# Improving Maximum Likelihood Training for Text Generation with Density Ratio Estimation

**Yuxuan Song**
Shanghai Jiao Tong University

**Ning Miao**
Bytedance AI lab

**Hao Zhou**
Bytedance AI lab

**Lantao Yu**
Stanford University

**Mingxuan Wang**
Bytedance AI lab

**Lei Li**
Bytedance AI lab

## Abstract

Autoregressive sequence generative models trained by Maximum Likelihood Estimation suffer the exposure bias problem in practical *finite* sample scenarios. The crux is that the number of training samples for Maximum Likelihood Estimation is usually limited and the input data distributions are different at training and inference stages. Many methods have been proposed to solve the above problem (Yu et al., 2017; Lu et al., 2018), which relies on sampling from the non-stationary model distribution and suffers from high variance or biased estimations. In this paper, we propose $\psi$-MLE, a new training scheme for autoregressive sequence generative models, which is effective and stable when operating at large sample space encountered in text generation. We derive our algorithm from a new perspective of self-augmentation and introduce bias correction with density ratio estimation. Extensive experimental results on synthetic data and real-world text generation tasks demonstrate that our method stably outperforms Maximum Likelihood Estimation and other state-of-the-art sequence generative models in terms of both quality and diversity.

## 1 Introduction

Deep generative models dedicate to learning a target distribution and have shown great promise in numerous scenarios, such as image generation (Arjovsky et al., 2017; Goodfellow et al., 2014), density estimation (Ho et al., 2019; Salimans et al., 2017; Kingma and Welling, 2013; Townsend et al., 2019), stylization (Ulyanov et al., 2016), and text generation (Yu et al., 2017; Li et al., 2016). Learning generative models for text data is an important task which has significant impact on several real world applications, *e.g.*, machine translation, literary creation and article summarization. However, text generation remains a challenging task due to the discrete nature of the data and the huge sample space which increases exponentially with the sentence length.

Text generation is nontrivial for its huge sample space. For generating sentences of various lengths, current text generation models are mainly based on density factorization instead of directly modeling the joint distribution, which results in the prosperity of neural autoregressive models on language modeling. As neural autoregressive models have explicit likelihood function, it is straightforward to employ Maximum Likelihood Estimation (MLE) for training. Although MLE is is asymptotically consistent, for practical **finite** sample scenarios, it is prone to overfit on the training set. Additionally, during the inference (generation) stage, the error at each time step will accumulate along the sentence generation process, which is also known as the *exposure bias* (Ranzato et al., 2015) problem.

Many efforts have been devoted to address the above limitations of MLE. Researchers have proposed several non-MLE methods based on minimizing different discrepancy measures, *e.g.*, Sequential GANs (Yu et al., 2017; Che et al., 2017; Kusner and Hernández-Lobato, 2016) and CoT (Lu et al., 2018). However, non-MLE methods typically relies on sampling from the gener-

Proceedings of the 23[rd]International Conference on Artificial Intelligence and Statistics (AISTATS) 2020, Palermo, Italy. PMLR: Volume 108. 

The work was done during the first author's internship at Bytedance AI lab.

ative distribution to estimate gradients, which results in high variance and instability during training, as the generative distribution is non-stationary during training process. Some recent study (Caccia et al., 2018) empirically shows that non-MLE methods potentially suffer from mode collapse problem and cannot actually outperform MLE in terms of quality and diversity tradeoff.

In this paper, we seek to leverage the ability of generative models itself for providing unlimited amount of samples to augment the training dataset, which has the potential of alleviating the overfitting problem due to limited samples, as well as addressing the exposure bias problem by providing the model with prefixes (input partial sequences) sampled from its own distribution. To correct the bias incurred by sampling from the model distribution, we propose to learn a progressive density ratio estimator based on Bregman divergence minimization. The above procedures together form a novel training scheme for sequence generative models, termed $\psi$-MLE.

Another essential difference between MLE and $\psi$-MLE lies in the fact that the likelihood of samples not in training set are equally penalized through normalization in MLE, whether near or far from the true distribution. While $\psi$-MLE takes the difference in the quality of unseen samples into account through the importance weight assigned by density ratio estimator, which can be expected to get further improvement.

Empirically, MLE with mixture training data gives the same performance as vanilla MLE training with only training data. But our proposed $\psi$-MLE consistently outperforms vanilla MLE training. Additionally, we empirically demonstrate the superiority of our algorithm over many strong baselines like GAN in terms of generative performance (in the quality-diversity space) with both synthetic and real-world datasets.

## 2 Preliminary

### 2.1 Notations

We denote the target data distribution as $p_{\text{data}}$ , and the empirical data distribution as $\hat{p}_{\text{data}}$ . The parameters of the generative model $G$ are presented by $\theta$ and the parameters of a density ratio estimator $r$ are presented by $\psi$. $p_\theta$ denotes the distribution implied by the tractable density generative model $G$. The objective is to fit the underlying data distribution $p_{\text{data}}$ with a parameterized model distribution $p_\theta$ with empirical samples from $p_{\text{data}}$. We use $s$ to stand for a sample sequence from datasets or from generator's output. And $s_l$ stands for the l-th token of $s$, where $s_0 = \emptyset$.

### 2.2 MLE vs Sequential GANs

It should be noticed that both MLE and GANs for sequence generation suffer from their corresponding issues. In this section, we delve deeply into the specific properties of MLE and GANs, and explore how these properties affect their performances in modeling sequential data.

**MLE** The objective of Maximum Likelihood Estimation (MLE) is:

$$L_{\text{MLE}}(\theta) = \mathbb{E}_{s\sim p_{\text{data}}} \left[\log p_\theta(s)\right] \tag{1}$$

where $p_\theta(s)$ is the learned probability of sequence $s$ in the generative model. Maximizing the objective is equivalent to minimizing the Kullback-Leibler (KL) divergence:

$$D_{\text{KL}}(p_{\text{data}} \,||p_\theta) = \mathbb{E}_{s\sim p_{\text{data}}} \log \frac{p_{\text{data}}\,(s)}{p_\theta(s)} \tag{2}$$

Though MLE has lots of attractive properties, it has two critical issues:

1) MLE is prone to overfitting on small training sets. Training an autoregressive sequence generative model with MLE on a training set consists of sentences of length $L$, the standard objective can be derived as following:

$$L_{\hat{\text{MLE}}}(\theta) = \mathop{\mathbb{E}}_{s\sim \hat{p}_{\text{data}}} \sum_{l=1}^{L} \log p_\theta\left(s_l|s_{1:l-1}\right) \tag{3}$$

The forced exposure to the ground-truth data shown in Eq. 3 is known as "teacher forcing", which causes the problem of overfitting. What makes thing worse is the exposure bias. During training, the model only learns to predict $s_l$ given $s_{1:l-1}$, which are fluent prefixes in the training set. During sampling, when there are some small mistakes and the first $l-1$ can no longer make up a very fluent sentence, the model may easily fail to predict $s_l$.

2) KL-divergence punishes the situation where the generation model gives real data points low probabilities much more severely than that where unreasonable data points are given high probabilities. As a result, models trained with MLE will focus more on not missing real data points than avoiding generating data points of low quality.

**Sequential GANs** Sequential GANs (Yu et al., 2017; Guo et al., 2018), are proposed to overcome the above shortcomings of MLE. The typical objective of

them is:

$$L_{\mathrm{GAN}}(\theta) = \min_{\theta} -\mathbb{E}_{s\sim p_\theta}\left[\sum_{t=1}^{n} Q_t\left(s_{1:t-1}, s_t\right)\cdot \log p_\theta\left(s_t|s_{1:t-1}\right)\right] \tag{4}$$

$Q_t\left(s_{1:t-1}, s_t\right)$ is action value, which is usually approximated by a discriminators evaluation on the complete sequences sampled from the prefix $s_{t+1} = [s_{1:t-1}, s_t]$. The main advantage of GANs is that when we update the generative model, error will be explicitly reduced by the effect of normalizing constant.

However, there is also a major drawback of GANs. As the gradient is estimated by REINFORCE algorithm (Yu et al., 2017), the generated distribution is non-stationary. As a result, the estimated gradient may suffer from high variance. Though many methods have been proposed to stabilize the training of sequential GANs, *e.g* control variate (Che et al., 2017) or MLE pretraining (Yu et al., 2017), there, they only have limited effect on sequential data. Moreover, as indicated by recent works (Caccia et al., 2018), sequential GANs sharpen density functions in the distribution's support, which sacrifices diversity for better quality.

## 3 Methodology

In order to combine the advantages of MLE, which directly trains the model on high-quality training samples, and GANs, which actively explore unseen spaces, we propose $\psi$-MLE. We further remove noise points of $\psi$-MLE by performing importance sampling whose weight is given by a density ratio estimator.

### 3.1 $\psi$-MLE for Sequence Generation

The different properties of MLE and GANs mainly result from $O$, the effect zone of supervision. To be concrete, $O$ is a subset of all possible data points, whose likelihoods are directly updated during training. MLE only maximizes the probabilities of points in the training set, which is discrete and finite. However, the actual data space contains far more points than the training set, on which there is no supervision. In contrast, as the generators of GANs are able to generate all possible data points, $O_{GAN}$ is essentially the whole data space. Large enough though $O_{GAN}$ is, the supervision signal, i.e., the gradients for updating GANs' generators usually have high variances compared with the gradients of MLE.

To combine the merits of both methods, we propose $\psi$-MLE which blends samples generated by the current generation model into training data:

$$p_{\mathrm{mix}}(S) = m p_{\mathrm{data}}(S) + (1-m) p_\theta(S), \tag{5}$$

where $m \in [0, 1]$ is the proportion of training data. By $\psi$-MLE, we extend $O$ to the whole space. And since there are real training data in the mixture samples, the gradients are more informative with lower variances.

For training, we directly minimize the forward KL divergence between $p_{\mathrm{mix}}$ and $p_\theta$, which is equivalent to performing MLE on samples from $p_{\mathrm{mix}}$. Since the training goal at each step is to maximize:

$$\mathrm{E}_{p_{\mathrm{mix}}(S)}[\log p_\theta(S)], \tag{6}$$

when the KL-divergence decrease, the gap between $p_\theta$ and $p_{data}$ get smaller. Eventually, when $p_\theta \approx p_{\mathrm{mix}}$, $p_\theta$ also approximates $p_{data}$.

However, $p_{mix}$ may be very different from $p_{data}$, especially at the beginning of training. This discrepancy may result in generating really poor samples which have high likelihoods in $p_\theta$ but not in $p_{data}$. As a result, the training set gets noisier, which may harm performance.

### 3.2 Noise Reduction by Importance Sampling

To make the distribution of training samples closer to $P_{data}$, we introduce the following importance sampling method. The main idea is to first get a batch of samples from $P_{mix}$, and then give each sample an importance weight $r$ according to its similarity with real samples. Then the training objective turns into:

$$\mathrm{E}_{p_{\mathrm{mix}}(S)}[r_\psi(S)\log p_\theta(S)], \tag{7}$$

where $\psi$ is the parameter of the importance weight estimator.

In ideal conditions, where $r_\psi(S) = r_{\mathrm{optimal}}(S) = \frac{p_{\mathrm{data}}(S)}{p_{\mathrm{mix}}(S)}$, the training essentially minimizes the KL-divergence between $P_\theta$ and real data distribution $P_{data}$:

$$\begin{aligned} &\mathrm{E}_{p_{\mathrm{mix}}(S)}[r_\psi(S)\log p_\theta(S)] \\ =&\mathrm{E}_{p_{\mathrm{mix}}(S)}[\frac{p_{\mathrm{data}}(S)}{p_{\mathrm{mix}}(S)}\log p_\theta(S)] \\ =&\mathrm{E}_{p_{\mathrm{data}}(S)}[\log p_\theta(S)] \\ \approx&\frac{1}{T}\sum_{i=1}^{T}\frac{p_{\mathrm{data}}\left(s^{(i)}\right)}{p_{\mathrm{mix}}\left(s^{(i)}\right)}\log p_\theta\left(s^{(i)}\right), \end{aligned} \tag{8}$$

where s in the last equation are samples from $P_{mix}$. We assert that dividing $p_{\mathrm{data}}$ by $p_{\mathrm{mix}}$ won't cause any numerical problem, since the support of $p_{\mathrm{data}}$ is a subset of $p_{\mathrm{mix}}$'s support.

However, it is infeasible to directly calculate $r_{\mathrm{optimal}} = \frac{p_{\mathrm{data}}}{p_{\mathrm{mix}}}$. So we need to approximate it by $r_\psi$. The first

thought is to use a new parametric model $p_\beta$ to approximate $p_{\text{mix}}$, and set $r_\psi = 1 - \frac{p_\theta}{p_\beta}$. But this method will lead to severe numerical instability. In this paper, we choose to directly approximate $r_{\text{optimal}}$ by training a discriminator between $p_{\text{data}}$ and $p_{\text{mix}}$. To be more concrete, we first assign positive labels $y = 1$ to samples from $p_{\text{data}}(s)$ and negative labels $y = 0$ to samples from $p_{\text{mix}}(s)$. Then we train a probabilistic classifier $c$: $\mathcal{S} \to [0, 1]$ to output the probability of $s$ belonging to each class. After the training of $c$ converges, we set $r_\psi = \gamma \frac{c(\mathbf{s})}{1-c(\mathbf{s})}$ and get the following proposition:

**Proposition 1** *With a Bayes optimal classifier c,*

$$r_\psi = r_{optimal}. \tag{9}$$

$\gamma = \frac{p(y=0)}{p(y=1)}$ is the amount ratio of negative samples and positive samples. We keep $\gamma = 1$ by using the same number of negative and positive samples in a mini-batch.

Note that the density ratio is obtained indirectly from the classifier $c$ which is typically poorly calibrated. Therefore we need to frequently calibrate $c$ to get a better density ratio estimation and avoid numerical problems caused by miscalibration (Turner et al., 2018). However, it can be quite computationally expensive to calibrate $c$ after each update. To sidestep the above obstacle, directly estimating $\frac{p_{\text{data}}(s)}{p_{\text{mix}}(s)}$ is a more general approach, which may lead to a more accurate density ratio estimation than the "classifier-based" method mentioned above.

Given two distributions $p(x)$ and $q(x)$, the target of direct density estimation is to obtain a density ratio model $r_\psi(x)$, which can directly approximate the true density ratio $r(x) = \frac{p(x)}{q(x)}$. (Sugiyama et al., 2012; Uehara et al., 2016) proposed to utilize the Bregman divergence as a measure of the discrepancy between two density ratio functions, which guides the training of density ratio model. The Bregman divergence is an extension of Euclidean distance which measures the distance between two data points $x_1$ and $x_2$, and the definition with respect to function $f$ is as following:

$$BR'_f(x_1 \| x_2) = f(x_1) - f(x_2) - \nabla f(x_2)(x_1 - x_2) \tag{10}$$

where $f : \Omega \to \mathcal{R}$ is a strictly convex and continuously differentiable function defined on a closed set $\Omega$.

The integration of Bregman divergence $BR'_f[r(x)\|r_\theta(x)]$ between an estimated density ratio function $r_\psi(x)$ and the real density ratio function $r(x)$ under measure $q(x)dx$ is as following:

$$\begin{aligned} BR_f(r\|r_\psi) &= \int BR'_f[r(x)\|r_\psi(x)]q(x)dx \\ &= \int (f(r(x)) - f(r_\psi(x)) - \\ &\quad \nabla f(r_\psi(x))(r(x) - r_\psi(x)))q(x)dx. \end{aligned} \tag{11}$$

Then the estimation procedure can be turned into an optimization procedure with respect to the parameter $\psi$. We leave the discussion with different selections of $f$ in Sec. 4.2. In practical training, we alternatively update $\psi$ and $\theta$. The whole training procedure is in Algorithm 1.

**Algorithm 1** Progressive Bias Correction

**Require:** Generator $p_\theta$ with parameter as $\theta$; Density Ratio estimator $r_\psi$ with parameter as $\psi$; Empirical data distribution as $p_{\hat{\text{data}}}$; A mixture weight $m$.
**repeat**
  Sample two minibatches of samples $\{x_1, \ldots, x_B\}$, $\{x_{B+1}, \ldots, x_{2B}\}$ from $p_\theta$.
  Sample a minibatch of samples $\{y_1, \ldots, y_B\}$ from $p_{\hat{\text{data}}}$
  Create a mixed minibatch $\{z_1, \ldots, z_B\}$ by mixing samples from $\{x_{B+1}, \ldots, x_{2B}\}$ and $\{y_1, \ldots, y_B\}$ according to the mixture weight $m$.
  **for** number of $\psi$ update **do**
    Update $\psi$ according to Eq. 11:
    $\psi^{t+1} = \psi^t - \nabla_\psi \frac{1}{B} \sum_{i=1}^{B} (\nabla f(r_\psi(x_i)) r_\psi(x_i) - f(r_\psi(x_i)) - \nabla f(r_\psi(y_i)))$
  **end for**
  **for** number of $\theta$ update **do**
    Update $\theta$ according to Eq. 19:
    $\theta^{t+1} = \theta^t - \frac{1}{B} \sum_{i=1}^{B} (r_\psi(z_i) \nabla_\theta \log p_\theta(z_i))$
  **end for**
**until** Convergence
**Output:**

## 4 Connection with other methods

In this section, we provide further investigation on direct density ratio estimation and theoretical justification for our proposed methods.

### 4.1 Relation with GANs

As introduced in Sec. 2.2, sequential GANs usually adopt policy gradient methods for training. Their objectives can be interpreted in a Reinforcement Learn-

ing(RL) fashion:

$$\mathcal{L}_{\text{RL-GAN}}(\boldsymbol{\theta};\tau,\hat{p}_{\text{data}}) = -\tau\mathbb{H}\left(p_\theta(s)\right) - \sum_{s\in\mathcal{S}} p_\theta(s) r\left(s,\hat{p}_{\text{data}}\right). \quad (12)$$

In this formula, $r\left(s,\hat{p}_{\text{data}}\right)$ is the reward function, which is usually implemented by a discriminator. In order to mitigate mode collapse, the regulation term $\mathbb{H}\left(p_\theta(s)\right)$ is added. We further introduce an *exponentiated payoff distribution* (Norouzi et al., 2016)

$$q\left(s;\tau\right) = \frac{1}{Z\left(s,\tau\right)} \exp\left\{r\left(s,\hat{p}_{\text{data}}\right)/\tau\right\} \quad (13)$$

Then, we can see that training discrete GANs are essentially minimizing the following KL divergence, $D_{\text{KL}}(p_\theta(s)||q\left(s;\tau\right))$, which is shown at following:

$$\begin{aligned} &D_{\text{KL}}(p_\theta(s)||q\left(s;\tau\right)) \\ &= \mathbb{E}_{s\sim p_\theta}(\log p_\theta(s) - \log(\frac{1}{Z(s,\tau)}\exp\{r(s,\hat{p}_{\text{data}})/\tau\})) \\ &= \frac{1}{\tau}\mathcal{L}_{\text{RL-GAN}}(\theta;\tau) + \text{ constant }. \end{aligned} \quad (14)$$

The last holds by the fact that $Z(s,\tau)$ is a constant during the optimization of $\theta$. Our method can be seen as optimizing the opposite direction of the KL divergence, *i.e*, $D_{\text{KL}}(q\left(s;\tau\right)||p_\theta(s))$. As it is intractable to directly sample from $q\left(s;\tau\right)$, we first sample from $p_{\text{mix}}$ and conduct importance sampling with weight $r_\psi$ to obtain unbiased estimation of $D_{\text{KL}}(q\left(s;\tau\right)||p_\theta(s))$.

### 4.2 Relation with $f$-Divergence

Density ratio estimation is closely to related f-divergence (Nowozin et al., 2016), which measures the difference between two probability distributions. Given two distributions with absolutely continuous density functions $p$ and $q$, the f-divergence is defined as:

$$D_f(p\|q) = \int_{\mathcal{X}} q(x) f\left(\frac{p(x)}{q(x)}\right)\mathrm{d}x = E_q\left[f\left(\frac{p(x)}{q(x)}\right)\right] \quad (15)$$

where $f:\mathbb{R}_+\to\mathbb{R}$ is a convex and lower-semicontinous function with $f(1)=0$.

If $f$ is a strictly convex and continuously differentiable function, the following conclusions can be derived.

**Proposition 2** *Minimizing Bregman divergence between two distributions $p$ and $q$ with respect to $f$ is essentially estimating the $f$-divergence between $p$ and $q$ with $\nabla f(r_\psi(x))$ as the dual coordinates.*

When the true density ratio are available, the f-divergence can also be obtained, it is not surprising that estimating density ratio by minimizing Bregman divergence with respect to function $f$ is essentially the dual of estimating the $f$-divergence by maximizing a variational bound. We rewrite the Eq. 11 as following:

$$\begin{aligned} &BD_f(r\|r_\psi) \\ =&\int (f(r(x)) - f(r_\psi(x)) \\ &- \nabla f(r_\psi(x))(r(x)-r_\psi(x)))q(x)dx \\ =&E_q\left[f\left(\frac{p(x)}{q(x)}\right)\right] - E_{x\sim p}\left[\nabla f\left(r_\psi(x)\right)\right] + \\ &E_{x\sim q}\left[\left(\nabla f\left(r_\psi(x)\right) r_\psi(x) - f\left(r_\psi(x)\right)\right)\right] \end{aligned} \quad (16)$$

After some simple operations, we get:

$$\begin{aligned} &E_{x\sim p}\left[\nabla f\left(r_\psi(x)\right)\right] + \\ &E_{x\sim q}\left[\left(\nabla f\left(r_\psi(x)\right) r_\psi(x) - f\left(r_\psi(x)\right)\right)\right] \\ =&E_q\left[f\left(\frac{p(x)}{q(x)}\right)\right] - BD_f(r\|r_\psi) \le E_q\left[f\left(\frac{p(x)}{q(x)}\right)\right] \end{aligned}$$

The inequality holds for the fact that $BD_f(r\|r_\psi)\ge 0$, and the equality holds if and only if $r_\psi(x)=r(x)$.

Meanwhile, the dual representation of $f$-divergence (Nowozin et al., 2016) is illustrated as follows:

$$\begin{aligned} &D_f(p\|q) \ge \sup_{T\in\mathcal{T}} \int_{\mathcal{X}} p(x)T(x)\mathrm{d}x - \int_{\mathcal{X}} q(x)f^*(T(x))\mathrm{d}x \\ =&\sup_{T\in\mathcal{T}}\left(\mathbb{E}_{x\sim p}[T(x)] - \mathbb{E}_{x\sim q}\left[f^*(T(x))\right]\right) \\ =&\sup_{r_\psi}(E_{x\sim p}\left[\nabla f\left(r_\psi(x)\right)\right] \\ &+ E_{x\sim q}\left[\left(\nabla f\left(r_\psi(x)\right) r_\psi(x) - f\left(r_\psi(x)\right)\right)\right]), \end{aligned} \quad (17)$$

where $\mathcal{T}$ is an arbitrary class of functions $T$ and $f^*$ corresponds to the Fenchel conjugate of $f$. Last equation in Eq. 17 is valid for the fact that $f^*\left(f^*\left(r_\theta(x)\right)\right) = f\left(r_\theta(x)\right) r_\theta(x) - f\left(r_\theta(x)\right)$. Above discussions also indicate the knowledge distillation perspective of our method, *i.e.*, we minimize the $f$-divergence between $r_\psi p_{\text{mix}}$ and $p_{\text{data}}$ and then distill the knowledge by minimizing the KL divergence between $r_\psi p_{\text{mix}}$ and $p_\theta$.

## 5 Experiments

To demonstrate the effectiveness of our method, we conduct experiments on a synthetic setting as well as two real-world Benchmark datasets. We compare our method with the several baseline methods, including MLE, SeqGAN (Yu et al., 2017), LeakGAN (Guo et al., 2018), COT (Lu et al., 2018) and MaliGAN (Che et al., 2017). Note an important hyperparameter of our method is the mixture weight $m$, which is set

Table 1: Likelihood-based benchmark and time statistics for synthetic Turing test within the temperature scope (Caccia et al., 2018).

| Model | $\text{NLL}_{oracle}$ | $\text{NLL}_{test}$ | best $\text{NLL}_{oracle}$ + $\text{NLL}_{test}$ |
|---|---|---|---|
| MLE | 5.53 | 7.58 | 16.28 |
| SeqGAN (Yu et al., 2017) | 8.12 | 7.92 | 18.44 |
| COT (Lu et al., 2018) | 6.20 | **7.56** | 16.32 |
| LeakGAN (Guo et al., 2018) | 10.01 | 8.52 | 19.45 |
| $\psi$-MLE | **5.09** | **7.56** | **15.98** |

Table 2: Test BLEU and Self-BLEU on EMNLPNEWS.

| | BLEU(↑) | | | | Self-BLEU(↓) | | | |
|---|---|---|---|---|---|---|---|---|
| | 2 | 3 | 4 | 5 | 2 | 3 | 4 | 5 |
| Training Data | 0.86 | 0.61 | 0.38 | 0.23 | 0.86 | 0.62 | 0.38 | 0.24 |
| SeqGAN (Yu et al., 2017) | 0.72 | 0.42 | 0.18 | 0.09 | 0.91 | **0.70** | **0.46** | 0.27 |
| MaliGAN (Che et al., 2017) | 0.76 | 0.44 | 0.17 | 0.08 | 0.91 | 0.72 | 0.47 | **0.25** |
| LeakGAN (Guo et al., 2018) | 0.84 | 0.65 | 0.44 | 0.27 | 0.94 | 0.82 | 0.67 | 0.51 |
| MLE($\alpha = 1.25^{-1}$) | **0.93** | 0.74 | 0.51 | **0.32** | 0.93 | 0.78 | 0.59 | 0.41 |
| $\psi$-MLE($\alpha = 1.25^{-1}$) | **0.93** | **0.76** | **0.54** | **0.33** | **0.91** | 0.75 | 0.56 | 0.38 |

Table 3: Test BLEU and Self-BLEU on Image COCO.

| | BLEU(↑) | | | | Self-BLEU(↓) | | | |
|---|---|---|---|---|---|---|---|---|
| | 2 | 3 | 4 | 5 | 2 | 3 | 4 | 5 |
| Training Data | 0.68 | 0.47 | 0.30 | 0.19 | 0.86 | 0.62 | 0.38 | 0.42 |
| SeqGAN (Yu et al., 2017) | **0.75** | 0.50 | 0.29 | 0.18 | 0.95 | 0.84 | 0.67 | 0.49 |
| MaliGAN (Che et al., 2017) | 0.67 | 0.43 | 0.26 | 0.16 | 0.92 | 0.78 | 0.61 | 0.44 |
| LeakGAN (Guo et al., 2018) | 0.74 | 0.52 | 0.33 | 0.21 | 0.93 | 0.82 | 0.66 | 0.51 |
| MLE | 0.74 | 0.52 | 0.33 | 0.21 | **0.89** | 0.72 | 0.54 | 0.38 |
| $\psi$-MLE($\alpha = 1.0^{-1}$) | **0.75** | **0.53** | **0.36** | **0.23** | **0.89** | **0.70** | **0.53** | **0.36** |

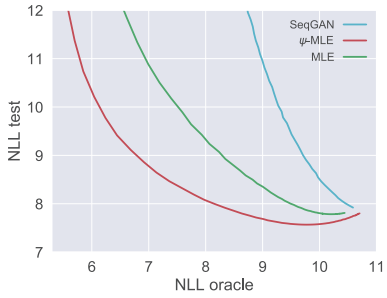


Figure 1: temperature curve for MLE, $\psi$-MLE and SeqGAN

as $\frac{1}{2}$ for default in all experiments except the ablation studies on $m$ in Sec. 5.4.

## 5.1 Implementation Details

### 5.1.1 Bregman Divergence Minimization

The density ratio in Sec. 3.2 is estimated through an optimization procedure towards Bregman divergence

A variety of functions meet the requirements of $f$, but in all the experiments, we choose $f(t) = t \log t - (1 + t) \log(1 + t)$ as the default objective for its numerical stability during training. The effect of using different objectives is also analyzed empirically in Section. 5.4.

### 5.1.2 Variance Reduction

The density ratio estimator, *i.e.*, $r_\psi$, can be seen as the importance weight for correcting the bias in the hybrid distribution $p_{\text{mix}}$. In order to increase sample quality, we apply two variance-reduction methods on importance sampling to our method (Owen, 2013; Grover et al., 2019):

- *self-normalization*: the self-normalized estimator normalizes the density ratio across a samples batch:

$$\mathrm{E}_{p_{\text{mix}}(s)}[r_\psi(s) \log p_\theta(s)] \approx \sum_{i=1}^{T} \frac{r_\psi(s)}{\sum_{i=1}^{T} r_\psi(s)} \log p_\theta(s) \tag{18}$$

where $T$ is the batch size.

Table 4: $f(t)$ and corresponding objectives($\sigma(\cdot)$ stands for sigmoid function)

| $f(t)$ | objectives |
|---|---|
| $(t-1)^2/2$ | $\frac{1}{2}E_{x\sim p_{\text{mix}}}(r_\psi^2(x)-1)-E_{x\sim p_{\text{data}}}(r_\psi(x)-1)$ |
| $t\log t-(1+t)\log(1+t)$ | $E_{x\sim p_{\text{mix}}}(-\log(r_\psi(x)+1))-E_{x\sim p_{\text{data}}}(\log(\frac{r_\psi(x)}{r_\psi(x)+1}))$ |
| $\ln\left(1+e^t\right)$ | $E_{x\sim p_{\text{mix}}}(\sigma(r_\psi(x))r_\psi(x)-\ln\left(1+e^{r_\psi(x)}\right))-E_{x\sim p_{\text{data}}}(\sigma(r_\psi(x)))$ |

- *ratio flattening*: the density ratio can be flattened to achieve an intermediate state between the original $r_\psi(s)$ and the uniform importance weights by a parameter $\alpha \geq 0$:

$$\mathrm{E}_{p_{\text{mix}}(s)}[r_\psi(s)\log p_\theta(s)] \approx \sum_{i=1}^{T} r_\psi(s)^\alpha \log p_\theta(s) \tag{19}$$

We find that self-normalization works best, so all experiments are implemented with self-normalization.

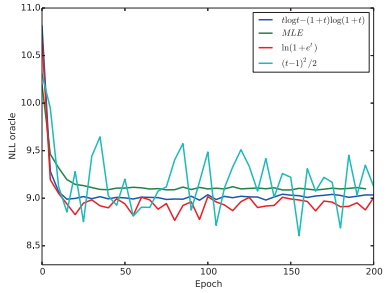


Figure 2: Training dynamics of MLE and $\psi$-MLE with different $f(t)$

### 5.2 Synthetic Experiments

The synthetic experiments are conducted following the typical settings of previous works (Yu et al., 2017; Guo et al., 2018; Lu et al., 2018). We use a randomly initialized LSTM as the oracle model. Then we test each generation model's ability in learning from samples generated by the oracle most. We use a single layer LSTM with 32 hidden units. The parameters are initialized by a standard normal distribution. With a fixed LSTM as the target, the ground-truth density is available. Hence it is possible to analyze the generation quality quantitatively by the negative log-likelihood $\text{NLL}_{oracle}$ given by the Oracle model. Besides, the log-likelihood the generative model assigns to the held-out test data, *i.e.*, $\text{NLL}_{test}$ is another metric used to evaluate sample diversity.

As is pointed out by (Caccia et al., 2018), evaluating quality alone is actually misleading for sequence generation task. Note the conditional probability is formalized as: $p_\theta(s_t|s_{1:t-1}) = \text{softmax}(o_t \cdot W/\alpha)$. Here $o_t$ is the pre-logit activation of generator, $W$ is the word embedding matrix and $\alpha$ is a Boltzmann temperature parameter. (Caccia et al., 2018) introduced a temperature sweep procedure, which enumerates the possible values of $\alpha$ in a predefined range, and report the corresponding $\text{NLL}_{oracle}$ and $\text{NLL}_{test}$. In the same way, we get a curve of $\text{NLL}_{oracle}$ and $\text{NLL}_{test}$ with different temperatures (Fig. 1). We find that the curve of our method is under the curves of all baseline methods, showing the superiority of our method.

Quantitative results are reported in Table 1, including the best $\text{NLL}_{oracle}$, $\text{NLL}_{test}$ and the comprehensive evaluation metric $\text{NLL}_{oracle} + \text{NLL}_{test}$. The quantitative results are obtained by tuning the temperature in the valid range defined in (Caccia et al., 2018), which indicates the quality, diversity and their trade-off of a training paradigm under the constraint that the tuned model is still a valid and information language model. Our method outperforms previous methods as it combines the strengths of MLE and GANs.

### 5.3 Real Data Experiments

We conduct real-data experiments on two text benchmark datasets, image COCO captions and EMNLP 2017 News. We use BLEU score between generated samples and the whole test set to evaluate the generation quality. At the same time, we use self-BLEU (Zhu et al., 2018) as a metric of diversity, which is the average bleu score between each generated sample and all other generated samples. Following (Caccia et al., 2018), the temperature is selected when the BLEU scores is similar to the reported numbers in (Guo et al., 2018) for fair comparison.

COCO mainly contains short image captions, while EMNLP News 2017 consists of longer formal texts. The results on COCO and EMNLP News 2017 are shown in Table 3 and Table 2 respectively. We find that our model achieves higher BLEU scores and lower self-BLEU scores, revealing better quality and higher diversity of our model.

### 5.4 Ablation Study and Sensitive Analysis

#### 5.4.1 Mixture Weight

One important hyperparameter in our model is the mixture weight $m$ for constructing the proposal distri-

Table 5: Hyperparameter Study on Mixture Weight $m$

| $m$ | 0 | 1/4 | 1/2 | 3/4 | 1 |
|---|---|---|---|---|---|
| $NLL_{\text{oracle}}$ | 7.60 | 6.23 | **5.09** | 5.63 | 5.53 |
| $NLL_{\text{test}}$ | 8.01 | 7.91 | 7.56 | **7.54** | 7.60 |
| $NLL_{\text{oracle}}$ $+NLL_{\text{test}}$ | 17.43 | 16.27 | 15.98 | **15.94** | 16.30 |

bution $p_{\text{mix}}$. To figure out how the method behaves with different $m$, we gradually increase $m$ from 0 to 1, and show the experiment results in Table 5. With a small $m$, it can be observed that our $\psi$-MLE has similar performance with sequential GANs. This is due to the fact that $p_{\text{mix}}$ are more similar to $p_\theta$ and specifically, our method actually degenerates to a variant of sequential GAN when $m = 0$. Correspondingly, the model is closer to MLE when $m$ approach 1. The best performance of $\psi$-MLE is achieved when $m$ is set as an intermediate value between 1 and 0, where both the exploration properties of GANs and the stability of MLE are incorporated. These experiments further justify the connection among $\psi$-MLE, MLE and GANs.

#### 5.4.2 Objective Density Functions

As illustrated in Table 4, we show a family of objectives which meet the definition of Bregman Divergence and are available for direct density ratio estimation. We conduct ablation studies within the synthetic experiment setting to find out the training dynamic of different objectives in practice. The results can be found in Fig. 2. Note the training procedure with $f(t) = (t-1)^2/2$ is not stable due to the numerical issues. When $f(t)$ is set as $\ln(1+e^t)$ and $t\log t - (1+t)\log(1+t)$, $\psi$-MLE get remarkable improvements upon the MLE with a more stable training procedure.

## 6 Related Works

In the context of sequence generation models, there have been fruitful lines of studies focusing on leveraging adversarial training for sequence generation task. These works are inspired by the generative adversarial nets (Goodfellow et al., 2014), an implicit generative model which seeks to minimize the Jensen-Shannon divergence between the generative distribution and the real data distribution through a two-play min-max game. In the sequence generation task, the gradients can not be directly back-propagated to the generative module as in continuous setting. Hence reparameterization (Kusner and Hernández-Lobato, 2016) or policy gradient (Yu et al., 2017; Guo et al., 2018; Che et al., 2017) is utilized to obtain unbiased estimate of gradients.

Our method actually can be seen as a more generalized objective family, with MLE and policy gradient based GANs as two special cases. Our method is close related to the methods leveraging tractable density distribution as noise to estimate another density, especially self-contrastive estimation (Goodfellow, 2014). While self-contrastive estimation is the degenerate version of $\psi$-MLE, *i.e.* directly using samples from $p_{\text{mix}}$ as ground truth to conduct MLE without the bias correction step with $r_\psi$. COT (Lu et al., 2018) also leverages tractable density as noise. Our approach differs from COT in the calculation of the density ratio. They introduced another generative module for estimating the denominator to obtain the density ratio, while we apply direct density ratio estimation methods which are more flexible and efficient.

Density ratio estimation has come into attention of the community of generative models. (Nowozin et al., 2016) indicates a general objective family for training GANs of which the density ratio is a key element. (Uehara et al., 2016) further investigates the connection between the GANs and density ratio estimation. Density ratio estimator also has been utilized within the settings when the aim is to improve a learned generative model. (Azadi et al., 2018; Turner et al., 2018) leveraged density ratio to conduct rejection sampling over the support of generative distribution for obtaining high-quality samples. Similarly, (Grover et al., 2019) utilized an importance sampling framework to correct the biased statistics of generated distribution which result in improvements in several application scenarios of generative model.

## 7 Discussion and Future Work

We propose $\psi$-MLE, a new sequence generation training paradigm which is effective and stable when operating at large sample space encountered in sequence generation. Our method is derived based on the concept termed *effect zone of supervision* which accounts for the properties of different sequence generation models. We propose a generalized family of *effect zone of supervision* through self augmentation and a following density-ratio based bias correction procedure to achieve unbiased optimization during each training step. Experimental results demonstrate that $\psi$-MLE is able to achieve better quality and diversity trade-off compared with previous sequence generation methods. An exciting avenue for future work is to extend the our training paradigm into the conditional text generation tasks, such as machine translation, dialog system and abstractive summarization. Also we look forward to providing further investigation on the consistency and generalization properties of our proposed approach.

**Acknowledgements**

We thank the anonymous reviewers for their insightful comments. Hao Zhou and Lei Li are the corresponding authors of this paper.

**References**

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223, 2017.

Samaneh Azadi, Catherine Olsson, Trevor Darrell, Ian Goodfellow, and Augustus Odena. Discriminator rejection sampling. *arXiv preprint arXiv:1810.06758*, 2018.

Massimo Caccia, Lucas Caccia, William Fedus, Hugo Larochelle, Joelle Pineau, and Laurent Charlin. Language gans falling short. *arXiv preprint arXiv:1811.02549*, 2018.

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

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

Ian J Goodfellow. On distinguishability criteria for estimating generative models. *arXiv preprint arXiv:1412.6515*, 2014.

Aditya Grover, Jiaming Song, Alekh Agarwal, Kenneth Tran, Ashish Kapoor, Eric Horvitz, and Stefano Ermon. Bias correction of learned generative models using likelihood-free importance weighting. *arXiv preprint arXiv:1906.09531*, 2019.

Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, and Jun Wang. Long text generation via adversarial training with leaked information. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Jonathan Ho, Xi Chen, Aravind Srinivas, Yan Duan, and Pieter Abbeel. Flow++: Improving flow-based generative models with variational dequantization and architecture design. *arXiv preprint arXiv:1902.00275*, 2019.

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

Matt J Kusner and José Miguel Hernández-Lobato. Gans for sequences of discrete elements with the gumbel-softmax distribution. *arXiv preprint arXiv:1611.04051*, 2016.

Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541*, 2016.

Sidi Lu, Lantao Yu, Weinan Zhang, and Yong Yu. Cot: Cooperative training for generative modeling of discrete data. *arXiv preprint arXiv:1804.03782*, 2018.

Mohammad Norouzi, Samy Bengio, Navdeep Jaitly, Mike Schuster, Yonghui Wu, Dale Schuurmans, et al. Reward augmented maximum likelihood for neural structured prediction. In *Advances In Neural Information Processing Systems*, pages 1723–1731, 2016.

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

Art B. Owen. *Monte Carlo theory, methods and examples.* 2013.

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

Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *arXiv preprint arXiv:1701.05517*, 2017.

Masashi Sugiyama, Taiji Suzuki, and Takafumi Kanamori. Density-ratio matching under the bregman divergence: a unified framework of density-ratio estimation. *Annals of the Institute of Statistical Mathematics*, 64(5):1009–1044, 2012.

James Townsend, Tom Bird, and David Barber. Practical lossless compression with latent variables using bits back coding. *arXiv preprint arXiv:1901.04866*, 2019.

Ryan Turner, Jane Hung, Yunus Saatci, and Jason Yosinski. Metropolis-hastings generative adversarial networks. *arXiv preprint arXiv:1811.11357*, 2018.

Masatoshi Uehara, Issei Sato, Masahiro Suzuki, Kotaro Nakayama, and Yutaka Matsuo. Generative adversarial nets from a density ratio estimation perspective. *arXiv preprint arXiv:1610.02920*, 2016.

Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. Texygen: A benchmarking platform for text generation models. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 1097–1100. ACM, 2018.