POLYPUBLIE

Adversarial Generation of Natural Language

Sandeep Subramanian, Sai Rajeswar Mudumba, Francis Dutil, Christopher J. Pal, Aaron Courville

Document en libre accès dans PolyPublie
Open Access document in PolyPublie

Date	2017
Type	Communication de conférence / Conference or Workshop Item Communication écrite/Paper
Version	Version officielle de l'éditeur / Published version
URL de PolyPublie **PolyPublie URL**	https://publications.polymtl.ca/54263/
Évaluation par les pairs **Peer reviewed**	Révisé par les pairs / Refereed
Conditions d'utilisation **Terms of use**	Creative Commons Attribution 4.0 International (CC BY)
Référence **Citation**	Subramanian, S., Rajeswar Mudumba, S., Dutil, F., Pal, C. J., & Courville, A. (août 2017). Adversarial Generation of Natural Language [Communication écrite]. 2nd Workshop on Representation Learning for NLP (RepL4NLP 2017), Vancouver, Colombie-Britannique, Canada. https://doi.org/10.18653/v1/w17-2629

Publication officielle
Official publication

Nom de la conférence **Conference name**	2nd Workshop on Representation Learning for NLP (RepL4NLP 2017)
Date et lieu **Date and location**	2017-08-03, Vancouver, Colombie-Britannique, Canada
Maison d'édition **Publisher**	Association for Computational Linguistics
URL officielle **Official URL**	https://doi.org/10.18653/v1/w17-2629
Mention légale **Legal notice**	ACL materials are Copyright © 1963–2026 ACL; other materials are copyrighted by their respective copyright holders. Materials prior to 2016 here are licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 3.0 International License. Permission is granted to make copies for the purposes of teaching and research. Materials published in or after 2016 are licensed on a Creative Commons Attribution 4.0 International License.

Ce fichier a été téléchargé à partir de PolyPublie, le dépôt institutionnel de Polytechnique Montréal
This file has been downloaded from PolyPublie, the institutional repository of Polytechnique Montréal
https://publications.polymtl.ca

Adversarial Generation of Natural Language

Sandeep Subramanian[♠*] **Sai Rajeswar**[♠*] **Francis Dutil**[♠]
Christopher Pal[♣♠] **Aaron Courville**[♠†]
[♠]MILA, Université de Montréal [♣]École Polytechnique de Montréal [†]CIFAR Fellow
{sandeep.subramanian.1,sai.rajeswar.mudumba,aaron.courville}@umontreal.ca,
frdutil@gmail.com, christopher.pal@polymtl.ca

Abstract

Generative Adversarial Networks (GANs) have gathered a lot of attention from the computer vision community, yielding impressive results for image generation. Advances in the adversarial generation of natural language from noise however are not commensurate with the progress made in generating images, and still lag far behind likelihood based methods. In this paper, we take a step towards generating natural language with a GAN objective alone. We introduce a simple baseline that addresses the discrete output space problem without relying on gradient estimators and show that it is able to achieve state-of-the-art results on a Chinese poem generation dataset. We present quantitative results on generating sentences from context-free and probabilistic context-free grammars, and qualitative language modeling results. A conditional version is also described that can generate sequences conditioned on sentence characteristics.

1 Introduction

Deep neural networks have recently enjoyed some success at modeling natural language (Mikolov et al., 2010; Zaremba et al., 2014; Kim et al., 2015). Typically, recurrent and convolutional language models are trained to maximize the likelihood of observing a word or character given the previous observations in the sequence $P(w_1 \dots w_n) = p(w_1) \prod_{i=2}^{n} P(w_i | w_1 \dots w_{i-1})$. These models are commonly trained using a technique called *teacher forcing* (Williams and Zipser, 1989) where the inputs to the network are fixed and the model is trained to predict only the next item in the sequence given all previous observations. This corresponds to maximum-likelihood training of these models. However this one-step ahead prediction during training makes the model prone to *exposure bias* (Ranzato et al., 2015; Bengio et al., 2015). Exposure bias occurs when a model is only trained conditioned on ground-truth contexts and is not exposed to its own errors (Wiseman and Rush, 2016). An important consequence to exposure bias is that generated sequences can degenerate as small errors accumulate. Many important problems in NLP such as machine translation and abstractive summarization are trained via a maximum-likelihood training objective (Bahdanau et al., 2014; Rush et al., 2015), but require the generation of extended sequences and are evaluated based on sequence-level metrics such as BLEU (Papineni et al., 2002) and ROUGE (Lin, 2004).

One possible direction towards incorporating a sequence-level training objective is to use Generative Adversarial Networks (GANs) (Goodfellow et al., 2014). While GANs have yielded impressive results for modeling images (Radford et al., 2015; Dumoulin et al., 2016), advances in their use for natural language generation has lagged behind. Some progress has been made recently in incorporating a GAN objective in sequence modeling problems including natural language generation. Lamb et al. (2016) use an adversarial criterion to match the hidden state dynamics of a teacher forced recurrent neural network (RNN) and one that samples from its own output distribution across multiple time steps. Unlike the approach in Lamb et al. (2016), sequence GANs (Yu et al., 2016) and maximum-likelihood augmented GANs (Che et al., 2017) use an adversarial loss at outputs of an RNN. Using a GAN at the outputs of an RNN however isn't trivial since sampling from these outputs to feed to the discrimi-

[♠*]Indicates first authors. Ordering determined by coin flip.

Proceedings of the 2nd Workshop on Representation Learning for NLP, pages 241–251,
Vancouver, Canada, August 3, 2017. ©2017 Association for Computational Linguistics

nator is a non-differentiable operation. As a result gradients cannot propagate to the generator from the discriminator. Yu et al. (2016) use policy gradient to estimate the generator's gradient and (Che et al., 2017) present an importance sampling based technique. Other alternatives include REINFORCE (Williams, 1992), the use of a Gumbel softmax (Jang et al., 2016) and the straighthrough estimator (Bengio et al., 2013) among others.

In this work, we address the discrete output space problem by simply forcing the discriminator to operate on continuous valued output distributions. The discriminator sees a sequence of probabilities over every token in the vocabulary from the generator and a sequence of 1-hot vectors from the true data distribution as in Fig. 1. This technique is identical to that proposed by Gulrajani et al. (2017), which is parallel work to this. In this paper we provide a more complete empirical investigation of this approach to applying GANs to discrete output spaces. We present results using recurrent as well as convolutional architectures on three language modeling datasets of different sizes at the word and character-level. We also present quantitative results on generating sentences that adhere to a simple context-free grammar (CFG), and a richer probabilistic context-free grammar (PCFG). We compare our method to previous works that use a GAN objective to generate natural language, on a Chinese poetry generation dataset. In addition, we present a conditional GAN (Mirza and Osindero, 2014) that generates sentences conditioned on sentiment and questions.

2 Generative Adversarial Networks

GANs (Goodfellow et al., 2014) are a general framework used in training generative models by formulating the learning process as a two player minimax game as formulated in the equation below. A generator network G tries to generate samples that are as close as possible to the true data distribution $P(x)$ of interest from a fixed noise distribution $P(z)$. We will refer to the samples produced by the generator as $G(z)$. A discriminator network is then trained to distinguish between $G(z)$ and samples from the true data distribution $P(x)$ while the generator network is trained using gradient signals sent by the discriminator by minimizing $\log(1 - D(G(z)))$. Goodfellow et al. (2014) have shown that, with respect to an optimal discriminator, the minimax formulation can be shown to minimize the Jensen Shannon Divergence (JSD) between the generator's output distribution and the true data distribution.

$$\min_G \max_D V(D,G) = \mathop{\mathbb{E}}_{x\sim P(x)}[\log D(x)] + \mathop{\mathbb{E}}_{z\sim P(z)}[\log(1 - D(G(z)))]$$

However, in practice, the generator is trained to maximize $\log(D(G(z)))$ instead, since it provides stronger gradients in the early stages of learning (Goodfellow et al., 2014).

GANs have been reported to be notoriously hard to train in practice (Arjovsky and Bottou, 2017) and several techniques have been proposed to alleviate some of the complexities involved in getting them to work including modified objective functions and regularization (Salimans et al., 2016; Arjovsky et al., 2017; Mao et al., 2016; Gulrajani et al., 2017). We discuss some of these problems in the following subsection.

Nowozin et al. (2016) show that it is possible to train GANs with a variety of f-divergence measures besides JSD. Wasserstein GANs (WGANs) (Arjovsky et al., 2017) minimize the earth mover's distance or Wasserstein distance, while Least Squared GANs (LSGANs) (Mao et al., 2016) modifies replaces the log loss with an L2 loss. WGAN-GP (Gulrajani et al., 2017) incorporate a gradient penalty term on the discriminator's loss in the WGAN objective which acts as a regularizer. In this work, we will compare some of these objectives in the context of natural language generation.

2.1 Importance of Wasserstein GANs

Arjovsky and Bottou (2017) argue that part of the problem in training regular GANs is that it seeks to minimize the JSD between the $G(z)$ and $P(x)$. When the generator is trying to optimized $log(1 - D(G(z)))$, the gradients that it receives vanish as the discriminator is trained to optimality. The authors also show that when trying to optimize the more practical alternative, $-log(D(G(z)))$, the generator might not suffer from vanishing gradients but receives unstable training signals. It is also important to consider the fact that highly structured data like images and language lie in low-dimensional manifolds (as is evident by studying their principal components). Wasserstein GANs (Arjovsky et al., 2017) overcome some of the problems in regular GAN train-

ing by providing a softer metric to compare the distributions lying in low dimensional manifolds. A key contribution of this work was identifying the importance of a lipschitz constraint which is achieved by clamping the weights of the discriminator to lie in a fixed interval. The lipschitz constraint and training the discriminator multiple times for every generator gradient update creates a strong learning signal for the generator.

Gulrajani et al. (2017) present an alternative to weight clamping that they call a gradient penalty to enforce lipschitzness since model performance was reported to be highly sensitive to the clamping hyperparameters. They add the following penalty to the discriminator training objective - $(||\nabla_{G(z)} D(G(z))||_2 - 1)^2$. A potential concern regarding our strategy to train our discriminator to distinguish between sequence of 1-hot vectors from the true data distribution and a sequence of probabilities from the generator is that the discriminator can easily exploit the sparsity in the 1-hot vectors to reach optimality. However, Wasserstein distance with a lipschitz constraint / gradient penalty provides good gradients even under an optimal discriminator and so isn't a problem for us in practice. Even though it is possible to extract some performance from a regular GAN objective with the gradient penalty (as we show in one of our experiments), WGANs still provide better gradients to the generator since the discriminator doesn't saturate often.

3 Model architecture

Let $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ be the input to our generator network G from which we will attempt to generate natural language. For implementation convenience, the sample $\mathbf{z}$ is of shape $n \times d$ where n is the length of sequence and d is a fixed length dimension of the noise vector at each time step. The generator then transforms $\mathbf{z}$ into a sequence of probability distributions over the vocabulary $G(z)$ of size $n \times k$ where k is the size of our true data distribution's vocabulary. The discriminator network D is provided with fake samples $G(z)$ and samples from the true data distribution $P(x)$. Samples from the true distribution are provided as a sequence of 1-hot vectors with each vector serving as an indicator of the observed word in the sample. As described in section 2, the discriminator is trained to discriminate between real and fake samples and the generator is trained to fool the discriminator as in Fig. 1.

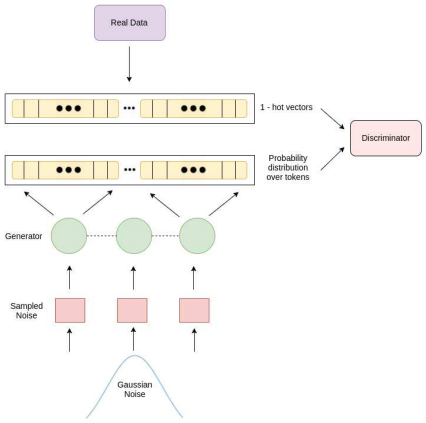


Figure 1: Model architecture

We investigate recurrent architectures as in (Lamb et al., 2016; Yu et al., 2016; Che et al., 2017) and convolutional architectures in both the generator as well as the discriminator. The following subsections detail our architectures.

3.1 Recurrent Models

Recurrent Neural Networks (RNNs), particularly Long short-term memory networks (LSTMs) (Hochreiter and Schmidhuber, 1997) and Gated Recurrent Networks (Cho et al., 2014) are powerful models that have been successful at modeling sequential data (Graves and Schmidhuber, 2009; Mikolov et al., 2010). They transform a sequence of input vectors $\mathbf{x} = x_1 \dots x_n$ into a sequence of hidden states $\mathbf{h} = h_1 \dots h_n$ where each hidden state maintains a summary of the input up until then. RNN language models are autoregressive in nature since the input to the network at time t depends on the output at time $t-1$. However, in the context of generating sequences from noise, the inputs are pre-determined and there is no direct correspondence between the output at time $t-1$ and the input at time t this fundamentally changes the auto-regressiveness of the RNN. The RNN does however carry forward information about its output at time t through subsequent time steps via its hidden states $\mathbf{h}$ as evident from its recurrent transition function. In order to incorporate an explicit dependence between subsequent RNN outputs, we add a peephole connection between the *output* probability distribution $\mathbf{y_{t-1}}$ at time $t-1$ and the hidden state $\mathbf{h_t}$ at time t as show in the LSTM equations below. Typical RNN lan-

guage models have a shared affine transformation matrix $\mathbf{W_{out}}$ that is shared across time all steps that projects the hidden state vector to a vector of the same size as the target vocabulary to generate a sequence of outputs $\mathbf{y} = y_1 \dots y_t$. Subsequently a softmax function is applied to each vector to turn it into a probability distribution over the vocabulary.

$$\mathbf{y}_t = \text{softmax}(\mathbf{W}_{out}\mathbf{h}_t + \mathbf{b}_{out}),$$

During inference, an output is sampled from the softmax distribution and becomes the input at the subsequent time step. While training the inputs are pre-determined. In all of our models, we perform greedy decoding where we always pick $\text{argmax}\, y_t$. When using the LSTM as a discriminator we use a simple binary logistic regression layer on the last hidden state h_n to determine the probability of the sample being from the generator's data distribution or from the real data distribution. $P(real) = \sigma(\mathbf{W}_{pred}\mathbf{h}_n + \mathbf{b}_{pred})$.

The LSTM update equations with an output peephole are :

$$\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}_{xi}\mathbf{x}_t + \mathbf{W}_{hi}\mathbf{h}_{t-1} + \mathbf{W}_{pi}\mathbf{y}_{t-1} + \mathbf{b}_i) \\
\mathbf{f}_t &= \sigma(\mathbf{W}_{xf}\mathbf{x}_t + \mathbf{W}_{hf}\mathbf{h}_{t-1} + \mathbf{W}_{pf}\mathbf{y}_{t-1} + \mathbf{b}_f) \\
\mathbf{o}_t &= \sigma(\mathbf{W}_{xo}\mathbf{x}_t + \mathbf{W}_{ho}\mathbf{h}_{t-1} + \mathbf{W}_{po}\mathbf{y}_{t-1} + \mathbf{b}_o) \\
\mathbf{c}_t &= \tanh(\mathbf{W}_{xc}\mathbf{x}_t + \mathbf{W}_{hc}\mathbf{h}_{t-1} + \mathbf{W}_{pc}\mathbf{y}_{t-1} + \mathbf{b}_c) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{c}_t \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t),
\end{aligned}$$

where σ is the element-wise sigmoid function, $\odot$ is the hadamard product, $\tanh$ is the element-wise $\tanh$ function. $\mathbf{W}_\cdot$ and $\mathbf{b}_\cdot$ are learn-able parameters of the model and $\mathbf{i_t}$, $\mathbf{f_t}$, $\mathbf{o_t}$ and $\mathbf{c_t}$ constitute the input, forget, output and cell states of the LSTM respectively.

3.2 Convolutional Models

Convolutional neural networks (CNNs) have also shown promise at modeling sequential data using 1-dimensional convolutions (Dauphin et al., 2016; Zhang et al., 2015). Convolution filters are convolved across time and the input dimensions are treated as channels. In this work, we explore convolutional generators and discriminators with residual connections (He et al., 2016).

Gulrajani et al. (2017) use a convolutional model for both the generator and discriminator. The generator consists of 5 residual blocks with 2 1-D convolutional layers each. A final 1-D convolution layer transforms the output of the residual blocks into a sequence of un-normalized vectors for each element in the input sequence (noise). These vectors are then normalized using the softmax function. All convolutions are 'same' convolutions with a stride of 1 followed by batch-normalization (Ioffe and Szegedy, 2015) and the ReLU (Nair and Hinton, 2010; Glorot et al., 2011) activation function without any pooling so as to preserve the shape of the input. The discriminator architecture is identical to that of the generator with the final output having a single output channel.

3.3 Curriculum Learning

In likelihood based training of generative language models, models are only trained to make one-step ahead predictions and as a result it is possible to train these models on relatively long sequences even in the initial stages of training. However, in our adversarial formulation, our generator is encouraged to generate entire sequences that match the true data distribution without explicit supervision at each step of the generation process. As a way to provide training signals of incremental difficulty, we use curriculum learning (Bengio et al., 2009) and train our generator to produce sequences of gradually increasing lengths as training progresses.

4 Experiments & Data

GAN based methods have often been critiqued for lacking a concrete evaluation strategy (Salimans et al., 2016), however recent work (Wu et al., 2016) uses an annealed importance based technique to overcome this problem.

In the context of generating natural language, it is possible to come up with a simpler approach to evaluate compute the likelihoods of generated samples. We synthesize a data generating distribution under which we can compute likelihoods in a tractable manner. We propose a simple evaluation strategy for evaluating adversarial methods of generating natural language by constructing a data generating distribution from a CFG or P−CFG. It is possible to determine if a sample belongs to the CFG or the probability of a sample under a P−CFG by using a constituency parser that is provided with all of the productions in a grammar. Yu et al. (2016) also present a simple idea to estimate the likelihood of generated samples by using a randomly initialized LSTM as their data gener-

ating distribution. While this is a viable strategy to evaluate generative models of language, a randomly initialized LSTM provides little visibility into the complexity of the data distribution itself and presents no obvious way to increase its complexity. CFGs and PCFGs however, provide explicit control of the complexity via their productions. They can also be learned via grammar induction (Brill, 1993) on large treebanks of natural language and so the data generating distribution is not synthetic as in (Yu et al., 2016).

Typical language models are evaluated by measuring the likelihood of samples from the true data distribution under the model. However, with GANs it is impossible to measure likelihoods under the model itself and so we measure the likelihood of the model's samples under the true data distribution instead.

We divide our experiments into four categories:

- Generating language that belongs to a toy CFG and an induced PCFG from the Penn Treebank (Marcus et al., 1993).
- Chinese poetry generation with comparisons to (Yu et al., 2016) and (Che et al., 2017).
- Generated samples from a dataset consisting of simple English sentences, the 1-billion-word and Penn Treebank datasets.
- Conditional GANs that generate sentences conditioned on certain sentence attributes such as sentiment and questions.

4.1 Simple CFG

We use a simple and publicly available CFG[1] that contains 248 productions. We then generate two sets of data from this CFG - one consisting of samples of length 5 and another of length 11. Each set contains 100,000 samples selected at random from the CFG. The first set has a vocabulary of 36 tokens while the second 45 tokens. We evaluate our models on this task by measuring the fraction of generated samples that satisfy the rules of the grammar and also measure the diversity in our generated samples. We do this by generating 1,280 samples from noise and computing the fraction of those that are valid under our grammar using the Earley parsing algorithm (Earley, 1970). In order to measure sample diversity, we simply the count the number of unique samples; while this assumes that all samples are orthogonal it still serves as a proxy measure of the entropy. We compare various generator, discriminator and GAN objectives on this problem.

4.2 Penn Treebank PCFG

To construct a more challenging problem than a simple CFG, we use sections 0-21 of the WSJ subsection of the Penn Treebank to induce a PCFG using simple count statistics of all productions.

$$P(A \rightarrow BC) = \frac{count(A \rightarrow BC)}{count(A \rightarrow *)}$$

We train our model on all sentences in the treebank and restrict the output vocabulary to the top 2,000 most frequently occurring words. We evaluate our models on this task by measuring the likelihood of a sample using a Viterbi chart parser (Klein and Manning, 2003). While such a measure mostly captures the grammaticality of a sentence, it is still a reasonable proxy of sample quality.

4.3 Chinese Poetry

Zhang and Lapata (2014) present a dataset of Chinese poems that were used to evaluate adversarial training methods for natural language in (Yu et al., 2016) and (Che et al., 2017). The dataset consists of 4-line poems with a variable number of characters in each line. We treat each line in a poem as a training example and use lines of length 5 (poem-5) and 7 (poem-7) with the train/validation/test split[2] specified in (Che et al., 2017). We use BLEU-2 and BLEU-3 to measure model performance on this task. Since there is no obvious "target" for each generated sentence, both works report corpus-level BLEU measures using the entire test set as the reference.

4.4 Language Generation

We generate language from three different datasets of varying sizes and complexity. A dataset comprising simple English sentences[3] which we will henceforth refer to as CMU−SE, the version of the Penn Treebank commonly used in language modeling experiments (Zaremba et al., 2014) and the Google 1-billion word dataset (Chelba et al.,

[1] http://www.cs.jhu.edu/~jason/465/hw-grammar/extra-grammars/holygrail

[2] http://homepages.inf.ed.ac.uk/mlap/Data/EMNLP14/

[3] https://github.com/clab/sp2016.11-731/tree/master/hw4/data

2013). We perform experiments at generating language at the word as well as character-level. The CMU−SE dataset consists of 44,016 sentences with a vocabulary of 3,122 words, while the Penn Treebank consists of 42,068 sentences with a vocabulary of 10,000 words. We use a random subset of 3 million sentences from the 1-billion word dataset and constrain our vocabulary to the top 30,000 most frequently occurring words. We use a curriculum learning strategy in all of our LSTM models (with and without the output peephole connection) that starts training on sentences of length 5 at the word level and 13 for characters and increases the sequence length by 1 after a fixed number of epochs based on the size of the data. Convolutional methods in (Gulrajani et al., 2017) are able to generate long sequences even without a curriculum, however we found it was critical in generating long sequences with an LSTM.

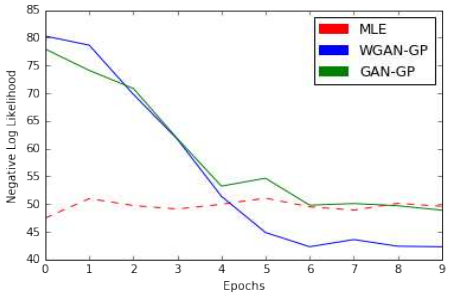


Figure 2: Negative log-likelihood of generated samples under the PCFG using an LSTM trained with the WGAN-GP, GAN-GP and a standard MLE objective on the PTB dataset

4.5 Conditional Generation of Sequences

GANs are able to leverage explicit conditioning on high-level attributes of data (Mirza and Osindero, 2014; Gauthier, 2014; Radford et al., 2015) to generate samples which contain these attributes. Recent work (Hu et al., 2017) generates sentences conditioned on certain attributes of language such as sentiment using a variational autoencoders (VAEs) (Kingma and Welling, 2013) and holistic attribute discriminators. In this paper, we use two features inherent in language - sentiment and questions. To generate sentences that are questions, we use the CMU−SE dataset and label sentences that contain a "?" as being questions and the rest as been statements. To generate sentences of positive and negative sentiment we use the Amazon review polarity dataset collected in (Zhang et al., 2015) and use the first 3 million *short* reviews with a vocabulary of the top 4,000 most frequently occurring words. Conditioning on sentence attributes is achieved by concatenating a single feature map containing either entirely ones or zeros to indicate the presence or absence of the attribute as in (Radford et al., 2015) at the output of each convolutional layer. The conditioning is done on both the generator and the discriminator. We experiment with conditional GANs using only convolutional methods since methods adding conditioning information has been well studied in these architectures.

4.6 Training

All models are trained using the back-propagation algorithm updating our parameters using the Adam optimization method (Kingma and Ba, 2014) and stochastic gradient descent (SGD) with batch sizes of 64. A learning rate of 2×10^{-3}, $\beta_1 = 0.5$ and $\beta_2 = 0.999$ is used in our LSTM generator and discriminators while convolutional architectures use a learning rate of 1×10^{-4}. The noise prior and all LSTM hidden dimensions are set to 128 except for the Chinese poetry generation task where we set it to 64.

5 Results and Discussion

Table. 1 presents quantitative results on generating sentences that adhere to the simple CFG described in Section 4.1. The Acc column computes the accuracy with which our model generates samples from the CFG using a sample of 1,280 generations. We observe that all models are able to fit sequences of length 5 but only the WGAN, WGAN-GP objectives are able to generalize to longer sequences of length 11. This motivated us to use only the WGAN and WGAN-GP objectives in our subsequent experiments. The GAN-GP criterion appears to perform reasonably as well but we restrict our experiments to use the WGAN and WGAN-GP criteria only. GANs have been shown to exhibit the phenomenon of "mode dropping" where the generator fails to capture a large fraction of the modes present in the data generating distribution (Che et al., 2016). It is therefore important to study the diversity in our generated samples. The Uniq column computes the number of unique samples in a sample 1,280 generations serves as a rough indicator of sample diversity. The WGAN-GP objective appears to encourage the generation

Gen	Disc	Objective	Length 5		Length 11	
			Acc (%)	Uniq	Acc (%)	Uniq
LSTM	LSTM	GAN	99.06	0.913	0	0.855
LSTM	LSTM	LSGAN	99.45	0.520	0	0.855
LSTM	LSTM	WGAN	93.98	0.972	98.04	0.924
LSTM-P	LSTM	WGAN	97.96	0.861	99.29	0.653
LSTM	LSTM	WGAN-GP	99.21	0.996	96.25	0.992
CNN	CNN	WGAN-GP	98.59	0.990	97.01	0.771
LSTM-P	LSTM	GAN-GP	98.68	0.993	96.32	0.995

Table 1: Accuracy and uniqueness measure of samples generated by different models. LSTM, LSTM-P refers to the LSTM model with the output peephole and the WGAN-GP and GAN-GP refer to models that use a gradient penalty in the discriminator's training objective

Models	Poem 5				Poem 7			
	BLEU-2		BLEU-3		BLEU-2		BLEU-3	
	Val	Test	Val	Test	Val	Test	Val	Test
MLE (Che et al., 2017)	-	0.693	-	-	-	0.318	-	-
Sequence GAN (Yu et al., 2016)	-	0.738	-	-	-	-	-	-
MaliGAN-basic (Che et al., 2017)	-	0.740	-	-	-	0.489	-	-
MaliGAN-full (Che et al., 2017)	-	0.762	-	-	-	0.552	-	-
LSTM (ours)	0.840	0.837	0.427	**0.372**	0.660	0.655	0.386	**0.405**
LSTM Peephole (ours)	0.845	**0.878**	0.439	0.363	0.670	**0.670**	0.327	0.355

Table 2: BLEU scores on the poem-5 and poem-7 datasets

of diverse samples while also fitting the data distribution well.

Fig. 2 shows the negative-log-likelihood of generated samples using a LSTM architecture using the WGAN-GP, GAN-GP and MLE criteria. All models used an LSTM generator. The sequence length is set to 7 and the likelihoods are evaluated at the end of every epoch on a set of 64 samples.

Table. 2 contains quantitative results on the Chinese poetry generation dataset. The results indicate that our straightforward strategy to overcome back-propagating through discrete states is competitive and outperforms more complicated methods.

Table. 5 contains sequences generated by our model conditioned on sentiment (positive/negative) and questions/statements. The model is able to pick up on certain consistent patterns in questions as well as when expressing sentiment and use them while generating sentences.

Tables 3 and 4 contain sequences generated at the word and character-level by our LSTM and CNN models. Both models are able to produce realistic sentences. The CNN model with a WGAN-GP objective appears to be able to maintain context over longer time spans.

6 Conclusion and Future work

In conclusion, this work presents a straightforward but effective method to train GANs for natural language. The simplicity lies in *forcing the discriminator to operate on continuous values* by presenting it with a sequence of probability distributions from the generator and a sequence of 1-hot vectors corresponding to data from the true distribution. We propose an evaluation strategy that involves learning the data distribution defined by a CFG or PCFG. This lets us evaluate the likelihood of a sample belonging to the data generating distribution. The use of WGAN and WGAN-GP objectives produce realistic sentences on datasets of varying complexity (CMU-SE, Penn Treebank and the 1-billion dataset). We also show that it is possible to perform conditional generation of text on high-level sentence features such as sentiment and questions. In future work, we would like to explore GANs in other domains of NLP such as non goal-oriented dialog systems where a clear train-

Level	Method	1-billion-word
Word	LSTM	An opposition was growing in China . This is undergoing operation a year . It has his everyone on a blame . Everyone shares that Miller seems converted President as Democrat . Which is actually the best of his children . Who has The eventual policy and weak ?
	CNN	Companies I upheld , respectively patented saga and Ambac. Independence Unit have any will MRI in these Lights It is a wrap for the annually of Morocco The town has Registration matched with unk and the citizens
Character	CNN	To holl is now my Hubby , The gry timers was faller After they work is jith a But in a linter a revent

Table 3: Word and character-level generations on the 1-billion word dataset

Level	Model	PTB	CMU-SE
Word	LSTM	what everything they take everything away from . may tea bill is the best chocolate from emergency . can you show show if any fish left inside . room service , have my dinner please .	<s>will you have two moment ? </s> <s>i need to understand deposit length . </s> <s>how is the another headache ? </s> <s>how there , is the restaurant popular this cheese ? </s>
	CNN	meanwhile henderson said that it has to bounce for. I'm at the missouri burning the indexing manufacturing and through .	<s>i 'd like to fax a newspaper . </s> <s>cruise pay the next in my replacement . </s> <s>what 's in the friday food ? ? </s>

Table 4: Word level generations on the Penn Treebank and CMU-SE datasets

POSITIVE	NEGATIVE
best and top notch newtonmom . good buy homeostasis money well spent kickass cosamin of time and fun . great britani ! I lovethis.	usuall the review omnium nothing non-functionable extreme crap-not working and eeeeeew a horrible poor imposing se400
QUESTION	STATEMENT
<s>when 's the friday convention on ? </s> <s>how many snatched crew you have ? </s> <s>how can you open this hall ? </s>	<s>i report my run on one mineral . </s> <s>we have to record this now . </s> <s>i think i deeply take your passenger .</s>

Table 5: Coditional generation of text. Top row shows generated samples conditionally trained on amazon review polarity dataset with two attributes 'positive' and 'negative'. Bottom row has samples conditioned on the 'question' attribute

ing and evaluation criterion does not exist.

Acknowledgements

The Authors would like to thank Ishaan Gulrajani, Martin Arjovsky, Guillaume Lample, Rosemary Ke, Juneki Hong and Varsha Embar for their advice and insightful comments. We are grateful to the Fonds de Recherche du Québec – Nature et Technologie for their financial support. We would also like to acknowledge NVIDIA for donating a DGX-1 computer used in this work.

Appendix

We demonstrate that our approach to solve the problem of discrete outputs produces reasonable outputs even when applied to images. Figure 3 shows samples generated on the binarized MNIST dataset (Salakhutdinov and Murray, 2008). We used a generator and discriminator architecture identical to (Radford et al., 2015) with the WGAN-GP criterion. The generator's outputs are continuous while samples from the true data distribution are binarized.



Figure 3: Binarized MNIST samples using a DCWGAN with gradient penalty

References

Martin Arjovsky and Léon Bottou. 2017. Towards principled methods for training generative adversarial networks. In *NIPS 2016 Workshop on Adversarial Training. In review for ICLR*. volume 2016.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein gan. *arXiv preprint arXiv:1701.07875* .

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*. pages 1171–1179.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432* .

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*. ACM, pages 41–48.

Eric Brill. 1993. Automatic grammar induction and parsing free text: A transformation-based approach. In *Proceedings of the workshop on Human Language Technology*. Association for Computational Linguistics, pages 237–242.

Tong Che, Yanran Li, Athul Paul Jacob, Yoshua Bengio, and Wenjie Li. 2016. Mode regularized generative adversarial networks. *arXiv preprint arXiv:1612.02136* .

Tong Che, Yanran Li, Ruixiang Zhang, R Devon Hjelm, Wenjie Li, Yangqiu Song, and Yoshua Bengio. 2017. Maximum-likelihood augmented discrete generative adversarial networks. *arXiv preprint arXiv:1702.07983* .

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005* .

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

Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. 2016. Language modeling with gated convolutional networks. *arXiv preprint arXiv:1612.08083* .

Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Alex Lamb, Martin Arjovsky, Olivier Mastropietro, and Aaron Courville. 2016. Adversarially learned inference. *arXiv preprint arXiv:1606.00704* .

Jay Earley. 1970. An efficient context-free parsing algorithm. *Communications of the ACM* 13(2):94–102.

Jon Gauthier. 2014. Conditional generative adversarial nets for convolutional face generation. *Class Project for Stanford CS231N: Convolutional Neural Networks for Visual Recognition, Winter semester* 2014(5):2.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Aistats*. volume 15, page 275.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. pages 2672–2680.

Alex Graves and Jürgen Schmidhuber. 2009. Offline handwriting recognition with multidimensional recurrent neural networks. In *Advances in neural information processing systems*. pages 545–552.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. 2017. Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028* .

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

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

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

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167* .

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

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2015. Character-aware neural language models. *arXiv preprint arXiv:1508.06615* .

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

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

Dan Klein and Christopher D Manning. 2003. A parsing: fast exact viterbi parse selection. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*. Association for Computational Linguistics, pages 40–47.

Alex M Lamb, Anirudh Goyal ALIAS PARTH GOYAL, Ying Zhang, Saizheng Zhang, Aaron C Courville, and Yoshua Bengio. 2016. Professor forcing: A new algorithm for training recurrent networks. In *Advances In Neural Information Processing Systems*. pages 4601–4609.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*. Barcelona, Spain, volume 8.

Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, and Zhen Wang. 2016. Least squares generative adversarial networks. *arXiv preprint ArXiv:1611.04076* .

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics* 19(2):313–330.

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

Mehdi Mirza and Simon Osindero. 2014. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784* .

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*. pages 807–814.

Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. 2016. f-gan: Training generative neural samplers using variational divergence minimization. In *Advances in Neural Information Processing Systems*. pages 271–279.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, pages 311–318.

Alec Radford, Luke Metz, and Soumith Chintala. 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434* .

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732* .

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685* .

Ruslan Salakhutdinov and Iain Murray. 2008. On the quantitative analysis of deep belief networks. In *Proceedings of the 25th international conference on Machine learning*. ACM, pages 872–879.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. 2016. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*. pages 2226–2234.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8(3-4):229–256.

Ronald J Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural computation* 1(2):270–280.

Sam Wiseman and Alexander M. Rush. 2016. Sequence-to-sequence learning as beam-search optimization. *CoRR* abs/1606.02960. http://arxiv.org/abs/1606.02960.

Yuhuai Wu, Yuri Burda, Ruslan Salakhutdinov, and Roger Grosse. 2016. On the quantitative analysis of decoder-based generative models. *arXiv preprint arXiv:1611.04273* .

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2016. Seqgan: sequence generative adversarial nets with policy gradient. *arXiv preprint arXiv:1609.05473* .

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329* .

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

Xingxing Zhang and Mirella Lapata. 2014. Chinese poetry generation with recurrent neural networks. In *EMNLP*. pages 670–680.