# Adversarial Discrete Sequence Generation without Explicit Neural Networks as Discriminators

**Zhongliang Li** Wright State University

**Tian Xia, Xingyu Lou, Kaihe Xu, Shaojun Wang, Jing Xiao** Ping An Technology

## Abstract

This paper presents a novel approach to train GANs for discrete sequence generation without resorting to an explicit neural network as the discriminator. We show that when an alternative mini-max optimization procedure is performed for the value function where a closed form solution for the discriminator exists in the maximization step, it is equivalent to directly optimizing the Jensen-Shannon divergence (JSD) between the generator's distribution and the empirical distribution over the training data without sampling from the generator, thus optimizing the JSD becomes computationally tractable to train the generator that generates sequences of discrete data. Extensive experiments on synthetic data and real-world tasks demonstrate significant improvements over existing methods to train GANs that generate discrete sequences.

## 1 Introduction

Discrete sequence generation is an important problem in unsupervised learning, and recently neural autoregressive models such as LSTMs have shown excellent performance (Graves, 2013), (Mikolov et al., 2010). A common approach to train an LSTM model is to maximize the log-likelihood of each ground-truth token given previous observed tokens (Graves, 2013), (Mikolov et al., 2010). However, maximum likelihood estimation (MLE) typically suffers from exposure bias (Bengio et al., 2015), (Ranzato et al., 2016), i.e., the discrepancy between training and inference stages. During inference, each token is generated based on previously generated tokens, while during training ground-truth tokens are used at each time step. A scheduled sampling approach (Bengio et al., 2015) is presented to address this problem, but is proven to be fundamentally inconsistent (Huszár, 2015).

Adversarial training has emerged as a powerful paradigm to train deep generative models. The generative adversarial nets (GANs) (Goodfellow et al., 2014) that use a discriminator to guide the training of the generator have enjoyed considerable success in generating real-valued data such as images. However, they have a very limited success to generate sequences of discrete data due to the difficulty of passing the gradient update from the discriminator to the generator because of the discrete output from the generator.

Attempts have been made to overcome such difficulties, which can be categorized into two paths. One path is to provide a continuous approximation of the discrete distribution (i.e., multinomial) on discrete sequence to make the model end-to-end differentiable. Distinctive works are TextGAN (Zhang et al., 2017), FMD-tGAN (Chen et al., 2018) and GumbelSoftmax GAN (Kusner and Hernández-Lobato, 2016). Most popular path is to treat discrete sequence generation as a sequential decision making process (Bachman and Precup, 2015), which can be potentially solved by reinforcement learning (RL) techniques (Sutton and Barto, 2018). The generative model is treated as an agent with a stochastic parameterized policy, the state is the generated tokens so far and the action is the next token to be generated. To overcome the difficulty that the gradient cannot pass back to the generative model when the output is discrete, the gradient of the generator is estimated via the policy-gradient algorithm (Sutton and Barto, 2018). Representative works include SeqGAN (Yu et al., 2017), LeakGAN (Guo et al., 2018), RankGAN (Lin et al., 2017) and MaskGAN (Fedus et al., 2018). Although promising results have been achieved, one major disadvantage with such RL-based strategies is that they typically yield high-variance gradient estimates, which is hard to overcome.

In this work, we present a novel RL-free approach to train GANs for discrete sequence generation without

Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS) 2019, Naha, Okinawa, Japan. PMLR: Volume 89. Copyright 2019 by the author(s).

resorting to an explicit neural network as the discriminator. We show that when an alternative mini-max optimization procedure is performed for the value function where a closed form solution for the discriminator exists in the maximization step, it is equivalent to directly optimizing the JSD between the generator's distribution and the empirical distribution over the training data, where the corresponding Kullback–Leibler (KL) divergence related to the generator's distribution and the empirical distribution over the training data is reduced to the sum over the training data. Thus surprisingly sampling from the generator over the entire space is not needed anymore, and there is no need to use the policy-gradient algorithm to compute the gradient of the generator. Optimizing the JSD becomes computationally tractable to train the generator that generates sequences of discrete data. We conduct extensive experiments on synthetic data and real-world tasks to demonstrate significant improvements in terms of generation quality based on either negative log-likelihood on the true distribution for synthetic data or BLEU statistics for real data over existing RL-based methods to train GANs that generate discrete sequences.

## 2 Related Work

In the original GAN paper (Goodfellow et al., 2014), theoretical studies have been developed for the optimality of the two-player minimax game and its global convergence in the functional space of probability density functions. The optimal discriminator is derived with obscurities between the GAN training objective with respect to the true but *unknown* data distribution, and its empirical counterpart used in practical training. Following the framework of empirical process in machine learning theory (Mohri et al., 2012), in the work of (Sinn and Rawat, 2018), the authors present a more rigorous mathematical analysis about the training of GAN to make a clear distinction between training with respect to the true but *unknown* data distribution, and its empirical counterpart. However the authors couldn't explore the particular property of the empirical distribution over training data to simplify the training algorithm as we're doing in this work, instead they pursue the idea of smoothing the JSD by incorporating noise in the input distributions of the discriminator, and the generator densities are replaced by kernel density estimates.

Huszar (2015) studies a fundamental problem in unsupervised learning and generative models: the choices of objective functions. He argues that the key organizing principle should be that the objective function used for training a probabilistic model matches the way the model will ultimately be used. It's well known that MLE is equivalent to minimizing the KL divergence between the empirical data distribution and the model distribution. However MLE tends to favor approximations of model distribution that overgeneralize the data distribution. Instead the reverse KL divergence is a good metric of evaluating generative models for perceptual quality, and it tends to favor under-generalization. The JSD is a kind of combination of these two KL divergences, and minimizing JSD would exhibit a behavior that is a kind of halfway between the two extremes. However the author couldn't propose any training algorithms to directly minimizing JSD beyond the original GAN paper (Goodfellow et al., 2014).

Various methods have been proposed for adversarial discrete sequence generation using the policy-gradient algorithm (Chen et al., 2018; Fedus et al., 2018; Guo et al., 2018; Lin et al., 2017; Subramanian et al., 2017; Zhang et al., 2017) in RL. In SeqGAN (Yu et al., 2017), the data generation process is modeled as a RL problem, the state is previously generated sub-sequence, the action is the next token to be generated, and the generator is a stochastic policy that maps current state to a distribution over the action space. Monte Carlo (MC) search is employed to approximate the state-action value. After the whole sequence is generated, it's fed to the discriminator to distinguish real and generated sample sequence, to get reward for updating the policy. LeakGAN (Guo et al., 2018) is an extension of SeqGAN that provides richer award information from the discriminator to the generator using techniques in hierarchical reinforcement learning (Vezhnevets et al., 2017). RankGAN (Lin et al., 2017) replaces the original binary classifier discriminator with a ranking model by taking a softmax over the expected cosine distances from the generated sequences to the real data. Finally Mask-GAN (Fedus et al., 2018) employs an actor-critic training procedure to provide rewards at each time step, thus reducing the high-variance of the gradient updates in a high action-space environment when operating at word-level in natural language. Nevertheless, in our approach, by exploring the particular property of the empirical distribution on training data, we proof that optimal discriminator has a closed form solution, and there is no need to use neural network to approximate the discriminator, and computing JSD can be purely based on training data, which implies that training the generator is purely based on training data, and policy gradient algorithm becomes unnecessary.

## 3 Adversarial Training without Explicit Neural Networks as Discriminators

In the well-known adversarial modeling framework (Goodfellow et al., 2014), there are two players, one is called the generator represented by a differentiable function $G$, the other is the discriminator represented by a differentiable function $D$. The generator $G$ creates samples that are intended to come from the same distribution as the training data, the discriminator $D(x)$ examines samples to determine whether they are real or fake via representing the probability that $x$ came from the data rather than $G$. The discriminator $D$ is trained to maximize the probability of assigning the correct label to both training examples and samples from G. The generator $G$ is simultaneously trained to minimize $\log(1 - D(x))$ where $x$ is generated by $G$. Thus, the discriminator $D$ and the generator $G$ play the following two-player minimax game with *distinguishability game value function* $V(G, D)$

$$\begin{aligned}\min_G \max_D V(G, D) = \mathtt{E}_{x\sim\tilde{p}_{\text{data}}(x)} \log[D(x)] \\ + \mathtt{E}_{x\sim p_G(x)}[\log(1 - D(x))]\end{aligned} \tag{1}$$

where $\tilde{p}_{\text{data}}(x)$ denotes the empirical distribution of $X$ over training data $\mathcal{C} = \{x_1, \cdots, x_N\}$ with $N$ samples, and

$$\tilde{p}_{\text{data}}(x) = \begin{cases} \frac{1}{N} & \text{if } x \in \mathcal{C} \\ 0 & \text{otherwise} \end{cases}$$

In all existing works of GANs (Goodfellow, 2016), two neural networks are used to represent the discriminator $D$ and the generator $G$ respectively, and an alternative mini-max optimization procedure is performed for the training of these two neural networks, where a mini-batch stochastic gradient ascend algorithm maximizing the value function $V(G, D)$ is used to train $D$ and the mini-batch samples are *selected* from training data set $\mathcal{C}$, and then a mini-batch stochastic gradient descend algorithm minimizing the value function $V(G, D)$ to train $G$ and the mini-batch samples are *generated* by $G$, this procedure is repeated until convergence.

In this paper, instead of using two neural networks for the discriminator $D$ and the generator $G$ respectively, we propose an approach that uses a neural network for the generator $G$ but not for the discriminator $D$, an alternative mini-max optimization procedure is performed for the value function (1) where a closed form solution for the discriminator $D$ exists in the maximization step, and it can be shown by Proposition 1 that it is equivalent to directly optimizing the JSD between the model's distribution and the empirical distribution $\tilde{p}_{\text{data}}(x)$ of $X$ over the training data $\mathcal{C}$ that is tractable, especially when $x$ is a sequence. Thus, directly optimizing the JSD between the model's distribution and the empirical distribution $\tilde{p}_{\text{data}}(x)$ of $X$ over the training data $\mathcal{C}$ implies an alternative minimax optimization procedure is implicitly performed with respect to the generator $G$ and the discriminator $D$.

**Proposition 1.** When optimizing the value function $V(G, D)$, for any given generator $G$, the optimal discriminator $D^*_G(x)$ is

$$D^*_G(x) = \begin{cases} \frac{\tilde{p}_{\text{data}}(x)}{\tilde{p}_{\text{data}}(x)+p_g(x)} & \text{if } x \in \mathcal{C} \\ 0 & \text{otherwise} \end{cases}$$

and the value function at the optimal discriminator $D^*_G(x)$ becomes

$$\begin{aligned} V(G, D^*_G(x)) =& 2JSD(\tilde{p}_{\text{data}}(x) \parallel p_G(x)) - \log 4 \\ =& \sum_{x\in\mathcal{C}} \tilde{p}_{\text{data}} \log[\frac{\tilde{p}_{\text{data}}(x)}{\tilde{p}_{\text{data}}(x) + p_G(x)}] \\ &+ \sum_{x\in\mathcal{C}} p_G(x)[\log \frac{p_G(x)}{\tilde{p}_{\text{data}}(x) + p_G(x)}] \end{aligned} \tag{2}$$

where $JSD(p \parallel q)$ is the Jensen-Shannon divergence between two distributions $p$ and $q$, and is symmetric.

Proof: Given any generator $G(x)$, the training criterion for the discriminator $D(x)$ is to maximize the value function $V(G, D)$ in (1) and could be re-written as.

$$\begin{aligned} V(G, D) =& \sum_{x\in\mathcal{C}} \tilde{p}_{\text{data}} \log[D(x)] + \int_{x\notin\mathcal{C}} 0 \log[D(x)] \\ &+ \sum_{x\in\mathcal{C}} p_G(x)[\log(1 - D(x))] \\ &+ \int_{x\notin\mathcal{C}} p_G(x)[\log(1 - D(x))]dx \end{aligned} \tag{3}$$

Taking derivative with respect to $D(x)$, we have

$$\frac{\partial V(G, D)}{\partial D(x)} = \begin{cases} \frac{\tilde{p}_{\text{data}}(x)}{D(x)} - \frac{p_G(x)}{(1-D(x))} & \text{if } x \in \mathcal{C} \\ -\frac{p_G(x)}{(1-D(x))} & \text{otherwise} \end{cases}$$

Thus, the optimal solution for $D(x)$ is

$$D^*_G(x) = \begin{cases} \frac{\hat{p}_{\text{data}}(x)}{\hat{p}_{\text{data}}(x)+p_G(x)} & \text{if } x \in \mathcal{C} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

Plug (4) into (3), we have

$$\begin{aligned} V(G, D^*_G(x)) =& \sum_{x\in\mathcal{C}} \tilde{p}_{\text{data}} \log[\frac{\tilde{p}_{\text{data}}(x)}{\tilde{p}_{\text{data}}(x) + p_G(x)}] \\ &+ \sum_{x\in\mathcal{C}} p_G(x)[\log \frac{p_G(x)}{\tilde{p}_{\text{data}}(x) + p_G(x)}] \\ &+ \int_{x\notin\mathcal{C}} p_G(x)[\log(1 - 0)]dx \\ =& \sum_{x\in\mathcal{C}} \tilde{p}_{\text{data}} \log[\frac{\tilde{p}_{\text{data}}(x)}{\tilde{p}_{\text{data}}(x) + p_G(x)}] \\ &+ \sum_{x\in\mathcal{C}} p_G(x)[\log \frac{p_G(x)}{\tilde{p}_{\text{data}}(x) + p_G(x)}] \end{aligned}$$

On the other hand,

$$\begin{aligned}
&2JSD(\tilde{p}_{\text{data}}(x) \parallel p_G(x)) - \log 4 \\
&= KL(\tilde{p}_{\text{data}}(x) \parallel \tilde{p}_{\text{data}}(x) + p_G(x)) \\
&\quad + KL(p_G(x) \parallel \tilde{p}_{\text{data}}(x) + p_G(x)) \\
&= \mathtt{E}_{x\sim\tilde{p}_{\text{data}}(x)} \log[\frac{\tilde{p}_{\text{data}}(x)}{\tilde{p}_{\text{data}}(x) + p_G(x)}] \\
&\quad + \mathtt{E}_{x\sim p_G(x)}[\log \frac{p_G(x)}{\tilde{p}_{\text{data}}(x) + p_G(x)}] \\
&= \sum_{x\in\mathcal{C}} \tilde{p}_{\text{data}}(x) \log[\frac{\tilde{p}_{\text{data}}(x)}{\tilde{p}_{\text{data}}(x) + p_G(x)}] \\
&\quad + \int_{x\notin\mathcal{C}} 0 \log[\frac{0}{0 + p_G(x)}] \\
&\quad + \sum_{x\in\mathcal{C}} p_G(x)[\log \frac{p_G(x)}{\tilde{p}_{\text{data}}(x) + p_G(x)} \\
&\quad + \int_{x\notin\mathcal{C}} p_G(x)[\log \frac{p_G(x)}{0 + p_G(x)}] \\
&= \sum_{x\in\mathcal{C}} \tilde{p}_{\text{data}}(x) \log[\frac{\tilde{p}_{\text{data}}(x)}{\tilde{p}_{\text{data}}(x) + p_G(x)}] \\
&\quad + \sum_{x\in\mathcal{C}} p_G(x)[\log \frac{p_G(x)}{\tilde{p}_{\text{data}}(x) + p_G(x)}]
\end{aligned} \tag{5}$$

Thus, equation (2) is true. □

In the original GAN paper (Goodfellow et al., 2014), similar proof was given by assuming $G$ has enough capacity to cover data distribution without distinction between the true but *unknown* data distribution and the empirical data distribution $\tilde{p}_{\text{data}}(x)$, let alone further exploring the empirical data distribution's property that it's 0 everywhere except on training data set $\mathcal{C}$. Previous works (Chen et al., 2018), (Fedus et al., 2018), (Guo et al., 2018), (Huszár, 2015), (Lin et al., 2017), (Yu et al., 2017) don't realize this property and claim that it is intractable to directly optimize $JSD(\tilde{p}_{\text{data}}(x) \parallel p_G(x))$ when $X$ is a high-dimensional data or a structured data such as a sequence, so they have to use REINFORCE (Williams, 1992) algorithm and Monte Carlo search approximation (Yu et al., 2017) or an actor-critic (Fedus et al., 2018) et al. But in fact, equation (2) shows that to compute $JSD(\tilde{p}_{\text{data}}(x) \parallel p_G(x))$, we need only to consider the samples over training data set $\mathcal{C}$, which it's indeed computationally tractable, and no sampling is needed in contrast to those in many GANs. This is especially advantageous over those that handle discrete sequences (Chen et al., 2018), (Fedus et al., 2018), (Guo et al., 2018), (Lin et al., 2017), (Yu et al., 2017).

Since the optimal solution for the discriminator $D$ has a closed form solution given by equation (4), thus, resorting to a neural network that has a limited capacity and is trained with limited data to represent the discriminator $D$ might not fully recover $D^*_G(x)$, it leads to a sub-optimal solution and thus is in fact unnecessary.

The form of $D^*_G(x)$ given by equation (4) implies that any sample generated by $G$ that is not the same as in the training data set $C$ should be treated as fake and discarded since the corresponding value on $D$ is 0 and the value function $V(G, D^*_G(x))$ is 0. Thus unlike in existing methods that a neural network is used as the discriminator, and the generator generates arbitrary samples and the discriminator gives a confidence value as coming from real or fake, in our approach, the optimal discriminator $D^*_G(x)$ just ignores any samples not the same as in the training data $\mathcal{C}$, and only considers the samples the same as in the training data $\mathcal{C}$ and gives a confidence value as coming from real. The generator assigns a probability to each sample in $\mathcal{C}$ and evaluate the effect of each sample on the value function. This means that the generator only needs the same samples as in the training data $\mathcal{C}$ to maximize the value function as shown by equation (2).

In the work of (Arjovsky and Bottou, 2017), the authors show that when the data distribution and the generator distribution have supports that are disjoint or lie on low dimensional manifolds, the optimal discriminator will be perfect and its gradient will be zero almost everywhere. For the traditional GAN, when a neural network is used to approximate the optimal discriminator better, either the gradient of the generator vanishes or unstable behavior appears in practice. They propose an improved alternative deemed Wasserstein-GAN (Arjovsky et al., 2017) to overcome these problems. Sinn and Rawat (2018) similarly but in a much simpler way prove that vanishing gradient and unstable behavior are an inevitable consequence unless the training samples coincide with the generated samples from the generator. In our approach, we force the generator to work on the support of empirical data distribution, and the manifold of $p_G(x)$ has to line up perfectly with the empirical data distribution $\tilde{p}_{\text{data}}(x)$. Vanishing gradient or unstable behavior during training should be avoided.

As discussed before, Proposition 1 shows that directly optimizing the JSD between the model's distribution and the empirical data distribution $\tilde{p}_{\text{data}}(x)$ of $X$ over the training data $\mathcal{C}$ implies that an alternative mini-max optimization procedure is implicitly performed with respect to the generator $G$ having parameters $\theta$ and the discriminator $D$ with an optimal form given by $D^*_G(x)$. The training algorithm in our approach is indeed *an adversarial training*. The traditional training algorithm for GANs requires finding the Nash equilibrium of a game and the discriminator must be synchronized well with the generator during training, this is a more difficult problem than optimizing an objective function as we are doing in this paper.

Now assume that we use a neural network with parameters $\theta$ as the generator $G$, then the probability of sample $x$ on the generator is denoted as $p^{\theta}_{G}(x)$, and the optimal discriminator is denoted as $D^{*}_{G^{\theta}}(x)$. Now by the result in (2), let's look at how to compute the gradient of $JSD(\tilde{p}_{\text{data}}(x) \parallel p^{\theta}_{G}(x))$ for a given sample $x \in \mathcal{C}$ as below.

$$\begin{aligned}
&\frac{\partial JSD(\tilde{p}_{\text{data}}(x) \parallel p^{\theta}_{G}(x))}{\partial \theta} \\
&\propto -\tilde{p}_{\text{data}}(x)\frac{\partial \log(\tilde{p}_{\text{data}}(x) + p^{\theta}_{G}(x))}{\partial \theta} + \frac{\partial (p^{\theta}_{G}(x)\log p^{\theta}_{G}(x))}{\partial \theta} \\
&\quad - \frac{\partial (p^{\theta}_{G}(x)\log(\tilde{p}_{\text{data}}(x) + p^{\theta}_{G}(x)))}{\partial \theta} \\
&= -\frac{\tilde{p}_{\text{data}}(x)p^{\theta}_{G}(x)}{\tilde{p}_{\text{data}}(x) + p^{\theta}_{G}(x)}\frac{\partial \log p^{\theta}_{G}(x)}{\partial \theta} \\
&\quad + p^{\theta}_{G}(x)(\log p^{\theta}_{G}(x) + 1)\frac{\partial \log p^{\theta}_{G}(x)}{\partial \theta} \\
&\quad - p^{\theta}_{G}(x)\log(\tilde{p}_{\text{data}}(x) + p^{\theta}_{G}(x))\frac{\partial \log p^{\theta}_{G}(x)}{\partial \theta} \\
&\quad - p^{\theta}_{G}(x)\frac{1}{\tilde{p}_{\text{data}}(x) + p^{\theta}_{G}(x)}p^{\theta}_{G}(x)\frac{\partial \log p^{\theta}_{G}(x)}{\partial \theta} \\
&= p^{\theta}_{G}(x)[\log p^{\theta}_{G}(x) - \log(\tilde{p}_{\text{data}}(x) + p^{\theta}_{G}(x))]\frac{\partial \log p^{\theta}_{G}(x)}{\partial \theta} \\
&= p^{\theta}_{G}(x)[\log(1 - D^{*}_{G^{\theta}}(x))]\frac{\partial \log p^{\theta}_{G}(x)}{\partial \theta}
\end{aligned}$$

This result shows that the gradient of $JSD(\tilde{p}_{\text{data}}(x) \parallel p^{\theta}_{G}(x))$ for a given sample $x \in \mathcal{C}$ is a modification of the gradient of the log-likelihood, and we use stochastic gradient descent (SGD) to optimize the JSD between the generator's output distribution and the empirical data distribution purely based on training data. When we use a mini-batch version of SGD in our implementation, we stack a few sequences together. Since the gradient has a term of $p^{\theta}_{G}(x)$, which makes the gradient really small, thus, we have to adapt the learning rate for each time step, moreover there is a large variation for the value of this term among these sequences, it becomes extremely hard to train the model. To overcome these problems, we normalize the term of $p^{\theta}_{G}(x)$ over the batch of sequences to make this term to be either close to 1 or 0, and the training becomes quite stable and easy to tune. The variation for the value of $\log(1 - D^{*}_{G^{\theta}}(x))$ is small, so we don't do normalization for this term.

## 4 Experimental Results

The experiments consist of two parts: synthetic data experiments and real-world scenarios' experiments with four standard benchmarks: Chinese Poems, MS COCO captions, Obama Speech and EMNLP2017 WMT news. We compare the proposed approach with four baseline models, i.e., MLE, SeqGAN (Yu et al., 2017), RankGAN (Lin et al., 2017) and LeakGAN (Guo et al., 2018).

For RL-based approaches such as SeqGAN and RankGAN, a key idea for success is the initialization of the policy in the REINFORCE algorithm to make sure that the model can effectively deal with the large action space of discrete sequence generation. Instead of starting from a poor random policy and training the model to converge towards the optimal policy, the generator is first trained with the cross-entropy loss for a number of epochs using the ground truth sequences. This ensures that we start off with a much better policy than random because now the model can focus on a good part of the search space. For LeakGAN, as proposed in (Guo et al., 2018), an interleaving of supervised training (i.e. MLE) and adversarial training (i.e. GAN) instead of full GAN after the pre-training is performed to train the model. For our approach, we either train the generator directly without pre-training or with MLE as pre-training.

### 4.1 Experiments on Synthetic Data

We conduct a simulated test with synthetic data similar to (Yu et al., 2017). We use a randomly initialized LSTM as the true model, aka, the oracle $G_{\text{oracle}}$, to generate the real data distribution $p_{\text{oracle}}(x_t|x_1, \cdots, x_{t-1})$. In our experiments, the parameters of the LSTM are initialized by the normal distribution $\mathcal{N}(0, 1)$. Then 10,000 sequences of length 20 and 40 are generated respectively as the training set $\mathcal{C}$ for the generative models.

Once the generator $G_\theta$ is learned through a learning algorithm, at the test stage, we use $G_\theta$ to generate 100,000 sequence samples, and calculate negative log-likelihood for each sample by $G_{\text{oracle}}$ and their average score, denoted as $\text{NLL}_{\text{oracle}}$ corresponding to the reverse KL divergence for perceptual quality (Huszár, 2015).

| Length | MLE | SeqGan | RankGan | LeakGan | Ours |
|---|---|---|---|---|---|
| 20 | 9.04 | 8.74 | 8.25 | 7.04 | 5.50 |
| 40 | 10.41 | 10.31 | 9.96 | 7.19 | 4.27 |

Table 1: The overall $\text{NLL}_{\text{oracle}}$ performance on synthetic data.

The overall $\text{NLL}_{\text{oracle}}$ performance is presented in Table 1 for sequence length 20 and 40 respectively. We only report our results without pre-training. Since the evaluation metric is fundamentally instructive, we can see that our approach outperforms MLE and the RL-based baselines, i.e., SeqGAN, RankGAN and LeakGAN, significantly.

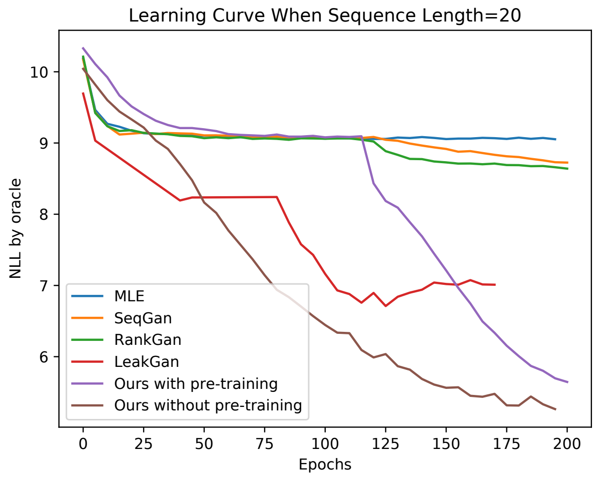

Figure 1: Learning curves of the $\text{NLL}_{\text{oracle}}$ w.r.t. the training epochs when the generated sequence length is 20.

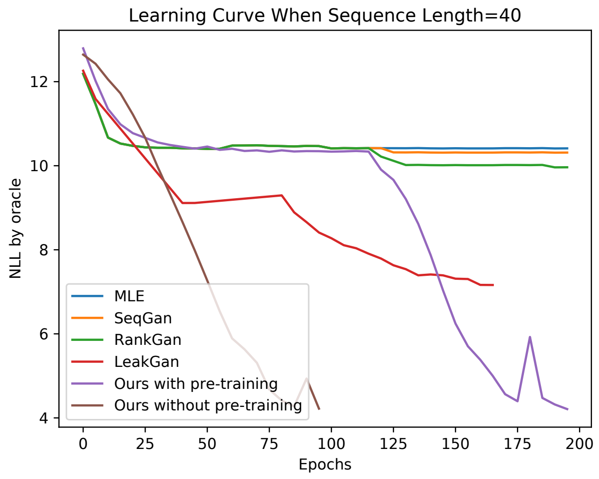

Figure 2: Learning curves of the $\text{NLL}_{\text{oracle}}$ w.r.t. the training epochs when the generated sequence length is 40.

The learning curves of the $\text{NLL}_{\text{oracle}}$ w.r.t. the training epochs are depicted in Figures 1 and 2 for sequence length 20 and 40 respectively. We include our results both without pre-training and with pre-training. For our method with pre-training, it has already shown observable performance superiority compared to other models, which indicates that the proposed RL-free approach itself brings improvement over the previous ones. Our method without pre-training shows a faster speed of convergence, and the local minimum it explores is significantly better than previous results. The results demonstrate the effectiveness of our proposed RL-free approach for generating both short and long discrete sequences.

The gradient of $JSD(\tilde{p}_{\text{data}}(x) \parallel p_G^{\theta}(x))$ for a given sample $x \in \mathcal{C}$ has a term that is the empirical distribution for a sample $x$, in theory, it is $\frac{1}{N}$. But we find that the results are not sensitive to the value of this term. Figures 3 and 4 illustrate the learning curves of the $\text{NLL}_{\text{oracle}}$ w.r.t. the training epochs when the generated sequence length is 20 and 40 respectively, and we vary $N$ in this term from 2, 16, 128, 256 and 512 respectively. When $N$ is 128, we get the best result, but the differences are not that much significant from other values. We think, it is mainly due to the normalization for $p_G^{\theta}(x)$, the value of $N$ doesn't have a large effect on the learning curve.

## 4.2 Experiments on Real-world Scenarios

For experiments on real-world scenarios, we choose four data sets: Chinese poem composition data set (Zhang and Lapata, 2014) as short length text generation, COCO image captions data set (Chen et al., 2015) as mid-length text generation, and Obama political speech data set (https://github.com/samim23/obama-rnn) as well as the EMNLP2017 WMT News data set (http://statmt.org/wmt17/translation-task.html) as long text generation.

Regarding to the Chinese poem composition data set, the Obama political speech data set and EMNLP2017 WMT News data set, we take 60 percent for our training, 20 percent for validating, and 20 percent for testing. While for the COCO image captions data set, we follow previous work's partition (Chen et al., 2015), which consists of only training data and test data, and the models are obtained from the last iteration.

The BLEU score (Papineni et al., 2002) is originally created to automatically measure the quality of machine translation results (Papineni et al. 2002). Here we use the BLEU score to judge the similarity between the generated texts and the human-created texts, where the main idea is to compare the similarity between the results created by machine and the references provided by human. Specifically, for poem evaluation, we set n-gram to be 2 (BLEU-2) since most words (dependency) in classical Chinese poems consist of one or two characters (Yi et al., 2017) and for the similar reason, we use BLEU-2, BLEU-3, BLEU-4 and BLEU-5 or BLEU-2, BLEU-3 and BLEU-4 to evaluate COCO image caption, Obama political speech or EMNLP2017 WMT News generation performance respectively. To compute the BLEU either on validation or test data set, we first generate 5000 sentences by the model learned from the training set. Then we calculate the co-occurrence $n$-grams between the generated sentences and the testing

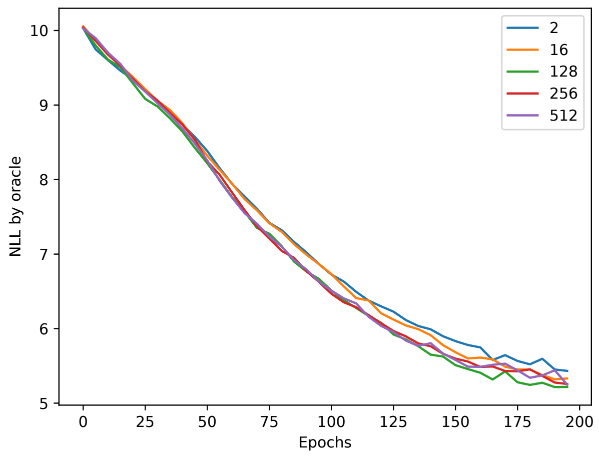

Figure 3: The learning curves of the $\mathrm{NLL}_{\mathrm{oracle}}$ w.r.t. the training epochs with different values of $N$ when the generated sequence length is 20.

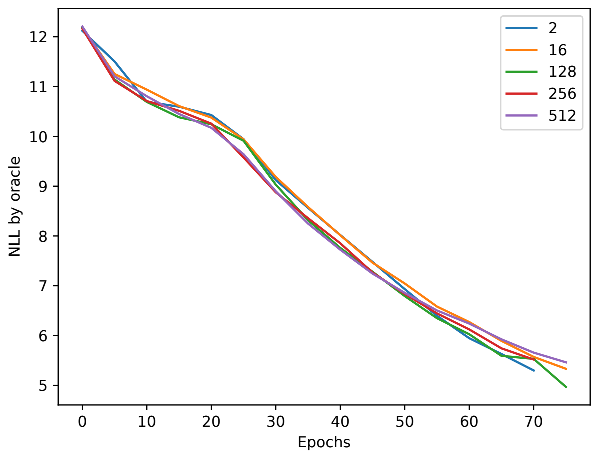

Figure 4: The learning curves of the $\mathrm{NLL}_{\mathrm{oracle}}$ w.r.t. the training epochs with different values of $N$ when the generated sequence length is 40.

data, which are required in evaluating the BLUE score.

Especially, all BLEU scores reported here are computed from our own implementation. We notice that all open source codes of previous works such as SeqGAN (Yu et al., 2017), RankGAN (Lin et al., 2017) and LeakGAN (Chen et al., 2018) are using the BLEU function from NLTK (https://www.nltk.org), a popular natural language processing library. But that implementation is not an efficient way to compute BLEU scores for our tasks, since it compares each generated sequence with each sequence in the test data set, and compute the $n$-grams for both sequences in order to obtain BLEU score. When the set of generated sequences and the one of sequences in the test data set are long, this implementation runs very slowly in practice. There is an efficient way to compute the BLEU scores by following the standard BLEU score definition, we can optimize it based on our task: The sequences in validation or test data set are fixed, so we pre-compute their $n$-grams in advance, then compare with each generated sequence to get the BLEU scores. This significantly fastens the speed to obtain the BLEU scores.

The Chinese poem composition data set is a corpus of 16,394 Chinese quatrains, each containing four lines of twenty characters in total, the vocabulary size is 1200. To focus on a fully automatic solution and stay general, we do not use any prior knowledge of special structure rules in Chinese poems such as specific phonological rules. Since each line quite short, we use the BLEU-2 scores as the evaluating metrics. The experimental results are provided in Table 2. The results on Chinese poem composition data set indicate that our approach successfully handles the short text generation tasks.

| | BLEU-2 |
|---|---|
| MLE | 0.421 |
| SeqGAN | 0.511 |
| RankGAN | 0.523 |
| LeakGAN | 0.408 |
| Ours | 0.536 |

Table 2: The testing performance measured by BLEU-2 on the Chinese poem composition data set.

The COCO image captions data set contains groups of image-description pairs. We take the image captions as the text to generate. Note that the COCO image captions data set is not a long text data set, in which most sentences are of about 10 words. Thus, we apply some pre-processing on the data set. The COCO Image Captions training data set consists of 20,734 words and 417,126 sentences. We remove the words with frequency lower than 10 as well as the sentence containing them. After the pre-processing, the data set includes 4,980 words. The authors of this data set randomly sample 80,000 sentences for the training set, and another 5,000 for the test set. So we just use this fixed partition, which can be found in their Github repository (https://github.com/CR-Gjx/LeakGAN). The BLEU scores results on the COCO image captions data set are provided in Table 3. The results indicate that in most cases our approach performs better than all of the baseline models in mid-length text generation task.

In the Obama political speech generation task, we use a corpus, which is a collection of 11,092 paragraphs from Obama's political speeches and the vocabulary size is 4357. The results of BLEU scores are provided in

| | BLEU-2 | BLEU-3 | BLEU-4 | BLEU-5 |
|---|---|---|---|---|
| MLE | 0.786 | 0.565 | 0.323 | 0.127 |
| SeqGAN | 0.871 | 0.712 | 0.519 | 0.322 |
| RankGAN | 0.901 | 0.734 | 0.514 | 0.412 |
| LeakGAN | 0.897 | 0.772 | 0.607 | 0.396 |
| Ours | 0.894 | 0.780 | 0.641 | 0.459 |

Table 3: The testing performance measured by BLEUs on the COCO image captions data set.

| | BLEU-2 | BLEU-3 | BLEU-4 |
|---|---|---|---|
| MLE | 0.814 | 0.567 | 0.281 |
| SeqGAN | 0.849 | 0.615 | 0.331 |
| RankGAN | 0.848 | 0.607 | 0.307 |
| LeakGAN | 0.924 | 0.756 | 0.541 |
| Ours | 0.912 | 0.752 | 0.560 |

Table 4: The testing performance measured by BLEUs on the Obama political speech data set.

Table 4. The results of the BLEU scores on the Obama political speech data set indicate that, in most cases, our approach performs better than all of the baseline models except LeakGan, and LeakGan achieves the best results on BLEU-2 and BLEU-3 in this long-length text generation task.

In the EMNLP2017 WMT news data set, we run preprocessing similar to the Obama data set. Because this data set is quite large, we sample about 70,000 sentences as the whole corpus which has 86577 words. Then we use the same vocabulary size as Obama's and replace the most infrequent words with '<unk>' token. The experimental results are shown in Table 5. Our approach performs better than all of the baseline models in this long-length text generation task.

| | BLEU-2 | BLEU-3 | BLEU-4 |
|---|---|---|---|
| MLE | 0.871 | 0.650 | 0.397 |
| SeqGAN | 0.906 | 0.725 | 0.487 |
| RankGAN | 0.932 | 0.783 | 0.561 |
| LeakGAN | 0.910 | 0.756 | 0.596 |
| Ours | 0.943 | 0.816 | 0.610 |

Table 5: The testing performance measured by BLEUs on the EMNLP2017 WMT news data set.

## 5 Conclusion and Future Work

By exploring the particular property of the empirical distribution on training data, in this paper, we present a novel approach for adversarial discrete sequence generation without explicit neural networks as discriminators. Since the optimal discriminator has a closed form solution, using neural network to approximate the discriminator becomes sub-optimal. Moreover computing JSD can be purely based on training data, this leads to a direct RL-free optimization algorithm to train the generator purely based on training data. A common question in GAN training is when to stop the training, directly computing JSD on training data or validation data provides such a metric.

When directly optimizing the JSD between the model's distribution and the empirical data distribution $\tilde{p}_{\text{data}}(x)$ of $X$ over the training data $\mathcal{C}$, an alternative mini-max optimization procedure is implicitly performed with respect to a neural network based generator $G$ with parameters $\theta$ and the discriminator $D$ with an optimal form given by $D^*_G(x)$, thus the proposed training algorithm is indeed an adversarial training.

In the synthetic data experiments, an oracle evaluation mechanism is used to explicitly illustrate the superiority of our proposed approach over RL-based baselines. For four real-world scenarios, i.e., Chinese poem composition, COCO image captions, Obama political speech and EMNLP2017 WMT news generation, in most cases, our approach show excellent performance on generating the creative sequences.

Contrary to neural language model trained by MLE, when applying the generation model trained by our approach to machine translation and/or speech recognition re-ranking, it's not viable to use the reverse KL divergence for the re-ranking of candidates since we don't know the true distribution of a sentence of nature language, instead we could use the JSD of a sentence as a feature for the re-ranking.

For sequence to sequence applications such as speech recognition, machine translation, abstractive summarization and image captioning etc., we have to adapt our approach for conditional text generation, aka conditional GAN. We don't pursue this and compare the performance with other approaches (Bengio et al., 2015), (Norouzi et al., 2016), (Nowozin et al., 2016) for conditional generative models in this work.

One limitation for our approach is that it is only applicable to the case of $p_G(x)$ having an explicit representation, but for many successful applications, especially in image generations, implicit representation is used instead.

For future work, we plan to investigate the theoretical properties such as consistency and generalization of the proposed approach. Also we look forward to extending the proposed approach to a broader family of divergence measures as f-GAN (Nock et al., 2017), (Nowozin et al., 2016) does to the original GAN.

## References

Arjovsky, M. and Bottou, L. (2017). Towards principled methods for training generative adversarial networks. In *International Conference on Learning Representations (ICLR)*.

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein generative adversarial networks. In *Proceedings of the 34nd International Conference on Machine Learning, ICML*.

Bachman, P. and Precup, D. (2015). Data generation as sequential decision making. In *Advances in Neural Information Processing Systems 28*, pages 3249–3257.

Bengio, S., Vinyals, O., Jaitly, N., and Shazeer, N. (2015). Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1171–1179.

Chen, L., Dai, S., Tao, C., Shen, D., Gan, Z., Zhang, H., Zhang, Y., and Carin, L. (2018). Adversarial text generation via feature-mover's distance. In *Advances in Neural Information Processing Systems*.

Chen, X., Fang, H., Lin, T., Vedantam, R., Gupta, S., Dollár, P., and Zitnick, C. L. (2015). Microsoft COCO captions: Data collection and evaluation server. *CoRR*, abs/1504.00325.

Fedus, W., Goodfellow, I. J., and Dai, A. M. (2018). MaskGAN: Better text generation via filling in the ______. In *The 6th International Conference on Learning Representations*.

Goodfellow, I. (2016). NIPS 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680.

Graves, A. (2013). Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Guo, J., Lu, S., Cai, H., Zhang, W., Yu, Y., and Wang, J. (2018). Long text generation via adversarial training with leaked information. In *The Thirty-Second AAAI Conference on Artificial Intelligence*, pages 5141–5148.

Huszár, F. (2015). How (not) to train your generative model: Scheduled sampling, likelihood, adversary? *arXiv preprint arXiv:1511.05101*.

Kusner, M. J. and Hernández-Lobato, J. M. (2016). GANs for sequences of discrete elements with the gumbel-softmax distribution. *CoRR*, abs/1611.04051.

Lin, K., Li, D., He, X., Zhang, Z., and Sun, M.-T. (2017). Adversarial ranking for language generation. In *Advances in Neural Information Processing Systems*, pages 3158–3168.

Mikolov, T., Karafiát, M., Burget, L., Cernocký, J., and Khudanpur, S. (2010). Recurrent neural network based language model. In *INTERSPEECH*, pages 1045–1048.

Mohri, M., Rostamizadeh, A., and Talwalkar, A. (2012). *Foundations of Machine Learning*. MIT Press.

Nock, R., Cranko, Z., Menon, A. K., Qu, L., and Williamson, R. C. (2017). f-GANs in an information geometric nutshell. In *Advances in Neural Information Processing Systems 30*, pages 456–464.

Norouzi, M., Bengio, S., Chen, Z., Jaitly, N., Schuster, M., Wu, Y., and Schuurmans, D. (2016). Reward augmented maximum likelihood for neural structured prediction. In *Advances in Neural Information Processing Systems 29*, pages 1723–1731.

Nowozin, S., Cseke, B., and Tomioka, R. (2016). f-GAN: Training generative neural samplers using variational divergence minimization. In *Advances in Neural Information Processing Systems*, pages 271–279.

Papineni, K., Roukos, S., Ward, T., and jing Zhu, W. (2002). BLEU: a method for automatic evaluation of machine translation. In *The 40th Annual meeting of the Association for Computational Linguistics*, pages 311–318.

Ranzato, M., Chopra, S., Auli, M., and Zaremba, W. (2016). Sequence level training with recurrent neural networks. In *The 4th International Conference on Learning Representations*.

Sinn, M. and Rawat, A. (2018). Non-parametric estimation of jensen-shannon divergence in generative adversarial network training. In *Proceedings of the 21st International Conference on Artificial Intelligence and Statistics*, pages 642–651.

Subramanian, S., Rajeswar, S., Dutil, F., Pal, C., and Courville, A. C. (2017). Adversarial generation of natural language. In *Proceedings of the 2nd*

*Workshop on Representation Learning for NLP, Rep4NLP@ACL 2017, Vancouver, Canada, August 3, 2017*, pages 241–251.

Sutton, R. and Barto, A. (2018). *Reinforcement Learning: An Introduction.* MIT Press, 2 edition.

Vezhnevets, A. S., Osindero, S., Schaul, T., Heess, N., Jaderberg, M., Silver, D., and Kavukcuoglu, K. (2017). Neural networks for hierarchical reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning*, pages 3540–3549.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Machine Learning*, page 229–256.

Yi, X., Li, R., and Sun, M. (2017). Generating Chinese classical poems with RNN encoder-decoder. In *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data - 16th China National Conference, CCL, - and - 5th International Symposium, NLP-NABD*, pages 211–223.

Yu, L., Zhang, W., Wang, J., and Yu, Y. (2017). SeqGAN: Sequence generative adversarial nets with policy gradient. In *Proceedings of the Thirty-first AAAI Conference on Artificial Intelligence*, pages 2852–2858.

Zhang, X. and Lapata, M. (2014). Chinese poetry generation with recurrent neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 670–680, Doha, Qatar. Association for Computational Linguistics.

Zhang, Y., Gan, Z., Fan, K., Chen, Z., Henao, R., Shen, D., and Carin, L. (2017). Adversarial feature matching for text generation. In *Proceedings of the 34th International Conference on Machine Learning*, pages 4006–4015.