ldots, x_{t-1}, \mathbf{z})$ instead with separate set of RNN parameters for each $\mathbf{z}$.

Variational autoencoders (VAEs), defined by Kingma and Welling (2013) and Rezende et al. (2014), is a combination of a deep latent variable model with a learning and inference technique known as amortized inference.

In VAEs, latent vector $\mathbf{z}$ is generated from a Gaussian prior; a decoder is used to estimate the distribution of the observation $\mathbf{x}$ given $\mathbf{z}$; amortized inference introduces an inference network RNN, which is sometimes referred to as the "encoder", to approximate the model's true posterior $p(\mathbf{z}|\mathbf{x})$ using another distribution $q(\mathbf{z}|\mathbf{x})$, in order to compute a lower bound on data marginal likelihood.

Ideally, log marginal likelihood is often used as the objective function to train the decoder of VAEs:

$$\log p_\theta(\mathbf{x}) = \log \int p_\theta(\mathbf{z}) p_\theta(\mathbf{x}|\mathbf{z}) d\mathbf{z}$$

However, for some decoders, this marginal probability is intractable. In this case, we have to find an approximation to it in order to maximize the log likelihood.

3.1 Evidence Lower Bound

In VAE, the evidence lower bound (ELBO) is used as a lower bound to estimate $\log p_\theta(x)$ in case that it is intractable. ELBO is defined as $\mathbb{E}_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x)||p(z))$. The below sections will prove the validity of ELBO as the lower bound of the marginal log likelihood.

3.1.1 Kullback–Leibler divergence Divergence

For distributions P and Q of a continuous random variable, the Kullback-Leibler divergence divergence is defined to be the integral:

$$D_{KL}(P||Q) = \int_{-\infty}^{\infty} p(x) \log\left(\frac{p(x)}{q(x)}\right) dx$$

An important property of KL divergence is that it is always non-negative:

$$D_{KL}(P||Q) \geq 0$$

with $D_{KL}(P||Q)$ zero if and only if $P = Q$.

Proof. With the inequality $\log x \leq x - 1$, we have $\log\left(\frac{q(x)}{p(x)}\right) \leq \frac{q(x)}{p(x)} - 1$. Therefore,

$$\begin{aligned}
&- D_{KL}(P||Q) = -\int_{-\infty}^{\infty} p(x) \log\left(\frac{p(x)}{q(x)}\right) dx \\
&= \int_{-\infty}^{\infty} p(x) \log\left(\frac{q(x)}{p(x)}\right) dx \\
&\leq \int_{-\infty}^{\infty} p(x) \left(\frac{q(x)}{p(x)} - 1\right) dx \\
&= \int_{-\infty}^{\infty} q(x) - p(x) dx \\
&= \int_{-\infty}^{\infty} q(x) dx - \int_{-\infty}^{\infty} p(x) dx \\
&= 1 - 1 \\
&= 0
\end{aligned}$$

$$\therefore D_{KL}(P||Q) \geq 0$$

□

3.1.2 Derivation of ELBO

Claim For any distribution $q(\mathbf{z})$, log marginal $\log p(\mathbf{x})$ is bounded by ELBO, the following inequality holds:

$$\begin{aligned}&\log p_\theta(x)\\ &\geq \mathop{\mathbb{E}}_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x)||p(z))\end{aligned}$$

Proof. First, we could expand all the terms in above inequality:

$$\begin{aligned}\log p_\theta(\mathbf{x}) &= \int_{-\infty}^{\infty} q_\phi(z|x)\log p_\theta(x)dz\\ \mathop{\mathbb{E}}_{q_\phi(z|x)}[\log p_\theta(x|z)] &= \int_{-\infty}^{\infty} q_\phi(z|x)\log p_\theta(x|z)dz\\ D_{KL}(q_\phi(z|x)||p(z)) &= \int_{-\infty}^{\infty} q_\phi(z|x)\log\frac{q_\phi(z|x)}{p(z)}dz\end{aligned}$$

Then subtract the right hand side terms from the left hand side:

$$\begin{aligned}&\log p_\theta(\mathbf{x}) -\\ &\left(\mathop{\mathbb{E}}_{q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x)||p(z))\right)\\ &= \int_{-\infty}^{\infty} q_\phi(z|x)\log\left(\frac{p_\theta(x)q_\phi(z|x)}{p_\theta(x|z)p(z)}\right)dz\\ &= \int_{-\infty}^{\infty} q_\phi(z|x)\log\left(\frac{q_\phi(z|x)}{p_\theta(z|x)}\right)dz\\ &= D_{KL}(q_\phi(z|x)||p_\theta(z|x))\\ &\geq 0\end{aligned}$$

□

Therefore, the above inequality holds, with equality if and only if $q_\phi(z|x) = p_\theta(z|x)$ for any $\mathbf{x}$ and $\mathbf{z}$.

Now given the ELBO as a effective lower bound of the loss function $\log p_\theta(\mathbf{x})$, we could derive a new objective function:

$$\max_{\theta,q(z|x)} \mathop{\mathbb{E}}_{q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x)||p(z))$$

3.2 Posterior Collapse

Posterior collapse is a phenomenon happened in VAE training when:

$$\begin{aligned}p_\theta(x|z) &= p_\theta(x)\\ q_\phi(z|x) &= p(z)\end{aligned}$$

TODO: why will posterior collapse happen

Considering VAE's objective function ELBO, since the second term KL divergence is always non-negative, to maximize the function, the best way is to make the second term equal to 0, and maximize the first term as possible.

As stated in the preceding section, $D_{KL}(q_\phi(z|x)||p(z))$ is equal to zero only when $q_\phi(z|x) = p(z)$ for any x and z.

For the first term, according to Jensen's Inequality $E_{p(z)}[\log p(x|z)] \leq \log\sum_z[p(x|z)p(z)] = \log p(x)$, and the equality holds when and only when x is independent of z. Thus $p_\theta(x|z) = p_\theta(x)$.

Posterior collapse problem is much investigated and researched. Xu and Durrett (2018) and He et al. (2019) are two successful methods to avoid posterior collapse.

4 Empirical Investigation

4.1 Corpus

The corpus used in this project is Penn Treebank (PTB), which is one of the most widely used datasets for evaluating language models. Penn Treebank selected 2,499 stories from a three year Wall Street Journal collection of 98,732 stories for syntactic annotation (Marcus et al., 1993). It consists of a train set (930k tokens), a validation set (74k tokens), and a test set (82k tokens). All words outside the 10k vocabulary were mapped to a special token (unknown word) (Mikolov et al., 2011).

4.2 Loss Functions for Language Model

The loss functions used for RNN language model and VAE are negative log likelihood (NLL) and negative ELBO (NegELBO), respectively.

$$\begin{aligned}&NLL = -\log p(\mathbf{x}) = -\prod_i p(x_i|x_1,\ldots,x_{i-1})\\ &NegELBO =\\ &D_{KL}(q_\phi(z|x)||p(z)) - \mathop{\mathbb{E}}_{q_\phi(z|x)}[\log p_\theta(x|z)]\end{aligned}$$

4.3 Results

4.3.1 Comparison of RNN Language Model and VAE

Fig.1 shows the loss over different epochs of both training and validation data of RNN LM and VAE, respectively. Specifically, different RNN variants, including basic RNN, LSTM and GRU are used separately.

It is shown on the figure that after the third epoch, the NLL of the validation set increased dramatically and after 10 epochs, the NLL was even larger than that before training.

Since ELBO is the lower bound of log likelihood, negative ELBO is also a upper bound of NLL. It is shown in the graphs that after 10 epochs, the NegELBO of the validation set in VAE is even higher that the NLL of RNN LM. We could simply tell from the loss function of the two models that VAE has better performance than RNN LM on the dataset. This conclusion holds for the three selected RNN variants.

4.3.2 Comparison of Different Variants of RNN Model

Fig.1 also compares the performance in terms of loss of different variants of RNN: basic RNN, LSTM and GRU.

For RNN LM, the three variants have similar scores on the training set, but LSTM showed better performance on the validation set.

For the VAE, LSTM has significantly lower loss on both training set and validation set than GRU and RNN. GRU did not gain any improvement in terms of loss compared to basic RNN model.

4.3.3 Annealing Function

Fig.2 shows the KL loss of training and validation set of the VAE model. A KL loss close to 0 often indicates posterior collapse. Both basic RNN and LSTM models suffered from posterior collapsing in this experiment. Their training set KL loss decreases to almost zero soon after the training process started.

In order to deal with the posterior collapse problem, an additional annealing function is applied on the KL term of the loss function NegELBO:

$$\max_{\theta,q(z|x)} \mathop{\mathbb{E}}_{q_\phi(z|x)} [\log p_\theta(x|z)] - \alpha(t) \cdot D_{KL}(q_\phi(z|x)||p(z))$$

The KL weight $\alpha(t)$ in the above objective function is the annealing function. t represents the steps in the training process of VAE, $\alpha(t)$ is a value in $[0, 1]$.

At the start of training, set KL weight to zero, so that the model learns to encode as much information in $\mathbf{z}$ as it can. Then as training progresses, we gradually increases the weight until it reaches 1 to force the model to smooth out its encodings and pack them into the prior. When KL weight is 1, the weighted cost function is equivalent to the true variational lower bound (Bowman et al., 2015).

In this project, experiments on a basic RNN model was conducted with several different annealing functions, including linear function, square root function, and Sigmoid function.

Linear annealing function

$$\alpha(t) = \begin{cases} kt & kt \leq 1 \\ 1 & kt > 1 \end{cases}$$

In this experiment, different values of the parameter k are selected based on the total steps, and the resulting KL loss and the NegELBO loss are illustrated in Fig.3.

When $k = 7.5e^{-3}$ and $k = 1e^{-4}$, KL loss only approaches to zero in the last few epochs. $k = 2e^{-4}$ does not help much in terms of posterior collapsing. However, The NegELBO of both training and validation set is nearly identical for various choices of k and even when no annealing is applied. Although the NegELBO seems similar, it is only a upper bound of NLL, which should be the true metric of the model.

Square root annealing function

$$\alpha(t) = \begin{cases} k\sqrt{t} & k\sqrt{t} \leq 1 \\ 1 & k\sqrt{t} > 1 \end{cases}$$

In this experiment, different values of the parameter k are selected based on the total steps, and the resulting KL loss and the NegELBO loss are illustrated in Fig.4.

Square root functions have sharper slope in the beginning of the training process. But the overall results, including its impact on KL loss or NegELBO on the dataset, are much similar to the linear annealing functions.

Sigmoid annealing function

$$sigmoid(z) = \frac{1}{1 + e^{-z}}$$

$$\alpha(t) = \frac{1}{1 + e^{-k(z-\mu)}}$$

Sigmoid function is a bounded function with a lower bound of 0 and a upper bound of 1. In addition, apply some linear transformation on t in order to make the KL weight 0 in the beginning of the training, and to avoid too steep slope in the function. See Fig.5.

The sigmoid annealing also prevented posterior collapse in a few epochs in the beginning, when

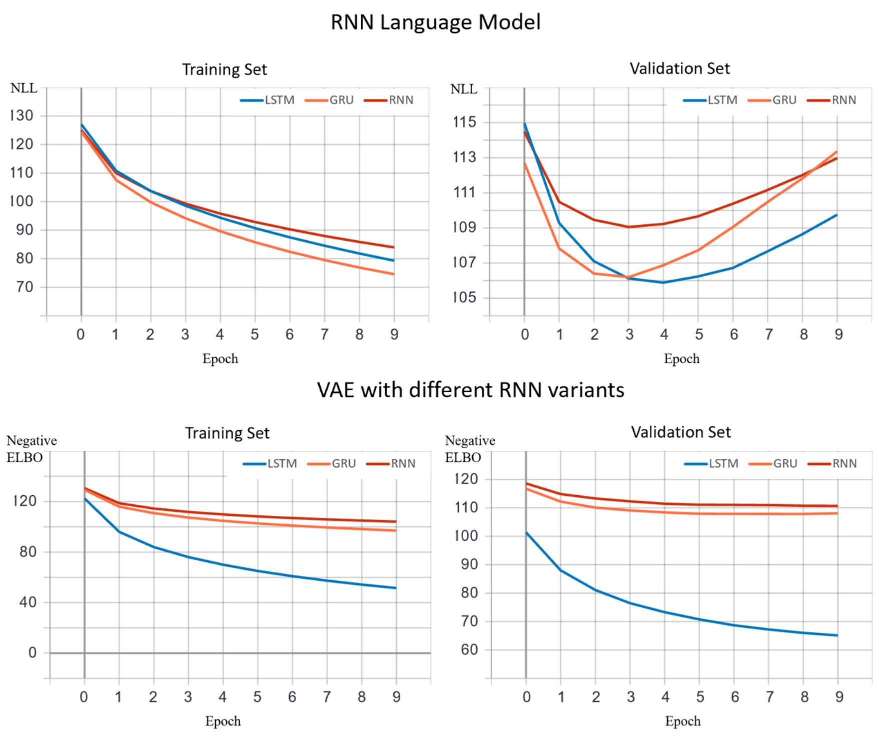

Figure 1: The upper graph illustrates the negative Log Likelihood of RNN Language Model over different epochs. The lower graph illustrates the Negative ELBO of VAE over different epochs. The red line denotes the use of basic RNN model, whereas the blue line indicates LSTM model and the orange line represents the result for GRU.

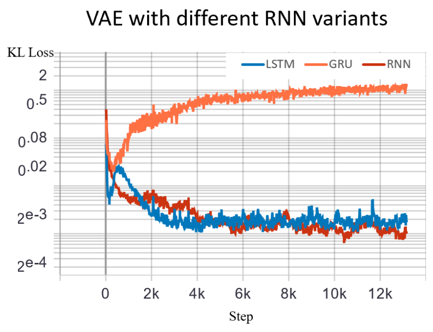

Figure 2: The KL loss of the training set of the VAE model. The x-axis represents the steps in the training process.

the KL weight is also small. And in the mean time, the NegELBO of the validation set if larger than that without annealing. In the last few epochs, the KL weight approached to 1, KL loss is close to 0, and the NegELBO of the validation set became identical, too.

In conclusion, application of annealing functions will help in preventing posterior collapse. The loss of both training and test data will probability remain the same. But since NegELBO is only an upper bound approximate of the NLL, we couldn't get much information about the impact of annealing on the log likelihood of the data.

4.3.4 Latent Size

Latent size is the number of elements in **z**. Intuitively, different choices of latent size may also have influence on the result. After all, the larger the latent size, the more categories of heterogeneous languages the VAE may recognize. The latent size of 8, 16, and 32 was selected, and the corresponding result is illustrated in Fig.6. We could tell from the figures that the NegELBO loss for the chosen latent sizes are nearly identical. The only difference can be spotted in KL loss for both training and validation sets.

References

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

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

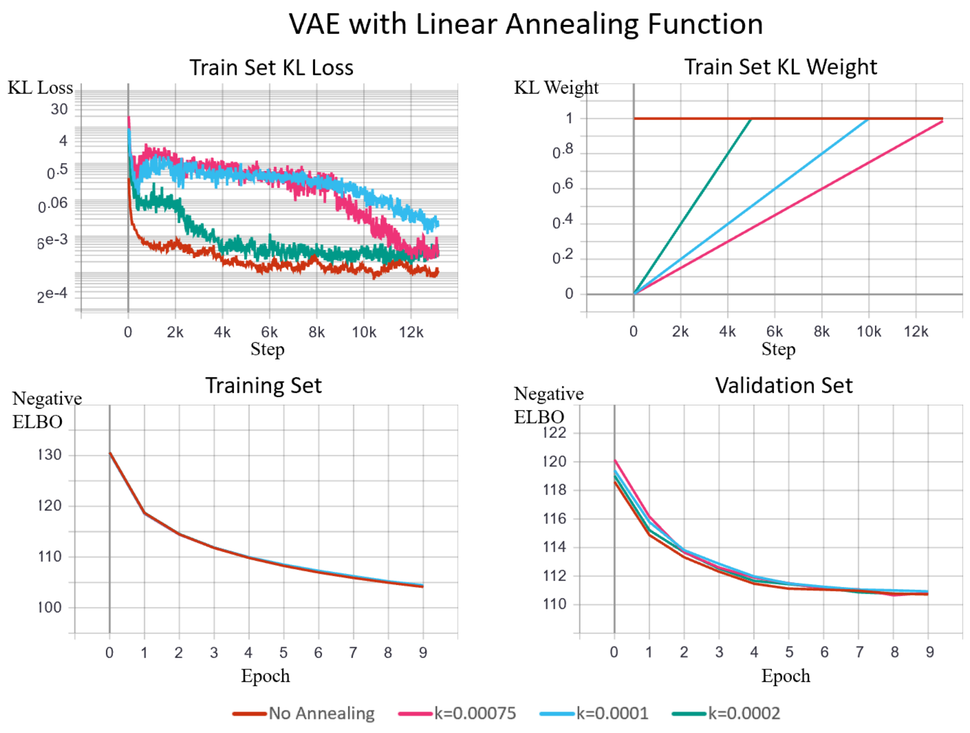

Figure 3: The KL loss and KL weight of VAE training data over steps with different linear annealing functions applied. The lower subfigures are the NegELBO loss of training and validation set over epochs. k is the slope of the linear function because reaching one.

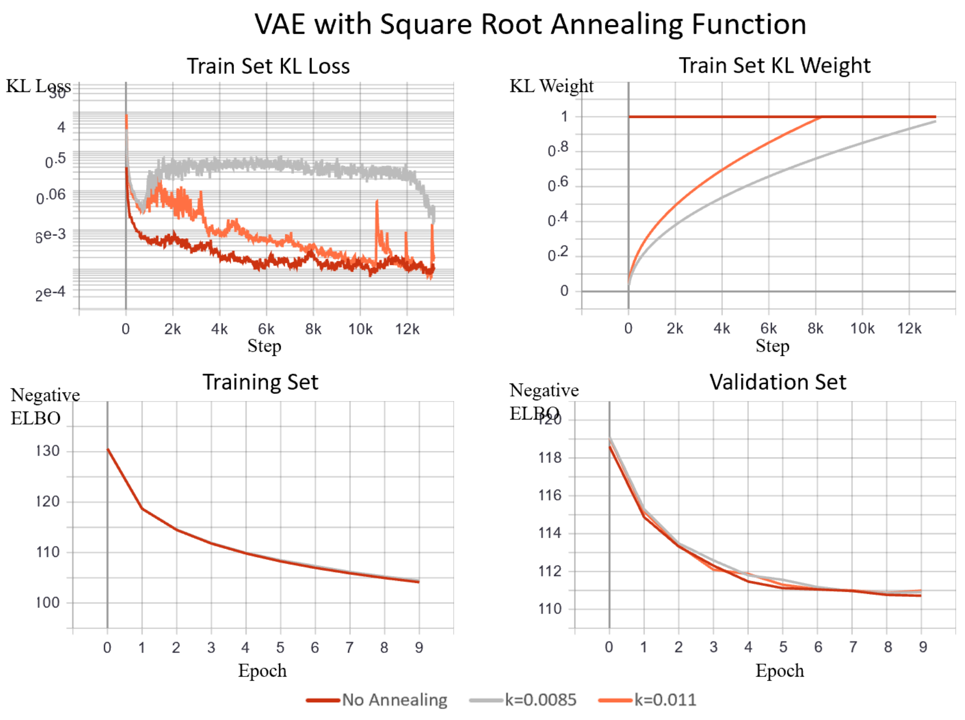

Figure 4: The KL loss and KL weight of VAE training data over steps with the square root annealing functions with different parameter k applied. The lower subfigures are the NegELBO loss of training and validation set over epochs.

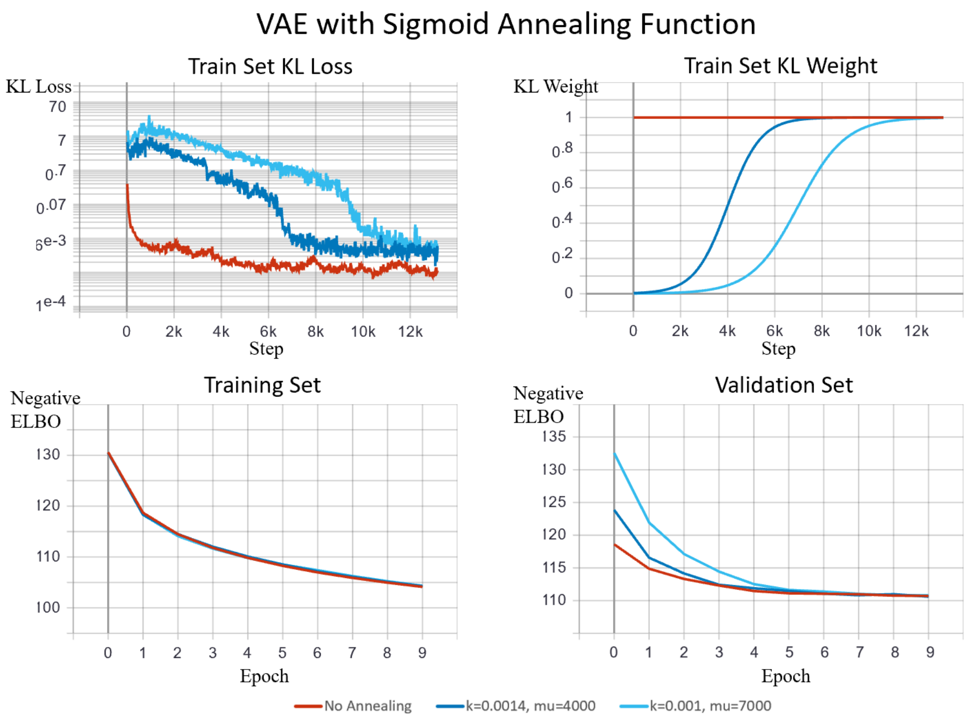

Figure 5: The KL loss and KL weight of VAE training data over steps with the sigmoid annealing functions with different parameter k and μ applied. The lower subfigures are the NegELBO loss of training and validation set over epochs.

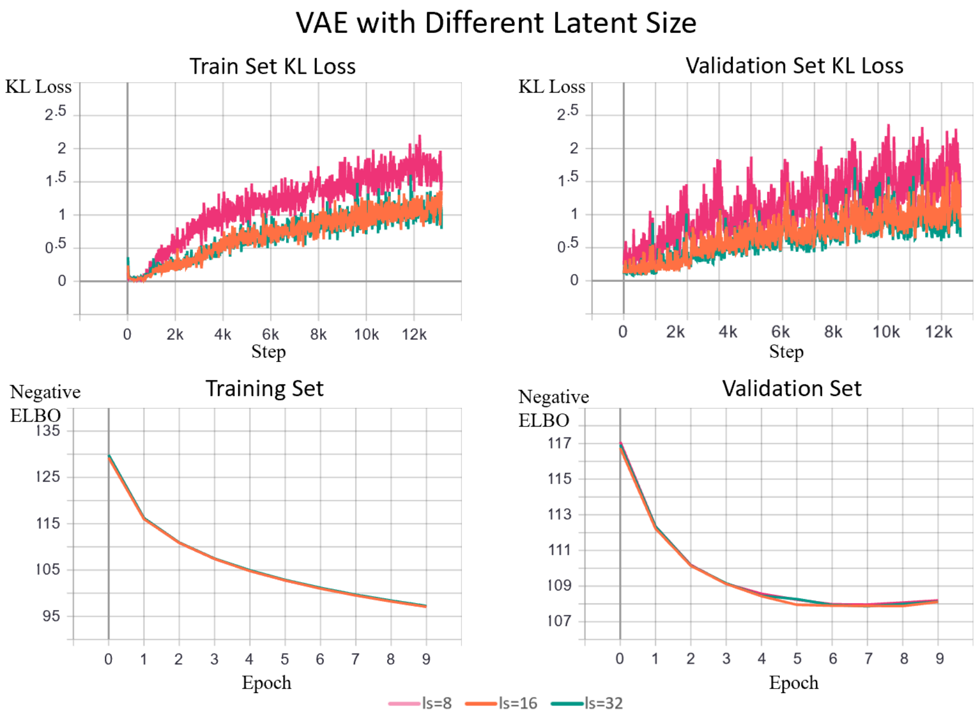

Figure 6: The KL loss and NegELBO of train and validation set when applying different latent size ls.

Junxian He, Daniel Spokoyny, Graham Neubig, and Taylor Berg-Kirkpatrick. 2019. Lagging inference networks and posterior collapse in variational autoencoders. *arXiv preprint arXiv:1901.05534*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank.

Tomáš Mikolov, Anoop Deoras, Stefan Kombrink, Lukáš Burget, and Jan Černockỳ. 2011. Empirical evaluation and combination of advanced language modeling techniques. In *Twelfth Annual Conference of the International Speech Communication Association*.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*.

Jiacheng Xu and Greg Durrett. 2018. Spherical latent spaces for stable variational autoencoders. *arXiv preprint arXiv:1808.10805*.