# Improved Variational Autoencoders for Text Modeling using Dilated Convolutions

**Zichao Yang** [1] **Zhiting Hu** [1] **Ruslan Salakhutdinov** [1] **Taylor Berg-Kirkpatrick** [1]

## Abstract

Recent work on generative text modeling has found that variational autoencoders (VAE) with LSTM decoders perform worse than simpler LSTM language models (Bowman et al., 2015). This negative result is so far poorly understood, but has been attributed to the propensity of LSTM decoders to ignore conditioning information from the encoder. In this paper, we experiment with a new type of decoder for VAE: a dilated CNN. By changing the decoder's dilation architecture, we control the size of context from previously generated words. In experiments, we find that there is a trade-off between contextual capacity of the decoder and effective use of encoding information. We show that when carefully managed, VAEs can outperform LSTM language models. We demonstrate perplexity gains on two datasets, representing the first positive language modeling result with VAE. Further, we conduct an in-depth investigation of the use of VAE (with our new decoding architecture) for semi-supervised and unsupervised labeling tasks, demonstrating gains over several strong baselines.

## 1. Introduction

Generative models play an important role in NLP, both in their use as language models and because of their ability to effectively learn from unlabeled data. By parameterzing generative models using neural nets, recent work has proposed model classes that are particularly expressive and can pontentially model a wide range of phenomena in language and other modalities. We focus on a specific instance of this class: the variational autoencoder[1] (VAE) (Kingma & Welling, 2013).

The generative story behind the VAE (to be described in detail in the next section) is simple: First, a continuous latent representation is sampled from a multivariate Gaussian. Then, an output is sampled from a distribution parameterized by a neural decoder, conditioned on the latent representation. The latent representation (treated as a latent variable during training) is intended to give the model more expressive capacity when compared with simpler neural generative models–for example, conditional language models. The choice of decoding architecture and final output distribution, which connect the latent representation to output, depends on the kind of data being modeled. The VAE owes its name to an accompanying variational technique (Kingma & Welling, 2013) that has been successfully used to train such models on image data (Gregor et al., 2015; Salimans et al., 2015; Yan et al., 2016).

The application of VAEs to text data has been far less successful (Bowman et al., 2015; Miao et al., 2016). The obvious choice for decoding architecture for a textual VAE is an LSTM, a typical workhorse in NLP. However, Bowman et al. (2015) found that using an LSTM-VAE for text modeling yields higher perplexity on held-out data than using an LSTM language model. In particular, they observe that the LSTM decoder in VAE does not make effective use of the latent representation during training and, as a result, VAE collapses into a simple language model. Related work (Miao et al., 2016; Larochelle & Lauly, 2012; Mnih & Gregor, 2014) has used simpler decoders that model text as a bag of words. Their results indicate better use of latent representations, but their decoders cannot effectively model longer-range dependencies in text and thus underperform in terms of final perplexity.

Motivated by these observations, we hypothesize that the contextual capacity of the decoder plays an important role in whether VAEs effectively condition on the latent representation when trained on text data. We propose the use of a dilated CNN as a decoder in VAE, inspired by the recent success of using CNNs for audio, image and language

[1]Carnegie Mellon University. Correspondence to: Zichao Yang <zichaoy@cs.cmu.edu>.

*Proceedings of the $34^{th}$ International Conference on Machine Learning*, Sydney, Australia, PMLR 70, 2017. 

[1]The name VAE is often used to refer to both a model class and an associated inference procedure.

modeling (van den Oord et al., 2016a; Kalchbrenner et al., 2016a; van den Oord et al., 2016b). In contrast with prior work where extremely large CNNs are used, we exploit the dilated CNN for its flexibility in varying the amount of conditioning context. In the two extremes, depending on the choice of dilation, the CNN decoder can reproduce a simple MLP using a bags of words representation of text, or can reproduce the long-range dependence of recurrent architectures (like an LSTM) by conditioning on the entire history. Thus, by choosing a dilated CNN as the decoder, we are able to conduct experiments where we vary contextual capacity, finding a sweet spot where the decoder can accurately model text but does not yet overpower the latent representation.

We demonstrate that when this trade-off is correctly managed, textual VAEs can perform substantially better than simple LSTM language models, a finding consistent with recent image modeling experiments using variational lossy autoencoders (Chen et al., 2016). We go on to show that VAEs with carefully selected CNN decoders can be quite effective for semi-supervised classification and unsupervised clustering, outperforming several strong baselines (from (Dai & Le, 2015)) on both text categorization and sentiment analysis.

Our contributions are as follows: First, we propose the use of a dilated CNN as a new decoder for VAE. We then empirically evaluate several dilation architectures with different capacities, finding that reduced contextual capacity leads to stronger reliance on latent representations. By picking a decoder with suitable contextual capacity, we find our VAE performs better than LSTM language models on two data sets. We also explore the use of dilated CNN VAEs for semi-supervised classification and find they perform better than strong baselines from (Dai & Le, 2015). Finally, we verify that the same framework can be used effectively for unsupervised clustering.

## 2. Model

In this section, we begin by providing background on the use of variational autoencoders for language modeling. Then we introduce the dilated CNN architecture that we will use as a new decoder for VAE in experiments. Finally, we describe the generalization of VAE that we will use to conduct experiments on semi-supervised classification.

### 2.1. Background on Variational Autoencoders

Neural language models (Mikolov et al., 2010) typically generate each token $x_t$ conditioned on the entire history of previously generated tokens:

$$p(\mathbf{x}) = \prod_t p(x_t|x_1, x_2, ..., x_{t-1}). \quad (1)$$

State-of-the-art language models often parametrize these conditional probabilities using RNNs, which compute an evolving hidden state over the text which is used to predict each $x_t$. This approach, though effective in modeling text, does not explicitly model variance in higher-level properties of entire utterances (e.g. topic or style) and thus can have difficulty with heterogeneous datasets.

Bowman et al. (2015) propose a different approach to generative text modeling inspired by related work on vision (Kingma & Welling, 2013). Instead of directly modeling the joint probability $p(\mathbf{x})$ as in Equation 1, we specify a generative process for which $p(\mathbf{x})$ is a marginal distribution. Specifically, we first generate a continuous latent vector representation $\mathbf{z}$ from a multivariate Gaussian prior $p_\theta(\mathbf{z})$, and then generate the text sequence $\mathbf{x}$ from a conditional distribution $p_\theta(\mathbf{x}|\mathbf{z})$ parameterized using a neural net (often called the generation model or decoder). Because this model incorporates a latent variable that modulates the entire generation of each whole utterance, it may be better able to capture high-level sources of variation in the data. Specifically, in contrast with Equation 1, this generating distribution conditions on latent vector representation $\mathbf{z}$:

$$p_\theta(\mathbf{x}|\mathbf{z}) = \prod_t p_\theta(x_t|x_1, x_2, ..., x_{t-1}, \mathbf{z}). \quad (2)$$

To estimate model parameters $\theta$ we would ideally like to maximize the marginal probability $p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{z})p_\theta(\mathbf{x}|\mathbf{z})d\mathbf{z}$. However, computing this marginal is intractable for many decoder choices. Thus, the following variational lower bound is often used as an objective (Kingma & Welling, 2013):

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

Here, $q_\phi(\mathbf{z}|\mathbf{x})$ is an approximation to the true posterior (often called the recognition model or encoder) and is parameterized by $\phi$. Like the decoder, we have a choice of neural architecture to parameterize the encoder. However, unlike the decoder, the choice of encoder does not change the model class – it only changes the variational approximation used in training, which is a function of both the model parameters $\theta$ *a*nd the approximation parameters $\phi$. Training seeks to optimize these parameters jointly using stochastic gradient ascent. A final wrinkle of the training procedure involves a stochastic approximation to the gradients of the variational objective (which is itself intractable). We omit details here, noting only that the final distribution of the posterior approximation $q_\phi(\mathbf{z}|\mathbf{x})$ is typically assumed to be Gaussian so that a re-parametrization trick can be used, and refer readers to (Kingma & Welling, 2013).

### 2.2. Training Collapse with Textual VAEs

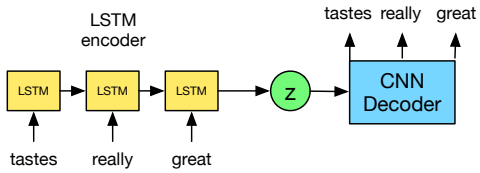
Together, this combination of generative model and variational inference procedure are often referred to as a variational autoencoder (VAE). We can also view the VAE as a regularized version of the autoencoder. Note, however, that while VAEs are valid probabilistic models whose likelihood can be evaluated on held-out data, autoencoders are not valid models. If only the first term of the VAE variational bound $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})]$ is used as an objective, the variance of the posterior probability $q_\phi(\mathbf{z}|\mathbf{x})$ will become small and the training procedure reduces to an autoencoder. It is the KL-divergence term, $\mathrm{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}))$, that discourages the VAE memorizing each $\mathbf{x}$ as a single latent point.

While the KL term is critical for training VAEs, historically, instability on text has been evidenced by the KL term becoming vanishingly small during training, as observed by Bowman et al. (2015). When the training procedure collapses in this way, the result is an encoder that has duplicated the Gaussian prior (instead of a more interesting posterior), a decoder that completely ignores the latent variable $\mathbf{z}$, and a learned model that reduces to a simpler language model. We hypothesize that this collapse condition is related to the contextual capacity of the decoder architecture. The choice encoder and decoder depends on the type of data. For images, these are typically MLPs or CNNs. LSTMs have been used for text, but have resulted in training collapse as discussed above (Bowman et al., 2015). Here, we propose to use a dilated CNN as the decoder instead. In one extreme, when the effective contextual width of a CNN is very large, it resembles the behavior of LSTM. When the width is very small, it behaves like a bag-of-words model. The architectural flexibility of dilated CNNs allows us to change the contextual capacity and conduct experiments to validate our hypothesis: decoder contextual capacity and effective use of encoding information are directly related. We next describe the details of our decoder.

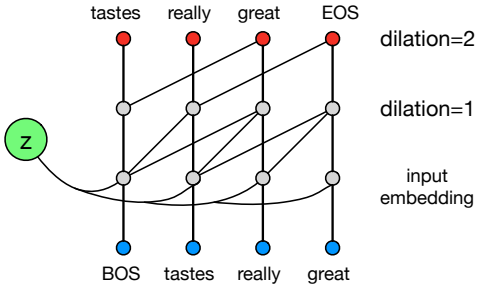
### 2.3. Dilated Convolutional Decoders

The typical approach to using CNNs used for text generation (Kalchbrenner et al., 2016a) is similar to that used for images (Krizhevsky et al., 2012; He et al., 2016), but with the convolution applied in one dimension. We take this approach here in defining our decoder.

**One dimensional convolution**: For a CNN to serve as a decoder for text, generation of $x_t$ must only condition on past tokens $x_{<t}$. Applying the traditional convolution will break this assumption and use tokens $x_{\geq t}$ as inputs to predict $x_t$. In our decoder, we avoid this by simply shifting the input by several slots (van den Oord et al., 2016b). With a convolution with filter size of $k$ and using $n$ layers, our effective filter size (the number of past tokens to condition to in predicting $x_t$) would be $(k-1) \times n + 1$. Hence, the filter size would grow linearly with the depth of the network.




(a) VAE training graph using a dilated CNN decoder.

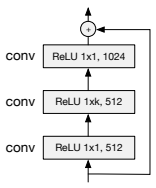



(b) Digram of dilated CNN decoder.

Figure 1: Our training and model architectures for textual VAE using a dilated CNN decoder.

**Dilation**: Dilated convolution (Yu & Koltun, 2015) was introduced to greatly increase the effective receptive field size without increasing the computational cost. With dilation $d$, the convolution is applied so that $d-1$ inputs are skipped each step. Causal convolution can be seen a special case with $d = 1$. With dilation, the effective receptive size grows exponentially with network depth. In Figure 1b, we show dilation of sizes of 1 and 2 in the first and second layer, respectively. Suppose the dilation size in the $i$-th layer is $d_i$ and we use the same filter size $k$ in all layers, then the effective filter size is $(k-1)\sum_i d_i + 1$. The dilations are typically set to double every layer $d_{i+1} = 2d_i$, so the effective receptive field size can grow exponentially. Hence, the contextual capacity of a CNN can be controlled across a greater range by manipulating the filter size, dilation size and network depth. We use this approach in experiments.

**Residual connection**: We use residual connection (He et al., 2016) in the decoder to speed up convergence and enable training of deeper models. We use a residual block (shown to the right) similar to that of (Kalchbrenner et al., 2016a). We use three convolutional layers with filter size $1 \times 1, 1 \times k, 1 \times 1$, respectively, and ReLU activation be-



tween convolutional layers.

**Overall architecture**: Our VAE architecture is shown in Figure 1a. We use LSTM as the encoder to get the posterior probability $q(\mathbf{z}|\mathbf{x})$, which we assume to be diagonal Gaussian. We parametrize the mean $\mu$ and variance $\sigma$ with LSTM output. We sample $\mathbf{z}$ from $q(\mathbf{z}|\mathbf{x})$, the decoder is conditioned on the sample by concatenating $\mathbf{z}$ with every word embedding of the decoder input.

### 2.4. Semi-supervised VAE

In addition to conducting language modeling experiments, we will also conduct experiments on semi-supervised classification of text using our proposed decoder. In this section, we briefly review semi-supervised VAEs of (Kingma et al., 2014) that incorporate discrete labels as additional variables. Given the labeled set $(x, y) \sim D_L$ and the unlabeled set $x \sim D_U$, (Kingma et al., 2014) proposed a model whose latent representation contains continuous vector $\mathbf{z}$ and discrete label $\mathbf{y}$:

$$p(\mathbf{x}, \mathbf{y}, \mathbf{z}) = p(\mathbf{y})p(\mathbf{z})p(\mathbf{x}|\mathbf{y}, \mathbf{z}). \tag{3}$$

The semi-supervised VAE fits a discriminative network $q(\mathbf{y}|\mathbf{x})$, an inference network $q(\mathbf{z}|\mathbf{x}, \mathbf{y})$ and a generative network $p(\mathbf{x}|\mathbf{y}, \mathbf{z})$ jointly as part of optimizing a variational lower bound similar that of basic VAE. For labeled data $(\mathbf{x}, \mathbf{y})$, this bound is:

$$\begin{aligned}\log p(\mathbf{x}, \mathbf{y}) \geq & \mathbb{E}_{q(\mathbf{z}|\mathbf{x},\mathbf{y})}[\log p(\mathbf{x}|\mathbf{y}, \mathbf{z})] \\ & - \mathrm{KL}(q(\mathbf{z}|\mathbf{x}, \mathbf{y})||p(\mathbf{z})) + \log p(\mathbf{y}) \\ = & L(\mathbf{x}, \mathbf{y}) + \log p(\mathbf{y}).\end{aligned}$$

For unlabeled data $\mathbf{x}$, the label is treated as a latent variable, yielding:

$$\begin{aligned}\log p(\mathbf{x}) \geq & U(\mathbf{x}) \\ = & \mathbb{E}_{q(\mathbf{y}|\mathbf{x})}\big[\mathbb{E}_{q(\mathbf{z}|\mathbf{x},\mathbf{y})}[\log p(\mathbf{x}|\mathbf{y}, \mathbf{z})] \\ & - \mathrm{KL}(q(\mathbf{z}|\mathbf{x}, \mathbf{y})||p(\mathbf{z})) + \log p(\mathbf{y}) - \log q(\mathbf{y}|\mathbf{x})\big] \\ = & \sum_y q(\mathbf{y}|\mathbf{x})L(\mathbf{x}, \mathbf{y}) - \mathrm{KL}(q(\mathbf{y}|\mathbf{x})||p(\mathbf{y})).\end{aligned}$$

Combining the labeled and unlabeled data terms, we have the overall objective as:

$$\begin{aligned}J = & \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim D_L}[L(\mathbf{x}, \mathbf{y})] + \mathbb{E}_{\mathbf{x}\sim D_U}[U(\mathbf{x})] \\ & + \alpha\, \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim D_L}[\log q(\mathbf{y}|\mathbf{x})],\end{aligned}$$

where $\alpha$ controls the trade off between generative and discriminative terms.

**Gumbel-softmax**: Jang et al. (2016); Maddison et al. (2016) propose a continuous approximation to sampling from a categorical distribution. Let $u$ be a categorical distribution with probabilities $\pi_1, \pi_2, ..., \pi_c$. Samples from $u$ can be approximated using:

$$y_i = \frac{\exp((\log(\pi_i) + g_i)/\tau)}{\sum_{j=1}^{c} \exp((\log(\pi_j) + g_j)/\tau)}, \tag{4}$$

where $g_i$ follows Gumbel(0, 1). The approximation is accurate when $\tau \to 0$ and smooth when $\tau > 0$. In experiments, we use Gumbel-Softmax to approximate the samples from $p(\mathbf{y}|\mathbf{x})$ to reduce the computational cost. As a result, we can directly back propagate the gradients of $U(\mathbf{x})$ to the discriminator network. We anneal $\tau$ so that sample variance is small when training starts and then gradually decrease $\tau$.

**Unsupervised clustering**: In this section we adapt the same framework for unsupervised clustering. We directly minimize the objective $U(\mathbf{x})$, which is consisted of two parts: reconstruction loss and KL regularization on $q(\mathbf{y}|\mathbf{x})$. The first part encourages the model to assign $\mathbf{x}$ to label $\mathbf{y}$ such that the reconstruction loss is low. We find that the model can easily get stuck in two local optimum: the KL term is very small and $q(\mathbf{y}|\mathbf{x})$ is close to uniform distribution or the KL term is very large and all samples collapse to one class. In order to make the model more robust, we modify the KL term by:

$$\mathrm{KL}_{\mathbf{y}} = \max(\gamma, \mathrm{KL}(q(\mathbf{y}|\mathbf{x})|p(\mathbf{y})). \tag{5}$$

That is, we only minimize the KL term when it is large enough.

## 3. Experiments

### 3.1. Data sets

Since we would like to investigate VAEs for language modeling and semi-supervised classification, the data sets should be suitable for both purposes. We use two large scale document classification data sets: Yahoo Answer and Yelp15 review, representing topic classification and sentiment classification data sets respectively (Tang et al., 2015; Yang et al., 2016; Zhang et al., 2015). The original data sets contain millions of samples, of which we sample 100k as training and 10k as validation and test from the respective partitions. The detailed statistics of both data sets are in Table 1. Yahoo Answer contains 10 topics including Society & Culture, Science & Mathematics etc. Yelp15 contains 5 level of rating, with higher rating better.

| Data | classes | documents | average #w | vocabulary |
|---|---|---|---|---|
| Yahoo | 10 | 100k | 78 | 200k |
| Yelp15 | 5 | 100k | 96 | 90k |

Table 1: Data statistics

| Model | Size | NLL (KL) | PPL |
|---|---|---|---|
| LSTM-LM | $< i$ | 334.9 | 66.2 |
| LSTM-VAE** | $< i$ | 342.1 (0.0) | 72.5 |
| LSTM-VAE** + init | $< i$ | 339.2 (0.0) | 69.9 |
| SCNN-LM | 15 | 345.3 | 75.5 |
| SCNN-VAE | 15 | 337.8 (13.3) | 68.7 |
| SCNN-VAE + init | 15 | 335.9 (13.9) | 67.0 |
| MCNN-LM | 63 | 338.3 | 69.1 |
| MCNN-VAE | 63 | 336.2 (11.8) | 67.3 |
| MCNN-VAE + init | 63 | 334.6 (12.6) | 66.0 |
| LCNN-LM | 125 | 335.4 | 66.6 |
| LCNN-VAE | 125 | 333.9 (6.7) | 65.4 |
| LCNN-VAE + init | 125 | **332.1 (10.0)** | **63.9** |
| VLCNN-LM | 187 | 336.5 | 67.6 |
| VLCNN-VAE | 187 | 336.5 (0.7) | 67.6 |
| VLCNN-VAE + init | 187 | 335.8 (3.8) | 67.0 |

(a) Yahoo

| Model | Size | NLL (KL) | PPL |
|---|---|---|---|
| LSTM-LM | $< i$ | 362.7 | 42.6 |
| LSTM-VAE** | $< i$ | 372.2 (0.3) | 47.0 |
| LSTM-VAE** + init | $< i$ | 368.9 (4.7) | 46.4 |
| SCNN-LM | 15 | 371.2 | 46.6 |
| SCNN-VAE | 15 | 365.6 (9.4) | 43.9 |
| SCNN-VAE + init | 15 | 363.7 (10.3) | 43.1 |
| MCNN-LM | 63 | 366.5 | 44.3 |
| MCNN-VAE | 63 | 363.0 (6.9) | 42.8 |
| MCNN-VAE + init | 63 | 360.7 (9.1) | 41.8 |
| LCNN-LM | 125 | 363.5 | 43.0 |
| LCNN-VAE | 125 | 361.9 (6.4) | 42.3 |
| LCNN-VAE + init | 125 | **359.1 (7.6)** | **41.1** |
| VLCNN-LM | 187 | 364.8 | 43.7 |
| VLCNN-VAE | 187 | 364.3 (2.7) | 43.4 |
| VLCNN-VAE + init | 187 | 364.7 (2.2) | 43.5 |

(b) Yelp

Table 2: Language modeling results on the test set. ** is from (Bowman et al., 2015). We report negative log likelihood (NLL) and perplexity (PPL) on the test set. The KL component of NLL is given in parentheses. Size indicates the effective filter size. VAE + init indicates pretraining of only the encoder using an LSTM LM.

### 3.2. Model configurations and Training details

We use an LSTM as an encoder for VAE and explore LSTMs and CNNs as decoders. For CNNs, we explore several different configurations. We set the convolution filter size to be 3 and gradually increase the depth and dilation from [1, 2, 4], [1, 2, 4, 8, 16] to [1, 2, 4, 8, 16, 1, 2, 4, 8, 16]. They represent small, medium and large model and we name them as SCNN, MCNN and LCNN. We also explore a very large model with dilations [1, 2, 4, 8, 16, 1, 2, 4, 8, 16, 1, 2, 4, 8, 16] and name it as VLCNN. The effective filter size are 15, 63, 125 and 187 respectively. We use the last hidden state of the encoder LSTM and feed it though an MLP to get the mean and variance of $q(\mathbf{z}|\mathbf{x})$, from which we sample $\mathbf{z}$ and then feed it through an MLP to get the starting state of decoder. For the LSTM decoder, we follow (Bowman et al., 2015) to use it as the initial state of LSTM and feed it to every step of LSTM. For the CNN decoder, we concatenate it with the word embedding of every decoder input.

The architecture of the Semi-supervised VAE basically follows that of the VAE. We feed the last hidden state of the encoder LSTM through a two layer MLP then a softmax to get $q(\mathbf{y}|\mathbf{x})$. We use Gumbel-softmax to sample $\mathbf{y}$ from $q(\mathbf{y}|\mathbf{x})$. We then concatenate $\mathbf{y}$ with the last hidden state of encoder LSTM and feed them throught an MLP to get the mean and variance of $q(\mathbf{z}|\mathbf{y}, \mathbf{x})$. $\mathbf{y}$ and $\mathbf{z}$ together are used as the starting state of the decoder.

We use a vocabulary size of 20k for both data sets and set the word embedding dimension to be 512. The LSTM dimension is 1024. The number of channels for convolutions in CNN decoders is 512 internally and 1024 externally, as shown in Section 2.3. We select the dimension of $\mathbf{z}$ from [32, 64]. We find our model is not sensitive to this parameter.

We use Adam (Kingma & Ba, 2014) to optimize all models and the learning rate is selected from [2e-3, 1e-3, 7.5e-4] and $\beta_1$ is selected from [0.5, 0.9]. Empirically, we find learning rate 1e-3 and $\beta_1 = 0.5$ to perform the best. We select drop out ratio of LSTMs (both encoder and decoder) from [0.3, 0.5]. Following (Bowman et al., 2015), we also use drop word for the LSTM decoder, the drop word ratio is selected from [0, 0.3, 0.5, 0.7]. For the CNN decoder, we use a drop out ratio of 0.1 at each layer. We do not use drop word for CNN decoders. We use batch size of 32 and all model are trained for 40 epochs. We start to half the learning rate every 2 epochs after epoch 30. Following (Bowman et al., 2015), we use KL cost annealing strategy. We set the initial weight of KL cost term to be 0.01 and increase it linearly until a given iteration $T$. We treat $T$ as a hyper parameter and select it from [10k, 40k, 80k].

### 3.3. Language modeling results

The results for language modeling are shown in Table 2. We report the negative log likelihood (NLL) and perplexity (PPL) of the test set. For the NLL of VAEs, we decompose it into reconstruction loss and KL divergence and report the KL divergence in the parenthesis. To better visualize these results, we plot the results of Yahoo data set (Table 2a) in Figure 2.

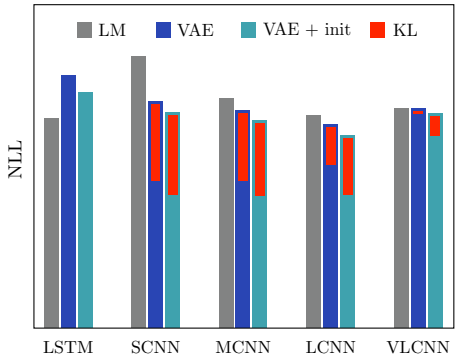
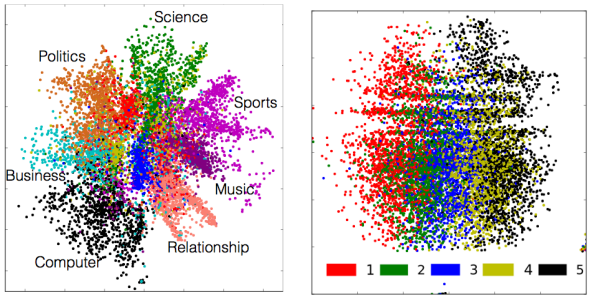



Figure 2: NLL decomposition of Table 2a. Each group consists of three bars, representing LM, VAE and VAE+init. For VAE, we decompose the loss into reconstruction loss and KL divergence, shown in blue and red respectively. We subtract all loss values with 300 for better visualization.

We first look at the LM results for Yahoo data set. As we gradually increase the effective filter size of CNN from SCNN, MCNN to LCNN, the NLL decreases from 345.3, 338.3 to 335.4. The NLL of LCNN-LM is very close to the NLL of LSTM-LM 334.9. But VLCNN-LM is a little bit worse than LCNN-LM, this indicates a little bit of over-fitting.

We can see that LSTM-VAE is worse than LSTM-LM in terms of NLL and the KL term is nearly zero, which verifies the finding of (Bowman et al., 2015). When we use CNNs as the decoders for VAEs, we can see improvement over pure CNN LMs. For SCNN, MCNN and LCNN, the VAE results improve over LM results from 345.3 to 337.8, 338.3 to 336.2, and 335.4 to 333.9 respectively. The improvement is big for small models and gradually decreases as we increase the decoder model contextual capacity. When the model is as large as VLCNN, the improvement diminishes and the VAE result is almost the same with LM result. This is also reflected in the KL term, SCNN-VAE has the largest KL of 13.3 and VLCNN-VAE has the smallest KL of 0.7. When LCNN is used as the decoder, we obtain an optimal trade off between using contextual information and latent representation. LCNN-VAE achieves a NLL of 333.9, which improves over LSTM-LM with NLL of 334.9.

We find that if we initialize the parameters of *LSTM encoder* with parameters of LSTM language model, we can improve the VAE results further. This indicates better encoder model is also a key factor for VAEs to work well. Combined with encoder initialization, LCNN-VAE improves over LSTM-LM from 334.9 to 332.1 in NLL and from 66.2 to 63.9 in PPL. Similar results for the sentiment data set are shown in Table 2b. LCNN-VAE improves over LSTM-LM from 362.7 to 359.1 in NLL and from 42.6 to 41.1 in PPL.




(a) Yahoo (b) Yelp

Figure 3: Visualizations of learned latent representations.

**Latent representation visualization:** In order to visualize the latent representation, we set the dimension of $\mathbf{z}$ to be 2 and plot the mean of posterior probability $q(\mathbf{z}|\mathbf{x})$, as shown in Figure 3. We can see distinct different characteristics of topic and sentiment representation. In Figure 3a, we can see that documents of different topics fall into different clusters, while in Figure 3b, documents of different ratings form a continuum, they lie continuously on the x-axis as the review rating increases.

| Model | ACCU | NLL (KL) |
|---|---|---|
| LSTM-VAE-Semi | 51.9 | 345.5 (9.3) |
| SCNN-VAE-Semi | **65.5** | 335.7 (10.4) |
| MCNN-VAE-Semi | 64.6 | 332.8 (7.2) |
| LCNN-VAE-Semi | 57.2 | **331.3** (2.7) |

Table 3: Semi-supervised VAE ablation results on Yahoo. We report both the NLL and classification accuracy of the test data. Accuracy is in percentage. Number of labeled samples is fixed to be 500.

### 3.4. Semi-supervised VAE results

Motivated by the success of VAEs for language modeling, we continue to explore VAEs for semi-supervised learning. Following that of (Kingma et al., 2014), we set the number of labeled samples to be 100, 500, 1000 and 2000 respectively.

**Ablation Study**: At first, we would like to explore the effect of different decoders for semi-supervised classification. We fix the number of labeled samples to be 500 and report both classification accuracy and NLL of the test set of Yahoo data set in Table. 5. We can see that SCNN-VAE-Semi has the best classification accuracy of 65.5. The accuracy decreases as we gradually increase the decoder contextual capacity. On the other hand, LCNN-VAE-Semi has the best NLL result. This classification accuracy and NLL trade off once again verifies our conjecture: with small contextual window size, the decoder is forced to use the encoder information, hence the latent representation is better

| Model | 100 | 500 | 1000 | 2000 |
|---|---|---|---|---|
| LSTM | 10.7 | 11.9 | 14.3 | 23.1 |
| LA-LSTM (Dai & Le, 2015) | 20.8 | 42.2 | 50.4 | 54.7 |
| LM-LSTM (Dai & Le, 2015) | 46.9 | 61.3 | 63.9 | 65.6 |
| SCNN-VAE-Semi | 55.4 | 65.6 | 66.0 | 65.8 |
| SCNN-VAE-Semi+init | **63.8** | **65.4** | **66.6** | **67.4** |

(a) Yahoo

| Model | 100 | 500 | 1000 | 2000 |
|---|---|---|---|---|
| LSTM | 22.6 | 25.4 | 27.9 | 29.9 |
| LA-LSTM (Dai & Le, 2015) | 35.2 | 46.4 | 49.8 | 52.2 |
| LM-LSTM (Dai & Le, 2015) | 46.9 | 54.1 | 57.2 | 57.7 |
| SCNN-VAE-Semi | 51.4 | 53.5 | 55.3 | 57.4 |
| SCNN-VAE-Semi+init | **52.6** | **57.3** | **58.9** | **59.8** |

(b) Yelp

Table 4: Semi-supervised VAE results on the test set, in percentage. LA-LSTM and LM-LSTM come from (Dai & Le, 2015), they denotes the LSTM is initialized with a sequence autoencoder and a language model.

learned.

Comparing the NLL results of Table 5 with that of Table 2a, we can see the NLL improves. The NLL of semi-supervised VAE improves over simple VAE from 337.8 to 335.7 for SCNN, from 336.2 to 332.8 for MCNN, and from 333.9 to 332.8 for LCNN. The improvement mainly comes from the KL divergence part, this indicates that better latent representations decrease the KL divergence, further improving the VAE results.

**Comparison with related methods**: We compare Semi-supervised VAE with the methods from (Dai & Le, 2015), which represent the previous state-of-the-art for semi-supervised sequence learning. Dai & Le (2015) pre-trains a classifier by initializing the parameters of a classifier with that of a language model or a sequence autoencoder. They find it improves the classification accuracy significantly. Since SCNN-VAE-Semi performs the best according to Table 5, we fix the decoder to be SCNN in this part. The detailed comparison is in Table 4. We can see that semi-supervised VAE performs better than LM-LSTM and LA-LSTM from (Dai & Le, 2015). We also initialize the encoder of the VAE with parameters from LM and find classification accuracy further improves. We also see the advantage of SCNN-VAE-Semi over LM-LSTM is greater when the number of labeled samples is smaller. The advantage decreases as we increase the number of labeled samples. When we set the number of labeled samples to be 25k, the SCNN-VAE-Semi achieves an accuracy of 70.4, which is similar to LM-LSTM with an accuracy of 70.5. Also, SCNN-VAE-Semi performs better on Yahoo data set than Yelp data set. For Yelp, SCNN-VAE-Semi is a little bit worse than LM-LSTM if the number of labeled samples is greater than 100, but becomes better when we initialize the encoder. Figure 3b explains this observation. It shows the documents are coupled together and are harder to classify. Also, the latent representation contains information other than sentiment, which may not be useful for classification.

### 3.5. Unsupervised clustering results

We also explored using the same framework for unsupervised clustering. We compare with the baselines that extract the feature with existing models and then run Gaussian Mixture Model (GMM) on these features. We find empirically that simply using the features does not perform well since the features are high dimensional. We run a PCA on these features, the dimension of PCA is selected from [8, 16, 32]. Since GMM can easily get stuck in poor local optimum, we run each model ten times and report the best result. We find directly optimizing $U(\mathbf{x})$ does not perform well for unsupervised clustering and we need to initialize the encoder with LSTM language model. The model only works well for Yahoo data set. This is potentially because Figure 3b shows that sentiment latent representations does not fall into clusters. $\gamma$ in Equation 5 is a sensitive parameter, we select it from the range between 0.5 and 1.5 with an interval of 0.1. We use the following evaluation protocol (Makhzani et al., 2015): after we finish training, for cluster $i$, we find out the validation sample $\mathbf{x}_n$ from cluster $i$ that has the best $q(y_i|\mathbf{x})$ and assign the label of $\mathbf{x}_n$ to all samples in cluster $i$. We then compute the test accuracy based on this assignment. The detailed results are in Table 5. We can see SCNN-VAE-Unsup + init performs better than other baselines. LSTM+GMM performs very bad probably because the feature dimension is 1024 and is too high for GMM, even though we already used PCA to reduce the dimension.

| Model | ACCU |
|---|---|
| LSTM + GMM | 25.8 |
| SCNN-VAE + GMM | 56.6 |
| SCNN-VAE + init + GMM | 57.0 |
| SCNN-VAE-Unsup + init | **59.9** |

Table 5: Unsupervised clustering results for Yahoo data set. We run each model 10 times and report the best results. LSTM+GMM means we extract the features from LSTM language model. SCNN-VAE + GMM means we use the mean of $q(\mathbf{z}|\mathbf{x})$ as the feature. SCNN-VAE + init + GMM means SCNN-VAE is trained with encoder initialization.

**Conditional text generation** With the semi-supervised VAE, we are able to generate text conditional on the label. Due to space limitation, we only show one example of

| | |
|---|---|
| **1 star** | the food was good but the service was horrible . took forever to get our food . we had to ask twice for our check after we got our food . will not return . |
| **2 star** | the food was good , but the service was terrible . took forever to get someone to take our drink order . had to ask 3 times to get the check . food was ok , nothing to write about . |
| **3 star** | came here for the first time last night . food was good . service was a little slow . food was just ok . |
| **4 star** | food was good , service was a little slow , but the food was pretty good . i had the grilled chicken sandwich and it was really good . will definitely be back ! |
| **5 star** | food was very good , service was fast and friendly . food was very good as well . will be back ! |

Table 6: Text generated by conditioning on sentiment label.

generated reviews conditioning on review rating in Table 6. For each group of generated text, we fix $\mathbf{z}$ and vary the label $\mathbf{y}$, while picking $\mathbf{x}$ via beam search with a beam size of 10.

## 4. Related work

Variational inference via the re-parameterization trick was initially proposed by (Kingma & Welling, 2013; Rezende et al., 2014) and since then, VAE has been widely adopted as generative model for images (Gregor et al., 2015; Yan et al., 2016; Salimans et al., 2015; Gregor et al., 2016; Hu et al., 2017b).

Our work is in line with previous works on combining variational inferences with text modeling (Bowman et al., 2015; Miao et al., 2016; Serban et al., 2016; Zhang et al., 2016; Hu et al., 2017a). (Bowman et al., 2015) is the first work to combine VAE with language model and they use LSTM as the decoder and find some negative results. On the other hand, (Miao et al., 2016) models text as bag of words, though improvement has been found, the model can not be used to generate text. Our work fills the gaps between them. (Serban et al., 2016; Zhang et al., 2016) applies variational inference to dialogue modeling and machine translation and found some improvement in terms of generated text quality, but no language modeling results are reported. (Chung et al., 2015; Bayer & Osendorfer, 2014; Fraccaro et al., 2016) embedded variational units in every step of a RNN, which is different from our model in using global latent variables to learn high level features.

Our use of CNN as decoder is inspired by recent success of PixelCNN model for images (van den Oord et al., 2016b), WaveNet for audios (van den Oord et al., 2016a), Video Pixel Network for video modeling (Kalchbrenner et al., 2016b) and ByteNet for machine translation (Kalchbrenner et al., 2016a). But in contrast to those works showing using a very deep architecture leads to better performance, CNN as decoder is used in our model to control the contextual capacity, leading to better performance.

Our work is closed related the recently proposed variational lossy autoencoder (Chen et al., 2016) which is used to predict image pixels. They find that conditioning on a smaller window of a pixels leads to better results with VAE, which is similar to our finding. Much (Rezende & Mohamed, 2015; Kingma et al., 2016; Chen et al., 2016) has been done to come up more powerful prior/posterior distribution representations with techniques such as normalizing flows. We treat this as one of our future works. This work is largely orthogonal and could be potentially combined with a more effective choice of decoder to yield additional gains.

There is much previous work exploring unsupervised sentence encodings, for example skip-thought vectors (Kiros et al., 2015), paragraph vectors (Le & Mikolov, 2014), and sequence autoencoders (Dai & Le, 2015). (Dai & Le, 2015) applies a pretrained model to semi-supervised classification and find significant gains, we use this as the baseline for our semi-supervised VAE.

## 5. Conclusion

We showed that by controlling the decoder's contextual capacity in VAE, we can improve performance on both language modeling and semi-supervised classification tasks by preventing a degenerate collapse of the training procedure. These results indicate that more carefully characterizing decoder capacity and understanding how it relates to common variational training procedures may represent important avenues for unlocking future unsupervised problems.

## References

Bayer, Justin and Osendorfer, Christian. Learning stochastic recurrent networks. *arXiv preprint arXiv:1411.7610*, 2014.

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

Chen, Xi, Kingma, Diederik P, Salimans, Tim, Duan, Yan, Dhariwal, Prafulla, Schulman, John, Sutskever, Ilya, and Abbeel, Pieter. Variational lossy autoencoder. *arXiv preprint arXiv:1611.02731*, 2016.

Chung, Junyoung, Kastner, Kyle, Dinh, Laurent, Goel, Kratarth, Courville, Aaron C, and Bengio, Yoshua. A recurrent latent variable model for sequential data. In *Advances in neural information processing systems*, pp. 2980–2988, 2015.

Dai, Andrew M and Le, Quoc V. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems*, pp. 3079–3087, 2015.

Fraccaro, Marco, Sønderby, Søren Kaae, Paquet, Ulrich, and Winther, Ole. Sequential neural models with stochastic layers. In *Advances in Neural Information Processing Systems*, pp. 2199–2207, 2016.

Gregor, Karol, Danihelka, Ivo, Graves, Alex, Rezende, Danilo Jimenez, and Wierstra, Daan. Draw: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*, 2015.

Gregor, Karol, Besse, Frederic, Rezende, Danilo Jimenez, Danihelka, Ivo, and Wierstra, Daan. Towards conceptual compression. In *Advances In Neural Information Processing Systems*, pp. 3549–3557, 2016.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Hu, Zhiting, Yang, Zichao, Liang, Xiaodan, Salakhutdinov, Ruslan, and Xing, Eric P. Controllable text generation. *arXiv preprint arXiv:1703.00955*, 2017a.

Hu, Zhiting, Yang, Zichao, Salakhutdinov, Ruslan, and Xing, Eric P. On unifying deep generative models. *arXiv preprint arXiv:1706.00550*, 2017b.

Jang, Eric, Gu, Shixiang, and Poole, Ben. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Kalchbrenner, Nal, Espeholt, Lasse, Simonyan, Karen, Oord, Aaron van den, Graves, Alex, and Kavukcuoglu, Koray. Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099*, 2016a.

Kalchbrenner, Nal, Oord, Aaron van den, Simonyan, Karen, Danihelka, Ivo, Vinyals, Oriol, Graves, Alex, and Kavukcuoglu, Koray. Video pixel networks. *arXiv preprint arXiv:1610.00527*, 2016b.

Kingma, Diederik and Ba, Jimmy. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

Kingma, Diederik P, Mohamed, Shakir, Rezende, Danilo Jimenez, and Welling, Max. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, pp. 3581–3589, 2014.

Kingma, Diederik P, Salimans, Tim, and Welling, Max. Improving variational inference with inverse autoregressive flow. *arXiv preprint arXiv:1606.04934*, 2016.

Kiros, Ryan, Zhu, Yukun, Salakhutdinov, Ruslan R, Zemel, Richard, Urtasun, Raquel, Torralba, Antonio, and Fidler, Sanja. Skip-thought vectors. In *Advances in neural information processing systems*, pp. 3294–3302, 2015.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Larochelle, Hugo and Lauly, Stanislas. A neural autoregressive topic model. In *Advances in Neural Information Processing Systems*, pp. 2708–2716, 2012.

Le, Quoc V and Mikolov, Tomas. Distributed representations of sentences and documents. In *ICML*, volume 14, pp. 1188–1196, 2014.

Maddison, Chris J, Mnih, Andriy, and Teh, Yee Whye. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

Makhzani, Alireza, Shlens, Jonathon, Jaitly, Navdeep, Goodfellow, Ian, and Frey, Brendan. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

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

Mikolov, Tomas, Karafiát, Martin, Burget, Lukas, Cernockỳ, Jan, and Khudanpur, Sanjeev. Recurrent neural network based language model. In *Interspeech*, volume 2, pp. 3, 2010.

Mnih, Andriy and Gregor, Karol. Neural variational inference and learning in belief networks. *arXiv preprint arXiv:1402.0030*, 2014.

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

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

Salimans, Tim, Kingma, Diederik P, Welling, Max, et al. Markov chain monte carlo and variational inference: Bridging the gap. In *ICML*, volume 37, pp. 1218–1226, 2015.

Serban, Iulian Vlad, Sordoni, Alessandro, Lowe, Ryan, Charlin, Laurent, Pineau, Joelle, Courville, Aaron, and Bengio, Yoshua. A hierarchical latent variable encoder-decoder model for generating dialogues. *arXiv preprint arXiv:1605.06069*, 2016.

Tang, Duyu, Qin, Bing, and Liu, Ting. Document modeling with gated recurrent neural network for sentiment classification. In *EMNLP*, pp. 1422–1432, 2015.

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

van den Oord, Aaron, Kalchbrenner, Nal, Espeholt, Lasse, Vinyals, Oriol, Graves, Alex, et al. Conditional image generation with pixelcnn decoders. In *Advances in Neural Information Processing Systems*, pp. 4790–4798, 2016b.

Yan, Xinchen, Yang, Jimei, Sohn, Kihyuk, and Lee, Honglak. Attribute2image: Conditional image generation from visual attributes. In *European Conference on Computer Vision*, pp. 776–791. Springer, 2016.

Yang, Zichao, Yang, Diyi, Dyer, Chris, He, Xiaodong, Smola, Alex, and Hovy, Eduard. Hierarchical attention networks for document classification. In *Proceedings of NAACL-HLT*, pp. 1480–1489, 2016.

Yu, Fisher and Koltun, Vladlen. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

Zhang, Biao, Xiong, Deyi, Su, Jinsong, Duan, Hong, and Zhang, Min. Variational neural machine translation. *arXiv preprint arXiv:1605.07869*, 2016.

Zhang, Xiang, Zhao, Junbo, and LeCun, Yann. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pp. 649–657, 2015.