Controlled Text Generation by Learning Disentangled Latent Variables with a Semi-Supervised Approach

Michael Przystupa
Department of Computer Science
University of British Columbia
michael.przystupa@gmail.com

Marjan Albooyeh
Department of Computer Science
University of British Columbia
albooyeh@cs.ubc.ca

Abstract

Learning disentangled representations is important for specifying features in generative models for text. Adversarial training has previously been applied for text generation models in order to achieve such a disentangled representation, but the learned discriminator only helps during training and otherwise cannot be applied for other tasks. We consider an alternative view towards controlled text generation without adversarial training by specifying a generative and inference graphical model which can be implemented in a probabilistic programming framework. The learned model can then be utilized to jointly classify text for specified features as well as generate new text conditioned on specified attributes. We achieve this by optimizing a semi-supervised variational auto-encoder with an importance-weighted objective as well as the Reweighted Wake-Sleep algorithm. We try the Reweighted Wake-Sleep algorithm because it alleviates the need for continuous approximations of discrete random variables which introduce bias or variance on gradient updates. Our results suggest that the end-to-end VAE performs much better than the Reweighted Wake-Sleep algorithm in terms of loss and classification accuracy, but further investigations are necessary to improve the coherence of generated text.

1 Introduction

Deep generative text models have utility for a number of natural language processing tasks such as machine translation, boosting datasets, and dialogue systems. Being able to specify the desired features in generated sentences can be of particular use because it allows more control from a user perspective for downstream tasks. For controlled text generation, some inherent difficulties with applying deep generative models include 1) capturing different aspects of text with separate latent variables requires learning disentangled representations; 2) optimizing these deep learning models is challenging due to the presence of discrete random variables.

In this work, we propose generating controlled text from a probabilistic programming perspective. We apply the semi-supervised framework of [23] which learns disentangled representations with a variational auto-encoder (VAE) conditioned on specified features and learns latent representation of the data. The framework of [23] has only previously been applied for conditional image generation such as the "MNIST" or "Yale B" image datasets. The challenge with extending this framework to text is the need for discrete random variables to represent a sequence of words. The typical solution is to approximate discrete random variables with a continuous approximation, but these methods have been found to have high variance, or provide biased estimates [25, 18]. Maximizing the objective becomes difficult because better gradient approximations then require more samples which cannot scale well to larger datasets.

To investigate this problem, we also optimize our model with an alternative objective function which attempts to address the problem with discrete variables directly without a relaxed approximation.

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

Specifically, we apply the Reweighted Wake-Sleep algorithm [8] as a solution to decrease gradient estimator variance in the presence of discrete random variables. The advantage of the Reweighted Wake-Sleep algorithm is that it doesn't require samples from the discrete distribution in order to optimize the deep network by alternating between updating the inference and generative portions of the network.

In our experiments, we illustrate the advantage of a semi-supervised learning approach in jointly learning an effective classifier for sentiment analysis as well as providing a generative model for text. From our results, we find that despite addressing the discrete random variable problem more principally, the Reweighted Wake-Sleep algorithm performs much worse compared to the importance weighted semi-supervised objective.

2 Related Work

A common theme in recent works for generative language models is the difficultly in passing gradients through the network due to the discrete nature of text. [16] originally proposed applying REINFORCE style updates to mitigate this problem to train a GAN architecture. This approach has since then been applied to provide longer text sequences [19], and high-quality language descriptions [22].

An alternative approach to REINFORCE is utilizing the Gumbel softmax trick [21, 15] which introduces a temperature parameter to the softmax function and allows end-to-end training of networks. [14] showed this approach was applicable for discrete latent variables in a GAN structure. [20] also use the Gumbel softmax trick, but in combination with adversarial training with a VAE for controlled text generation for end-to-end training.

Often these continuous relaxation approaches come with high variance, or give bias approximations to the true discrete distribution making model training difficult. One solution to lower the variance, is introducing a baseline which lowers variance with introducing bias into the gradient. This has been successfully applied in other fields like reinforcement learning [3]. [17] provide a lower variance updates in the GAN framework which incorporates using a baseline in-conjunction with a novel re-framing of the GAN loss function specifically for discrete random variables.

However, researchers have approached text generation in other ways. [24] used a CNN VAE network in combination with RNNs to control semantic relations on a character level representation of text. In the work of [29] the authors leverage word embedding representations to apply GANs models to decode ciphertext.

Our work widely differs from previous work as we consider text generation from a probabilistic programming perspective. Incorporating the semi-supervised VAE approach has only otherwise been considered by [26] who only consider word level transduction where as we investigate sentence, or even discourse length generation.

3 Methodology

Our model aims to generate meaningful sentences conditioned on given semantic structures or attributes. We need to learn disentangled representations to ensure each of the latent variables captures only a particular semantic, while remaining independent of other features. Our goal is to have an explicit latent variable c that controls sentence semantic structure and another latent variable z, which captures other aspects of the text. In this model we assume labeled data is partially available for variable c, and otherwise utilize the unlabeled data for unsupervised learning by marginalizing out the variable c. We fully specify a directed graphical model to represent all the conditional and joint distributions between latent variables in the generative and inference networks which are trained with a semi-supervised VAE objective.

Consider an unsupervised dataset of sentences $D^{unsup} = \{x^1, ..., x^N\}$ and a supervised dataset of sentences with label c, $D^{sup} = \{(x^1, c^1), ..., (x^M, c^M)\}$. The proposed generative model in our VAE aims to train an encoder with parameters θ that generates sentences conditioned on z and c. The inference network in the VAE is modeled by a decoder with parameters ϕ to approximate posterior distribution $p_\theta(z, c|x)$ with $q_\phi(z, c|x)$.

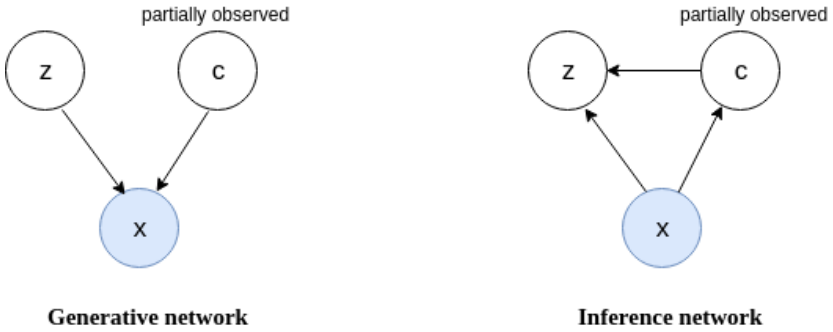

Figure 1: Generative and Inference networks

Figure 1 shows the graphical representation of these networks. In the generative model, the joint distribution is $p_\theta(x, z, c) = p_\theta(x|c, z)p(c)p(z)$ and we assume c and z are independent. In the inference model, in order to disentangle attribute c from latent z, we assume conditional dependencies are as $q_\phi(z, c|x) = q_{\phi_z}(z|c, x)q_{\phi_c}(c|x)$. Note that one of the advantages of this model over standard VAEs is that we can train the inference network in a supervised framework which helps us to regulate the inference network.

Both generative and inference networks will be jointly trained to optimize an objective that can be written as sum of two separate objectives based on supervised and unsupervised part of training [23]:

$$\mathcal{L}(\theta, \phi, D^{unsup}, D^{sup}) = \underbrace{\sum_{n=1}^{N} \mathcal{L}^{unsup}(\theta, \phi; x^n)}_{\text{unsupervised}} + \gamma \underbrace{\sum_{m=1}^{M} \mathcal{L}^{sup}(\theta, \phi; x^m, c^m)}_{\text{supervised}} \tag{1}$$

where γ is a constant parameter to control the effect of supervised term, depending on the relative sizes of the supervised and unsupervised datasets.

For the unsupervised part, the objective is to maximize the Expected Lower Bound (ELBO) $\mathcal{L}(\theta, \phi; D) \leq \log p_\theta(D^{unsup})$ using stochastic gradient ascent:

$$\begin{aligned}
\sum_{n=1}^{N} \mathcal{L}^{unsup}(\theta, \phi; x^n) &= \sum_{n=1}^{N} \mathbb{E}_{q_\phi(z,c|x)}\big[\log \frac{p_\theta(x^n, z, c)}{q_\phi(z, c|x^n)}\big] \\
&= \sum_{n=1}^{N} \mathbb{E}_{q_\phi(z,c|x)}\big[\log p_\theta(x^n|z, c) + \log p(z) + \log p(c) - \log q_\phi(z, c|x^n)\big] \\
&= \mathbb{E}_{q_\phi(z,c|x)}[\log p_\theta(x^n|z, c)] - KL(q_{\phi z}(z|c, x)||p(z)) - KL(q_{\phi c}(c|x)||p(c))
\end{aligned} \tag{2}$$

We can approximate $\mathbb{E}_{q_\phi(z,c|x)}[\log p_\theta(x^n|z, c)]$ with Monte Carlo sampling and the two KL divergence terms can be computed analytically.

The supervised term in Eq.(1) can be expressed as expectation over posterior plus a weighted component as in [12]:

$$\mathcal{L}^{sup}(\theta, \phi; x^m, c^m) = \mathbb{E}_{q_\phi(z|x^m,c^m)}\big[\log \frac{p_\theta(x^m, z, c^m)}{q_\phi(z, c^m|x^m)}\big] + \alpha \log q_\phi(c^m|x^m) \tag{3}$$

this expectation can be approximated with importance sampling by generating samples $z^{m,s} \sim q_\phi(z|x^m)$ and then calculating the weights with: $w^{m,s} = \frac{q_\phi(c^m, z^{m,s}|x^m)}{q_\phi(z^{m,s}|x^m)}$. To approximate the second term in Eq.(3), we maximize the lower bound with a Monte Carlo estimator and finally we obtain,

$$\mathcal{L}^{sup}(\theta, \phi; x^m, c^m) = \sum_{s=1}^{S} \frac{w^{m,s}}{\sum_{s'=1}^{S} w^{m.s'}} \log \frac{p_\theta(x^m, z^{m,s}, c^m)}{q_\phi(z^{m,s}, c^m|x^m)} + (1 + \alpha) \log w^{m,s} \tag{4}$$

We optimize the supervised and unsupervised objectives with respect to θ and ϕ to train the VAE. This objective allows us to incorporate multiple unlabeled and labeled datasets to control for multiple axis of variation in our generative model and could helps us produce generic and realistic sentences for multiple features.

In our model, sentence label c is a discrete random variable from a categorical distribution. As we mentioned earlier, gradient estimates in the presence of discrete random variables can have high variance, which causes slow convergence of optimization process. To solve this problem, when the label c is unavailable we marginalize it out in our distribution $q_\phi(c|x)$. Other variance reduction techniques have been proposed such as replacing the discrete random variables with a continuous approximation using the reparameterization trick to approximate gradient, or introducing control variates. All these techniques could help with decreasing the variance of gradient estimator. However, some practical issues including hyperparameter tuning and evaluation of all possible paths in case of branching, make these techniques less applicable to real world problems. Therefore, we will use Reweighted Wake-Sleep algorithm which uses another way of deriving the ELBO to the problem of taking gradient of a discrete random variables and consequently decreases the variance of gradient estimation.

4 Reweighted Wake-Sleep

The general idea of the Wake-Sleep algorithm is to update parameters of the generative and inference model separately by defining a single objective over parameters θ and ϕ [1]. The algorithm consists of two phases for updating parameters. In the wake phase, the inference network parameters are fixed and an observation is sampled from the training distribution. The observation is passed through the inference network to generate the latent values and together will be used to take a step of gradient ascent to maximize the likelihood over the joint distribution. This will update the generative model parameters θ to maximize the $ELBO(\theta, \phi, D)$ over dataset D. In the sleep phase, the generative parameters are fixed and inference network parameters are updated to minimize the KL divergence between the true posterior and approximation.

Reweighted Wake-Sleep [8] uses a tighter lower bound which is derived in [13] for Importance Weighted Autoencoders (IWAE). In our model this lower bound can be defined for both supervised and unsupervised objectives as follows:

$$\mathcal{L}_{\text{p}}^{unsup}(\theta, \phi, x^n) := \mathbb{E}_{Q_\phi(z^{(1:K)}, c^{(1:K)}|x^n)}\left[\log\left(\frac{1}{K}\sum_{k=1}^{K}\frac{p_\theta(x^n, z^{(k)}, c^{(k)})}{q_\phi(z^{(k)}, c^{(k)}|x^n)}\right)\right] \tag{5}$$

$$Q_\phi(z^{(1:K)}, c^{(1:K)}|x^n) = \prod_{k=1}^{K} q_\phi(z^{(k)}, c^{(k)}|x^n)$$

$$\mathcal{L}_{\text{p}}^{Sup}(\theta, \phi, x^m, c^m) := \mathbb{E}_{Q_\phi(z^{(1:K)}|x^m, c^m)}\left[\log\left(\frac{1}{K}\sum_{k=1}^{K}\frac{p_\theta(x^m, c^m, z^{(k)})}{q_\phi(z^{(k)}|x^m, c^m)}\right)\right] + \alpha \log q_\phi(c^m|x^m) \tag{6}$$

$$Q_\phi(z^{(1:K)}|x^m, c^m) = \prod_{k=1}^{K} q_\phi(z^{(k)}|x^m, c^m)$$

where k is the number of samples used for importance sampling. The key difference between Reweighted Wake-Sleep and IWAE is that it uses these objectives for updating θ and inference network parameters are updated by minimizing the KL divergence between generative model posterior and inference network.

$$\mathcal{L}_{\text{q}}^{unsup}(\theta, \phi, x^n) := -\mathbb{E}_{p(x^n)}\left[KL\big(p_\theta(z, c|x^n)||q_\phi(z, c|x^n)\big)\right] \tag{7}$$

$$\mathcal{L}_{\text{q}}^{sup}(\theta, \phi, x^m, c^m) := -\mathbb{E}_{p(x^m)p(c^m)}\left[KL\big(p_\theta(z|x^m, c^m)||q_\phi(z|x^m, c^m)\big)\right] \tag{8}$$

4.1 Wake phase θ update

In this phase, the inference model parameters are held fixed while the generative model parameters are updated for both supervised and unsupervised objectives.

$$\nabla_\theta \mathcal{L}_p^{unsup}(\theta, \phi, x^n) = \nabla_\theta \mathbb{E}_{Q_\phi(z^{1:K}, c^{1:K}|x^n)} \left[\log \left(\frac{1}{K} \sum_{k=1}^{K} \frac{p(z^{(k)}, c^{(k)}, x^n)}{q(z^{(k)}, c^{(k)}|x^n)} \right) \right] \tag{9}$$
$$= \mathbb{E}_{Q_\phi(z^{1:K}, c^{1:K}|x^n)} \left[\sum_{k=1}^{K} \hat{w}^k \nabla_\theta \log p(z^{(k)}, c^{(k)}, x^n) \right]$$

where $c^{(k)} \sim q_{\phi c}(c|x)$, $z^{(k)} \sim q_{\phi z}(z|x, c)$, $w^{(k)} = \frac{p_\theta(z^{(k)}, c^{(k)}, x^n)}{q_\phi(z^{(k)}, c^{(k)}|x^n)}$, and $\hat{w}^k = \frac{w^{(k)}}{\sum_{k'} w^{(k')}}$. Since $Q_\phi(z^{1:K}, c^{1:K}|x^n)$ is independent of θ, the gradient can be calculated without needing reparameterization trick and continuous approximation. The same approach can be used for updating θ in $\mathcal{L}_p^{sup}(\theta, \phi, x^m, c^m)$.

4.2 Wake phase ϕ update

In this phase, the inference network parameters are updated to maximize the negative KL divergence between the posteriors of both itself and the generative model

$$\nabla_\phi \mathcal{L}_q^{unsup}(\theta, \phi, x^n) = \nabla_\phi - \mathbb{E}_{p_\theta(x^n)} \left[KL\big(p_\theta(z, c|x^n)||q_\phi(z, c|x^n)\big) \right] \tag{10}$$
$$= \nabla_\phi \mathbb{E}_{p_\theta(z,c,x^n)} \left[\log q_\phi(z, c|x^n) - \log p_\theta(z, c|x^n) \right]$$
$$= \mathbb{E}_{p_\theta(z,c,x^n)} \left[\nabla_\phi \log q_\phi(z, c|x^n) \right]$$

This expectation term can be estimated using importance sampling with K particles sampled from the proposal distribution $q_\phi(z, c|x^n)$.

$$\nabla_\phi \mathcal{L}_q^{unsup}(\theta, \phi, x^n) = \sum_{k=1}^{K} \hat{w}^{(k)} \nabla_\phi \log q_\phi(z^{(k)}, c^{(k)}|x^n) \tag{11}$$

where $z^{(k)}$ and $\hat{w}^{(k)}$ are the same as for the θ update. Same approach can be used for the supervised objective. In this phase, the optimizer directly minimizes the expected KL divergence between true and approximate posteriors which results in a better approximation of posterior.

5 Experiments

After training our model, we evaluate the performances of the learned classifiers against a test set of examples, and report the final validation and test accuracy for both models. We then generate latent codes z and condition our generator model to produce positive and negative sentences to determine the quality of generated sequences.

5.1 Set-up

Network Architectures We built our deep generative model using the Pyro deep probabalistic programming framework [28]. For our encoder network, we use a convolutional neural network proposed in [10]. We chose a CNN because of previous work utilizing CNNs for text generation [24] as well as the success of CNNs in sequence generation [27]. We chose a recurrent neural network for our decoder. Specifically, we use a gated recurrent neural network [9] in order to pass the CNN encoding as the initial hidden state where as Long Short Term Memory units require an initial hidden and cell state [2]. To control sentiment, we concatenate the labels to the initial hidden state. We trained all our models with the Adam optimizer [11] and mini-batch sizes of 32.

Data Set We train our model on the Large Movie Review Dataset [6]. The labeled portions of data consist of 25000 labeled samples for the train and test sets, as well as an additional 50000 unlabeled sentences for semi-supervised learning. For our experiments, we pre-processed the text to remove

Model	IMDB Sentiment	
	Validation	Test
end-to-end VAE	0.593	0.71
Reweighted Wake-Sleep	0.546	0.674

Table 1: Sentiment accuracy of generated sentences on IMDB Sentiment dataset. End-to-end VAE is the first approach which updates generative and inference parameters in one step. Reweighted Wake Sleep uses different objectives for updating parameters separately in each step.

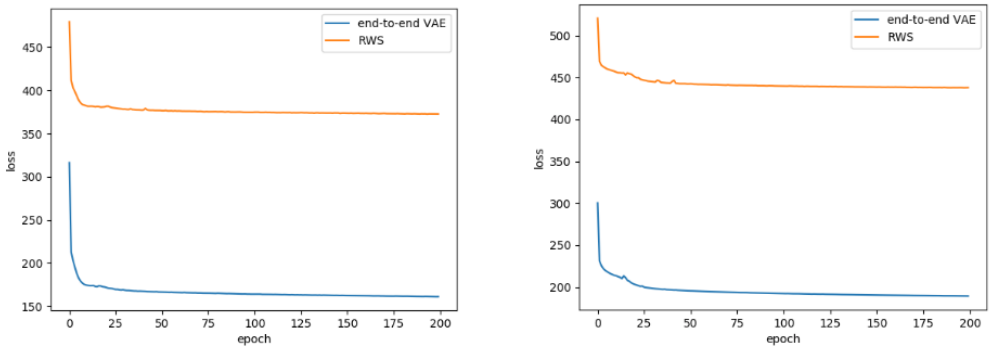

Figure 2: Comparing Average training loss of end-to-end VAE and Reweighted Wake-Sleep (RWS)

179 common stop words based on the NLTK toolkit default list [4]. We trained a Word2Vec [7] model from scratch with the Gensim library [5] using the full training set. Our Word2Vec model used a context window of size 10 and produced embeddings of dimension 300, and otherwise used the default configurations in Gensim. This process gave us an embedding matrix of 47191 words.

For training and testing our neural networks, we restricted ourselves to a maximum of 30 words per movie review. This left us with 1163 labeled sentences and 2246 unlabeled sentences for a total of 3409 training samples. We used 100 of our training samples for validation, and tested with 1272 which met our 30 word limit.

5.2 Discussion

Table 1 demonstrates the classification accuracy between the Reweighted Wake-Sleep algorithm and the end-to-end VAE approach. We see for the supervised setting, the end-to-end VAE approach works marginally better. However, from Figure 2 we clearly see that the Reweighted Wake-Sleep algorithm fails to achieve as good a loss compared to the end-to-end objective for both the supervised and unsupervised terms. One explanation for this may be the frequency in which we update the generative and inference networks during each epoch, but more gradient updates should happen per model before switching networks. Also, in Reweighted Wake-Sleep since generative and inference parameters are updated separately it becomes harder to achieve a good minima for both objectives.

positive sentence	**negative sentence**
marveling movie good shocking	winnie little acting
good vincent written nice clumsy factor	disenchanted ridiculous point wonder

Table 2: Part of sequences generated by end-to-end VAE conditioned on sentiments.

positive sentence	negative sentence
great actor talent also conveys great story	dismal love wistful worse poor
swedish terrific times since crew	astute hated phobias annabel disintegrates

Table 3: Part of sequences generated by Reweighted Wake Sleep approach conditioned on sentiments.

6 Conclusion

In this work we framed controlled text generation as a probabilistic program implementable in a probabilistic programming framework like Pyro. Our results leave much room for improvement, our current classifiers are not comparable to the state-of-the-art and our current generated sequences are quite noisy. Besides debugging our existing implementation, we are considering re-framing our graphical model to consider the sequential relationships of text with structures like Markov chains.

As future work, we want to extend our work to other NLP tasks such as semi-supervised machine translation, as well as incorporating multiple features of control such as style or entailment. If we can successfully generate coherent shorter sentences, we also would like to extend the model to longer sequences such as discourses or dialogues.

Also, claims of disentangled representations for text generation have so far have only been shown empirically to be true. [20] justified their claim by showing that without their independence constraint the meaning of sentences would change. However, [30] challenges these claims by proposing, in an adversarial learning scenario, the disentangled representations are actually entangled. They justify this by training new classifiers to recover the controlled latent variables from the other latent variables showing high accuracy for these models. We would like to investigate this entanglement phenomenon in our model to make sure our latent features are completely disentangled.

References

[1] Geoffrey E Hinton et al. “The" wake-sleep" algorithm for unsupervised neural networks”. In: *Science* 268.5214 (1995), pp. 1158–1161.

[2] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory”. In: *Neural Comput.* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. URL: http://dx.doi.org/10.1162/neco.1997.9.8.1735.

[3] Richard S. Sutton et al. “Policy Gradient Methods for Reinforcement Learning with Function Approximation.” In: *NIPS*. Vol. 99. 1999, pp. 1057–1063. URL: http://mlg.eng.cam.ac.uk/rowan/files/rl/PolicyGradientMatejAnnotations.pdf.

[4] Edward Loper and Steven Bird. “NLTK: The Natural Language Toolkit”. In: *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1*. ETMTNLP ’02. Philadelphia, Pennsylvania: Association for Computational Linguistics, 2002, pp. 63–70. DOI: 10.3115/1118108.1118117. URL: https://doi.org/10.3115/1118108.1118117.

[5] Radim Řehůřek and Petr Sojka. “Software Framework for Topic Modelling with Large Corpora”. English. In: *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. http://is.muni.cz/publication/884893/en. Valletta, Malta: ELRA, May 2010, pp. 45–50.

[6] Andrew L. Maas et al. “Learning Word Vectors for Sentiment Analysis”. In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, June 2011, pp. 142–150. URL: http://www.aclweb.org/anthology/P11-1015.

[7] Tomas Mikolov et al. “Distributed Representations of Words and Phrases and their Compositionality”. In: *CoRR* abs/1310.4546 (2013). arXiv: 1310.4546. URL: http://arxiv.org/abs/1310.4546.

[8] Jörg Bornschein and Yoshua Bengio. “Reweighted wake-sleep”. In: *arXiv preprint arXiv:1406.2751* (2014).

[9] Kyunghyun Cho et al. “Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation”. In: *CoRR* abs/1406.1078 (2014). arXiv: 1406.1078. URL: http://arxiv.org/abs/1406.1078.

[10] Yoon Kim. “Convolutional Neural Networks for Sentence Classification”. In: *CoRR* abs/1408.5882 (2014). arXiv: 1408.5882. URL: http://arxiv.org/abs/1408.5882.

[11] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *CoRR* abs/1412.6980 (2014). arXiv: 1412.6980. URL: http://arxiv.org/abs/1412.6980.

[12] Diederik P Kingma et al. “Semi-supervised learning with deep generative models”. In: *Advances in Neural Information Processing Systems*. 2014, pp. 3581–3589.

[13] Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. “Importance weighted autoencoders”. In: *arXiv preprint arXiv:1509.00519* (2015).

[14] Matt J. Kusner and José Miguel Hernández-Lobato. “GANS for Sequences of Discrete Elements with the Gumbel-softmax Distribution”. In: *CoRR* abs/1611.04051 (2016).

[15] Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. “The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables”. In: *CoRR* abs/1611.00712 (2016). arXiv: 1611.00712. URL: http://arxiv.org/abs/1611.00712.

[16] Lantao Yu et al. “SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient”. In: *CoRR* abs/1609.05473 (2016). arXiv: 1609.05473. URL: http://arxiv.org/abs/1609.05473.

[17] Tong Che et al. “Maximum-Likelihood Augmented Discrete Generative Adversarial Networks”. In: *CoRR* abs/1702.07983 (2017). arXiv: 1702.07983. URL: http://arxiv.org/abs/1702.07983.

[18] Will Grathwohl et al. “Backpropagation through the Void: Optimizing control variates for black-box gradient estimation”. In: *CoRR* abs/1711.00123 (2017).

[19] Jiaxian Guo et al. “Long Text Generation via Adversarial Training with Leaked Information”. In: *CoRR* abs/1709.08624 (2017). arXiv: 1709.08624. URL: http://arxiv.org/abs/1709.08624.

[20] Zhiting Hu et al. “Controllable Text Generation”. In: *CoRR* abs/1703.00955 (2017). arXiv: 1703.00955. URL: http://arxiv.org/abs/1703.00955.

[21] Eric Jang, Shixiang Gu, and Ben Poole. “Categorical Reparameterization with Gumbel-Softmax”. In: 2017. URL: https://arxiv.org/abs/1611.01144.

[22] Kevin Lin et al. “Adversarial Ranking for Language Generation”. In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., 2017, pp. 3155–3165. URL: http://papers.nips.cc/paper/6908-adversarial-ranking-for-language-generation.pdf.

[23] Siddharth Narayanaswamy et al. “Learning disentangled representations with semi-supervised deep generative models”. In: *Advances in Neural Information Processing Systems*. 2017, pp. 5925–5935.

[24] Stanislau Semeniuta, Aliaksei Severyn, and Erhardt Barth. “A Hybrid Convolutional Variational Autoencoder for Text Generation”. In: *CoRR* abs/1702.02390 (2017).

[25] George Tucker et al. “REBAR: Low-variance, unbiased gradient estimates for discrete latent variable models”. In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., 2017, pp. 2627–2636. URL: http://papers.nips.cc/paper/6856-rebar-low-variance-unbiased-gradient-estimates-for-discrete-latent-variable-models.pdf.

[26] Chunting Zhou and Graham Neubig. “Multi-space Variational Encoder-Decoders for Semi-supervised Labeled Sequence Transduction”. In: *CoRR* abs/1704.01691 (2017). arXiv: 1704.01691. URL: http://arxiv.org/abs/1704.01691.

[27] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. *Convolutional Sequence Modeling Revisited*. 2018. URL: https://openreview.net/forum?id=rk8wKk-R-.

[28] Eli Bingham et al. “Pyro: Deep Universal Probabilistic Programming”. In: *Journal of Machine Learning Research* (2018).

[29] Aidan N. Gomez et al. “Unsupervised Cipher Cracking Using Discrete GANs”. In: *CoRR* abs/1801.04883 (2018). arXiv: 1801.04883. URL: http://arxiv.org/abs/1801.04883.

[30] Anonymous. “Multiple-Attribute Text Rewriting”. In: *Submitted to International Conference on Learning Representations*. under review. 2019. URL: https://openreview.net/forum?id=H1g2NhC5KQ.