Preventing Index Collapse in Discrete VAEs for Sentences

Asher Spector Jason Ren Tristan Yang

Abstract

We introduce a discrete latent variable model for sentence generation based on the Vector-Quantized Variational Autoencoder (VQ-VAE) introduced in Oord et al. (2017). To prevent the problem of *index collapse*, where usage of the discrete latents is limited to only a small number of values, we propose a modification to the VQ-VAE training scheme based on K-Means clustering. Empirically, we achieve superior performance in both index collapse, reconstruction perplexity, and language modeling perplexity compared to the training methods used in Oord et al. (2017) and Kaiser et al. (2018). Though we demonstrate the viability of discrete latent variables for text, we are unable to fully replicate the performance of continuous VAE's on sentences, which we hypothesize relates to the degeneracy of the variational family used by VQ-VAE.

1. Introduction

Language modeling, defined as learning the joint distribution of words in sentences, lies at the core of many NLP related tasks. Recent research, including most notably Devlin et al. (2018) and (Radford, 2018), has demonstrated that language models trained on enormous corpora of text can be fine-tuned to reach state of the art performance in almost every domain of NLP.

However, these huge language models lack interpretability, and informally, it is often difficult to tell whether these models have simply learned the biases in their training datasets. Moreover, many authors such as (Hu et al., 2017) have demonstrated interest in finder control over the types of text generated by language models. As a result, there has increased interest recently on latent variable language models, i.e. models which specify some relationship between the observed words in natural language and underlying "latent variables" which may control their distribution. (Among others, see Bowman et al. (2016), Hu et al. (2017), He et al. (2019)).

We propose using *discrete* latent variables to model text for two reasons. Firstly, we believe that the nature of language is inherently discrete; i.e. latent features such as tense or negation are discrete properties. Secondly, our model and training procedure avoids the issues of *index collapse* and *posterior collapse*, which occur when the latent variables are ignored in generation.

Although there has been previous work on avoiding index collapse for discrete latent variables, we find they do not perform as well as possible, especially for high-dimensional latent spaces. Our main contribution in this work is to show that our novel training scheme, which relies heavily on K-Means clustering updates, fully solves the issue of index collapse.

2. Background

2.1. Language Models

Most recent language models are autoregressive. If $x = x_{1...L}$ denotes a sentence of L words, this means that the likelihood is factorized as follows:

$$p(x) = \prod_{i=1}^{L} p(x_i | x_{1...i-1})$$

Language models are traditionally evaluated by the *perplexity* metric, which is the exponential of the negative log likelihood of the text under the model.

There is a plethora of language modeling data available, at least in English; we use the standard relatively small Penn Treebank dataset (Marcus et al., 1993).

2.2. Variational Autoencoders

We focus on the variational autoencoder (VAE) as the main type of latent variable model discussed here. A VAE consists of two main components. First, VAEs consist of a generative model which predicts the likelihood of observed words x in a sentence given some latent variables z, $p(x|z)$. However, inferring the likelihood $p(x)$ using only this generative model requires summing over the entire possible latent space:

$$p(x) = \mathbb{E}_{z \sim p(z)} p(x|z)$$

This sum or integral is usually intractable, and this intractability constitutes the main difficulty of training latent variable models. As a result, most latent variable models

instead optimize the variational lower bound or the ELBO:

$$\log p(x) \geq \mathbb{E}_{z \sim q(z|x)}[\log p(x|z)] + D_{\mathrm{KL}}(q(z|x) \| p(z))$$

for any arbitrary *variational distribution* q. The difference between the likelihood and the ELBO is minimized when the KL divergence is minimized between the posterior $p(z|x)$ and the variational distribution $q(z|x)$. This motivates the second component of the VAE, the learned inference network, which predicts the parameters of a distribution $q(z|x)$ as a deterministic function of the observations x.

Training discrete VAEs yields another problem, however, which is that if the z's are discrete, then the predictions of the inference network may not be differentiable with respect to its parameters. We discuss this problem further in future sections.

2.3. Posterior Collapse and Index Collapse

In a variational autoencoder, *posterior collapse* occurs when the variational distribution $q(z|x)$ is exactly equal to the prior $p(z)$, which minimizes the KL term in the ELBO. However, in this case the input distribution of z used to calculate $p(x|z)$ during training is independent of x, and thus the VAE does not learn to use the z to predict x. However, this still may achieve good performance if $p(x|z)$ is parameterized by a powerful neural network, which is often occurs; in our case this reduces to a normal non-variational language model.

Index collapse is a problem with discrete latent variables that occurs when $q(z|x)$ is supported only on a single small subset of the discrete latent space across all x. In this case, the discrete VAE only learns to use a small portion of the latents to compute $p(x|z)$, and most of the latent space is meaningless. This can also be reflected in the prior if it is learned (see section 5.1).

3. Related Work

3.1. Continuous Sentence VAE

There has been some recent work on continuous variational autoencoders. Initially, Bowman et al. (2016) showed that continuous VAEs approach the performance of simple language models for small datasets like the Penn Treebank. They trained their VAE using the ELBO objective, and their main innovation was to anneal the KL term in the ELBO slowly over time as to prevent the model from initially minimizing the KL term, thus preventing posterior collapse.

He et al. (2019) showed that aggressively training the inference network in a VAE also helps address the problem of posterior collapse, since with a better inference network, the decoder is less likely to get stuck in a local minimum during training where it ignores the inference network's outputs.

Although He et al. (2019) achieve better results, our work is most easily compared to Bowman et al. (2016), since we train on the same dataset (Penn Treebank) with extremely similar architectures. We obtain worse overall bounds on language modeling likelihood/perplexity than both (Bowman et al., 2016) and (He et al., 2019). However, this is to be expected since training discrete latent variable models is particularly challenging: we still think our training procedure represents an important incremental step in training discrete latent variable models.

3.2. Discrete Latent Variable Models

There has been much research on training discrete latent variable models. Most research tended to focus on sampling-based methods to minimize the ELBO (see Mnih & Gregor (2014), Mnih & Rezende (2016)), until Oord et al. (2017) applied the technique of *vector quantization* (VQ), which was the first technique that allowed discrete VAEs to approach the performance of continuous ones in certain domains, particularly vision.

Informally, VQ involves generating a set of continuous latent variables and discretizing them by doing a nearest neighbor search on a dictionary of discrete embeddings. Although this search induces gradients which are zero almost everywhere, Oord et al. (2017) solved this problem by using a straight-through gradient and tacking an extra term onto the loss function to minimize the L^2 norm between the dictionary embeddings and inference network outputs. VQ will be discussed in depth in Section 5.2.

VQ is quite effective. Oord et al. (2017) and Kaiser et al. (2018) both show that it substantially outperforms the Gumbel-Softmax sampling method introduced by (Jang et al., 2017), which approximates the categorical distribution with a continuous distribution with a temperature constant that converges to the categorical distribution, allowing the reparamtrization trick. Similarly, (Kaiser et al., 2018) show that VQ also outperforms other discretization techniques with straight-through gradients such as semantic hashing. They also first identify the problem of index collapse in VQ-VAE's, and propose addressing this by decomposing the encoder outputs and running VQ on each decomposed chunk. They call this method decomposed vector quantization (DVQ).

3.3. Discrete Latent Variables for Sentences

Despite the literature on continuous VAE's for sentences, there has been little work on discrete counterparts. Oord et al. (2017) do not evaluate their original VQ-VAE on text data. Kaiser et al. (2018) use VQ-VAE to train a "latent

transformer" for the purpose of speeding up the translation decoding process, but do not evaluate the performance of VQ-VAE as a language model. While Kaiser & Bengio (2018) also applies a discrete autoencoder to text, they do not use a *variational* autoencoder and thus their model is not a true generative language model. Consequently, they only report reconstruction perplexity. Our work is the first to our knowledge to consider VQ-VAE as a generative model for sentences.

4. Model

We model a sentence $x = x_{1...L}$ from dataset X as generated by a sequence $z = z_{1...\ell}$ of discrete latent variables $z_j \in [K]$. The latent sequence length ℓ is set to $\ell = \lceil L/C \rceil$ for some downsizing factor C.

The Variational Autoencoder introduces a variational distribution $q(z|x)$ and minimizes the ELBO:

$$\log p(x) \geq \mathbb{E}_{z\sim q(z|x)}[\log p(x|z)] + D_{\mathrm{KL}}(q(z|x)\|p(z)).$$

As in Oord et al. (2017), we restrict $q(z|x)$ to be from the family of *degenerate distributions*; i.e. distributions whose support is a single element. Let $z^{(x)} = \arg\max_z q(z|x)$ denote the latent corresponding to x. The ELBO then simplifies to:

$$\log p(x) \geq \log p(x|z^{(x)}) + \log p(z^{(x)}) \tag{4.1}$$

We note that this is equivalent to the inequality $p(x) \geq p(x, z^{(x)})$ which follows from the law of total probability, and is also identical to the IWAE likelihood bound from Burda et al. (2016) for degenerate q.

To evaluate (4.1), we factor the first term as

$$p(x|z^{(x)}) = \prod_{i=1}^{L} p(x_i|x_{<i}, z^{(x)}) \tag{4.2}$$

which we parameterize by an autoregressive generator network. Rather than explicitly specify the prior in advance, we also parameterize

$$p(z^{(x)}) = \prod_{j=1}^{\ell} p(z_j^{(x)}|z_{<j}^{(x)}) \tag{4.3}$$

by an autoregressive network that represents a *learned prior*, as is done in Oord et al. (2017).

5. Training Procedure

5.1. Overview

We focus on modifying the original VQ-VAE discretization update scheme to address index collapse, while preserving the rest of the VQ-VAE training procedure. Following Oord et al. (2017), we do not train the prior network jointly with the rest of the model. Rather, we first train the inference and generator networks to minimize the ELBO bound according to a *uniform* prior, using gradient descent and the discretization method of section 5.2. Under this formulation, the KL-divergence term in (4.1) is constant, and can be dropped from the loss. We then fix the inference network and train the prior network to minimize $D_{\mathrm{KL}}(q(z|x)\|p(z))$. According to the factorization (4.3), this is equivalent to training a language model with vocabulary in the latent space.

5.2. Discretization with K-Means Clustering

The VQ-VAE introduces a set of K latent embeddings $e_1, \ldots, e_K \in \mathbb{R}^d$ used to calculate both $q(z|x)$ and $p(x|z)$. Specifically, the inference network maps

$$x_{1...L} \mapsto \tilde{e}_{1...\ell}^{(x)} \in \mathbb{R}^{\ell d}$$

from which we define:

$$z_j^{(x)} = \arg\min_k \|\tilde{e}_j^{(x)} - e_k\|_2. \tag{5.1}$$

The generator network takes as inputs the embeddings:

$$e_{z^{(x)}} = (e_{z_1^{(x)}}, \ldots, e_{z_\ell^{(x)}}).$$

Because we cannot backpropogate through the $\arg\min$ in (5.1) to train the inference network parameters λ, we instead use the "straight-through" gradient approximation:

$$\frac{\partial \mathcal{L}}{\partial \lambda} \approx \frac{\partial \mathcal{L}}{\partial e_{z^{(x)}}} \frac{\partial \tilde{e}^{(x)}}{\partial \lambda}. \tag{5.2}$$

In order for this approximation to work well, we want to train the embeddings e_k to minimize the distances between $\tilde{e}^{(x)}$ and $e_{z^{(x)}}$. Oord et al. (2017) use gradient descent on the L^2 distance $\|\mathrm{sg}(\tilde{e}_j^{(x)}) - e_k\|$, where the stop-gradient operator sg denotes treating the inference network outputs $\tilde{e}^{(x)}$ as fixed constants and only differentiating with respect to the embeddings. Alternatively, Kaiser et al. (2018) and Roy et al. (2018) set e_k to be an exponential moving average of the inference network outputs assigned to it during training. However, as noted in Roy et al. (2018), minimizing $\|\tilde{e}_j^{(x)} - e_k\|$ across the dataset is equivalent to the *k-means clustering* objective for cluster centers $e_1, \ldots, e_K$ and data points

$$\left\{\tilde{e}_j^{(x)} : x_{1...L} \in X, j \in \left[\lceil L/C \rceil\right]\right\}.$$

We thus propose performing a full k-means clustering run before each epoch to determine the embeddings $e_1, \cdots, e_K$. This allows us to leverage the convergence guarantees provided by Lloyd's algorithm and specific initializations such as K-Means++ or k-means‖ (Bahmani et al., 2012). [1]

[1] We also hypothesize that, given the importance of initialization in k-means clustering, re-initialization of the embeddings in

Rather than compute the initial clusters from the outputs of inference network at random initialization, we first treat the inference and generator networks as the encoder/decoder of a continuous, non-variational autoencoder and pretrain them for a small number of epochs. Then, we compute initial clusters based on the outputs of the encoder and use this to initialize VQ-VAE. Empirically, we find that this training scheme significantly reduces index collapse compared to the Kaiser et al. (2018).

Finally, we follow Oord et al. (2017) in further forcing the inference network outputs to be close to the cluster centers by adding $\|sg(\tilde{e}^{(x)}) - e_{z^{(x)}}\|_2$ to the gradient descent loss.

5.3. Recap

To recap, the overall training procedure is:

1. Pretrain the inference and generator networks as the encoder and decoder of a continuous autoencoder, and use this to initialize K-Means clusters.
2. Jointly optimize the inference and generator networks using the VQ-VAE procedure on the modified ELBO bound using a uniform prior and K-Means updates.
3. Run the inference network over the entire training dataset to calculate the empirical $z^{(x)}$'s
4. Train the autoregressive prior as a language model on the empirical $z^{(x)}$'s

6. Network Architectures

We consider only the case where C is a power of 2. The inference network consists of $\log_2 C$ layers, where each layer is a bidirectional single-layer LSTM followed by a convolution with stride 2. We use hidden size 512 for the LSTMs and kernel size 7 for the convolutions. The last layer is a linear layer that maps to the latent embedding size to produce $\tilde{e}^{(x)}$.

To encourage usage of the latents, we use a weaker generator network. First we upsize $e_{z^{(x)}}$ by using convolutions of stride 1 that double the hidden dimension, which is then reshaped to double the sequence length; these also use kernel size 7. We then feed the upsized sequence into a single layer (unidirectional) LSTM with hidden size 512. We further encourage latent usage by randomly dropping out 10% of the words fed into the LSTM, requiring it to only rely on the latents in those cases, following (Bowman et al., 2016).

Finally, for the learned prior, we also use a single layer LSTM with hidden size 512.

addition to computing Lloyd's between epochs may slightly benefit the clustering objective as the $\tilde{e}$ move during training.

7. Results

7.1. Experimental Overview

We test three discretization training procedures, with identical architectures for the inference network, generator networks, and the autoregressive prior. The first implements the base VQ-VAE as described by Oord et al. (2017). The second modifies discretization using the Decomposed Vector-Quantization procedure described by Kaiser et al. (2018), and updates the embeddings using the exponential-moving-average procedure (EMA). We refer to these models as DVQ-EMA models. Lastly, our proposal uses vanilla VQ-VAE in the discretization step, but additionally uses K-Means to update the discrete embedding dictionary every epoch. All procedures involve pretraining the inference and generator networks without discretization for $1-2$ epochs, as we find it improves performance, but only the K-Means method uses the continuous pretraining to initialize the embeddings.

For each of the three model classes, we report three types of experimental results. First, during training, we monitor the L^2 norm between the discrete embeddings and the encoder outputs, as a lower L^2 norm will allow more accurate straight through gradients. Second, we report the final perplexities of the generative model ($p(x|z)$), the autoregressive prior ($p(x|z)$), as well as the overall bound on the likelihood $p(x)$. Lastly, we visualize and quantify the use of the latents for all three model classes after they have been fully trained.

We train on *sentences* from the Penn Treebank dataset, using a vocabulary size of $10,000$ and the default PyTorch tokenizer.

7.2. L^2 Norms

Below, we report the average L^2 distance between the discrete embedding over the entire training set, which we recall should be low to achieve accurate straight-through gradients. We can see that the K-Means model significantly outperforms the other procedures.

7.3. Perplexities

Below, we report the perplexities per sequence unit corresponding to both terms on the right hand side of (4.1), which we call reconstruction perplexity, and "KL/Prior perplexity" (equal to $\exp(-D_{\mathrm{KL}}(q(z|x)\|p(z)))$). Note that this correpsonds to perplexity per word for reconstruction perplexity, and the *C'th power* of perplexity per word for KL/Prior perplexity. Finally, we report the total perplexity of the language model under the calculation (7.1).

Large KL/Prior perplexity in Table 1 corresponds to a large KL term in the ELBO, which indicates the lack of posterior

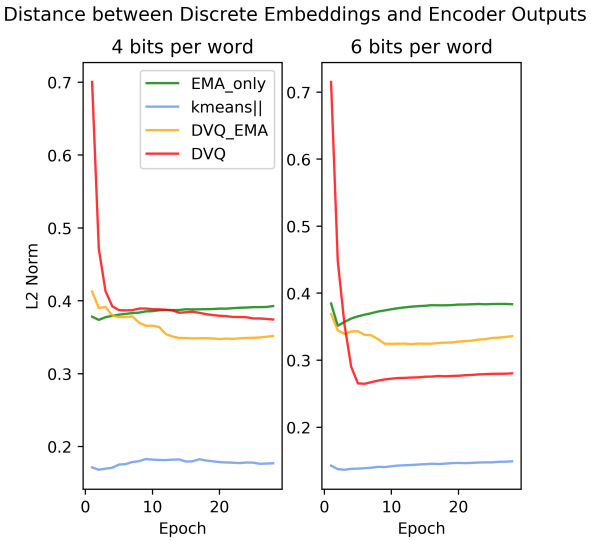

Figure 1. Distance between Discrete Embeddings and Encoder Outputs, K = 16 to 64

Training Method	Reconstr. ppl.	KL ppl.	ppl.
Base VQVAE	63.9	18.7	276.3
EMA + DVQ	38.8	21.2	178.4
Full K-Means	20.0	72.1	169.7

Table 1. K = 256 (8 bits/word), C = 2

collapse (see section 8.1).

7.4. Index Collapse

We also directly examine index collapse by considering the histogram:

$$\sum_{x \in X} q(z|x)$$

for the three training methods, as shown in figures 2 and 3. Because the prior $p(z)$ is fit to q, a histogram that is ≈ 0 for many values of z means that most of the latent space is not used by the generative model.

8. Discussion

8.1. Index Collapse

Figures 2 and 3 make clear that even using the EMA procedure, discrete VAE's trained using their procedure barely use more than a few latents. Indeed, for K = 1000, training with DVQ suffers from major index collapse while almost all latents (even beyond top 100) are used a good amount when training with our procedure. For K= 256, DVQ does not suffer from index collapse as much, but we still see our procedure do better in terms of using the latents. We note that index collapse was so bad in the base VQVAE training method that it made the visualization difficult. This is consistent with (Kaiser et al., 2018)'s observation that index collapse worsens with large K.

The initial loss presented by (Oord et al., 2017) helps explain why. Their loss, as briefly described in section 5.2, is written as

$$\mathcal{L} = \log p(x|e_{z^{(x)}}) + \|sg(\tilde{e}^{(x)}) - e_{z^{(x)}}\|_2$$

where the first term corresponds to the ELBO with a straight-through gradient, and the second term penalizes the distance between the encoder outputs and the discretized versions. However, note that under this loss, gradient updates only flow to discrete embeddings which are already the nearest neighbor of encoder outputs. As a result, any embeddings which already are closest to many encoder outputs get pulled even closer to those encoder outputs, and the other embeddings receive no gradient. In other words, under the vanilla VQ-VAE training procedure, the discrete embeddings get stuck in a "local minimum" where only one or two embeddings are used.

The K-Means update solves this problem precisely because the coordinate ascent procedure in K-Means (with proper initialization, as noted in (Bahmani et al., 2012)) is guaranteed to converge to within a constant factor of the global optimum in expectation, which necessitates usage of all the clusters. We see that the much lower L^2 norms in figure 3 correspond to improvements in reconstruction perplexity for our method in Table 1.

Furthermore, we note that the greater usage of index collapse corresponds to a larger KL term in the ELBO given the autoregressive prior p:

$$D_{\mathrm{KL}}(q(z|x)||p(z)) = p(z^{(x)})$$

which is also an indicator of good latent usage and lack of posterior collapse, as in (Bowman et al., 2016). (Here, since we use a degenerate q distribution, the perplexity of the autoregressive prior is equivalent to the exponentiated KL term in the ELBO between the inference network's predictions and the autoregressive prior.)

8.2. Language Modeling Perplexities

A higher KL term (autoregressive prior perplexity) is both a good sign and a bad sign—it corresponds to higher latent usage, which means that we have successfully trained a true latent variable model, but it also detracts from the final bound on the language modeling perplexity. Indeed, despite a reconstruction perplexity of 20, the full K-Means model only achieves a total perplexity of approximately 169 due to the KL term. We observe on experiments for higher values of C that the reconstruction and the KL/Prior

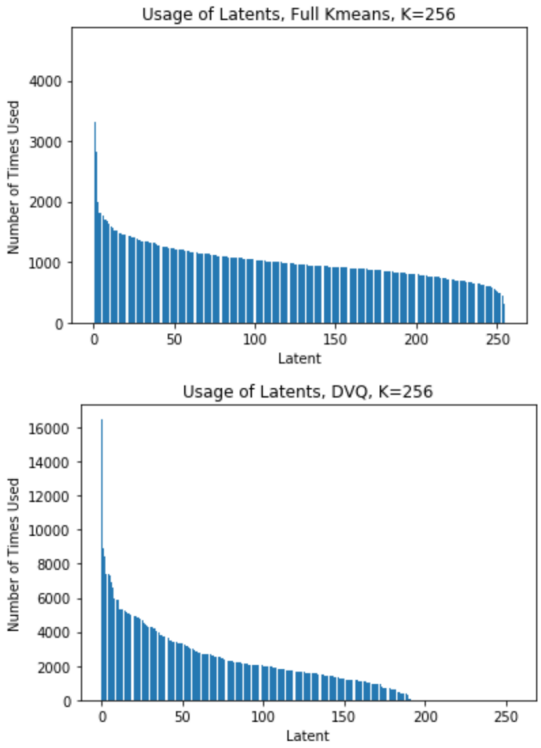

Figure 2. Latent Usage over 1 Epoch of Training, K=256

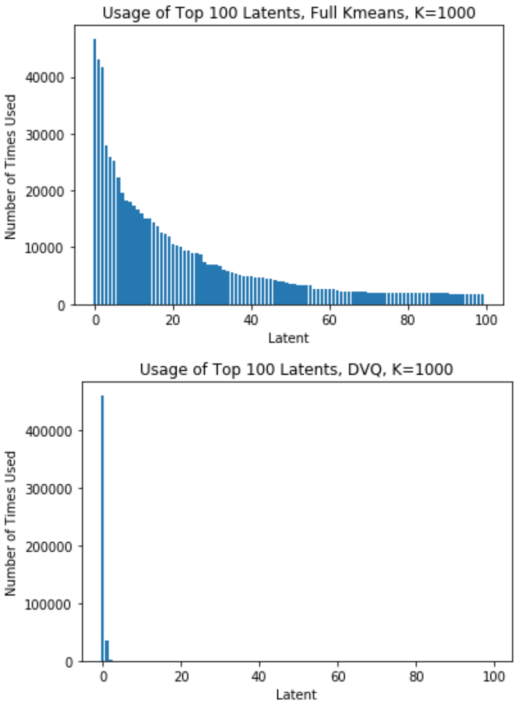

Figure 3. Latent Usage over 1 Epoch of Training, K=1000

perplexities increase. However, the contribution of the KL/Prior perplexity to the final perplexity *decreases* due to the C'th root compression factor noted in section 7.3. One natural explanation for higher KL/Prior perplexity for larger C may be greater independence between large sentence chunks obtained during increased downsizing that led to difficulty for training of the learned prior.

One direction for future work may be improving the inference procedure while preserving the overall probabilistic model and training procedure that we focused on for this work. Specifically, we observed in some preliminary tests that strengthening the bound in (4.1) by summing over a few more z's close to $z^{(x)}$ in hamming distance, while using the same generator network trained with VQ-VAE, significantly improved perplexity by up to ≈ 50 in some settings, and it seems like sampling more z's, as opposed to sampling from the degenerate q, would further improve performance. A related direction for exploration is usage of richer variational family for q during both training and inference.

9. Conclusion

We have introduced a new method of training discrete VAEs that improves reconstruction perplexity, perplexity, and greatly addresses the problem of index collapse. Although this is a step in the right direction, there is still quite a bit room to improve the final bounds on language modeling perplexities. If we find ways to lower perplexity to be competitive with continuous VAE variants or even non-latent models, we conjecture the learned discrete embeddings from the discrete VAE will serve as useful representations in a variety of NLP problems including low resource tasks, summarization, style transfer, and more.

10. Acknowledgements

We would like to give special thanks to Yoon Kim and Alexander Rush for their guidance and comments on this project. Thanks for a wonderful semester - we've learned so much!

References

Bahmani, Bahman, Moseley, Benjamin, Vattani, Andrea, Kumar, Ravi, and Vassilvitskii, Sergei. Scalable k-means++. *PVLDB*, 5:622–633, 2012.

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

Burda, Yuri, Grosse, Roger B., and Salakhutdinov, Rus-

lan R. Importance weighted autoencoders. *CoRR*, abs/1509.00519, 2016.

Devlin, Jacob, Chang, Ming-Wei, Lee, Kenton, and Toutanova, Kristina. Bert: Pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

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

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

Jang, Eric, Gu, Shixiang, and Poole, Ben. Categorical reparameterization with gumbel-softmax. *CoRR*, abs/1611.01144, 2017.

Kaiser, Lukasz and Bengio, Samy. Discrete autoencoders for sequence models. *CoRR*, abs/1801.09797, 2018.

Kaiser, Lukasz, Bengio, Samy, Roy, Aurko, Vaswani, Ashish, Parmar, Niki, Uszkoreit, Jakob, and Shazeer, Noam. Fast decoding in sequence models using discrete latent variables. In *ICML*, 2018.

Marcus, Mitchell P., Santorini, Beatrice, and Marcinkiewicz, Mary Ann. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19:313–330, 1993.

Mnih, Andriy and Gregor, Karol. Neural variational inference and learning in belief networks. 2014.

Mnih, Andriy and Rezende, Danilo J. Variational inference for monte carlo objectives, 2016.

Oord, Aäron Van Den, Vinyals, Oriol, and Kavukcuoglu, Koray. Neural discrete representation learning. In *NIPS*, 2017.

Radford, Alec. Improving language understanding by generative pre-training. 2018.

Roy, Aurko, Vaswani, Ashish, Neelakantan, Arvind, and Parmar, Niki. Theory and experiments on vector quantized autoencoders. *CoRR*, abs/1805.11063, 2018.