# Better Text Generation with MLE-GAN

**Elad Amrani†, Noam Yefet‡ and Kira Radinsky‡**
†Faculty of Electrical Engineering
‡Faculty of Computer Science
Technion Israel Institute of Technology
Haifa, Israel 3200003
{elad.amrani,snyefet}@campus.technion.ac.il
kirar@cs.technion.ac.il

## Abstract

Generating text is an important task in many Natural Language Processing applications, such as machine translation, text summarization, image captioning, etc. Although successful, common methods suffer from a number of problems. In this paper, we present a new model called MLE-GAN for alleviating those problems. By combining the strength of a Maximum Likelihood model with the advantages of a Generative Adversarial Network we are able to demonstrate a substantial improvement compared to a Maximum Likelihood baseline model.

## 1 Introduction

Text generation is an important task in Natural Language Processing. It plays a key role in many applications, such as machine translation (Cho et al., 2014), text summarization (Nallapati et al., 2016), image captioning (Xu et al., 2015), question answering (Moritz Hermann et al., 2015), etc. Most previous work focuses on supervised settings and a common approach is to train a recurrent / convolutional neural network (RNN / CNN) to maximize the log predictive likelihood (MLE) of each true token in the training sequence given the previous observed tokens (Zaremba et al., 2014).
Although being very successful, they suffer from a number of problems. One such problem is the so-called exposure bias during inference (Bengio et al., 2015). In the inference stage, true previous target tokens are unavailable, and are thus replaced by tokens generated by the model itself, yielding a discrepancy between how the model is used in training and inference. The main problem is that mistakes made early in the sequence generation process are fed as input to the model and can be quickly amplified. A second problem with MLE, is the fact it rewards the model for producing high frequency word combinations. This makes the generated text repeated and prevents the model from producing low frequency but meaningful words.
To tackle these problems, we propose a new model called MLE-GAN[1]. The key idea is to combine the strength of MLE with the advantages of a Generative Adversarial Network (GAN). GAN proposed by Goodfellow et al. (2014), leverages a discriminative model D, that aims to estimates the probability that a sample came from the training data rather than the generative model G. The training procedure for G is to maximize the probability of D making a mistake. This framework has shown great promise for alleviating the above problems (Yu et al., 2017). Optimizing for MLE and GAN objective functions in parallel (i.e., MLE-GAN) instead of using a GAN only model, helps to get rid of bad local minimums, while enjoying the benefits of a GAN training scheme. The analysis of the experimental results show that MLE-GAN model demonstrates substantial improvement over the baseline model (MLE).

## 2 Related Work

Recurrent Neural Network (RNN) is the most popular way for text generation. The most used training scheme of this architecture is *teacher forcing*. In this training scheme, in inference, the output of the time step $i$ is the input of time step $i + 1$; and in training, the input of time step $i + 1$ is being replaced by the ground truth. However, as mentioned above, this solution is vulnerable to exposure bias.
GAN has been applied successfully for computer

[1]Code for our model and evaluation methods is available at https://github.com/elad-amrani/mle_gan

vision tasks in recent years (Goodfellow et al., 2014). Nonetheless, it still remains a challenging task for text generation since GAN was designed for generating real valued continuous data. Deriving the gradient of the loss of D w.r.t the outputs of G is impossible when the output of G is based on discrete tokens. A number of possible solutions were suggested by researchers. One such solution is suggested by Rajeswar et al. (2017). They use RNN for both the Generator and the Discriminator. The fake input of D at each time-step is the probability distribution generated by G instead of a discrete token. A second solution is suggested by Kusner and Hernández-Lobato (2016), that is based on Jang et al. (2016) and one which we adopt in our model as well. They replace the regular Softmax function with the Gumbel-Softmax estimator, a continuous distribution over the simplex that can approximate samples from a categorical distribution.

*Professor forcing* is introduced by Lamb et al. (2016). It acts as a regularizer for recurrent networks. They create a teacher forcing model (RNN trained by *teacher forcing*) and a "free-running" model (inputs are self-generated), that both have shared variables. D is trained to maximize the likelihood of correctly classifying a sequence as generated by the teacher forcing model or the "free running model". The suggested scheme is an alternative way of training RNNs which explicitly seeks to make the generative behavior and the teacher-forced behavior match as closely as possible.

Press et al. (2017) use the GAN objective function for a character-based model. They use *teacher helping* - G learns to generate long sequences by conditioning on shorter ground truth sequences; *curriculum learning* and *variable length* - slowly teaching the model to generate sequences of increasing variable length. These extensions are helping G to converge and to generate more meaningful text.

Yu et al. (2017) bypasses the Generator differentiation problem by directly performing gradient policy update. The reinforcement learning reward signal comes from the Discriminator judged on a complete sequence, and is passed back to the intermediate state-action steps using Monte Carlo search.

Our work is distinct in that we use MLE training method in parallel with GAN training method.

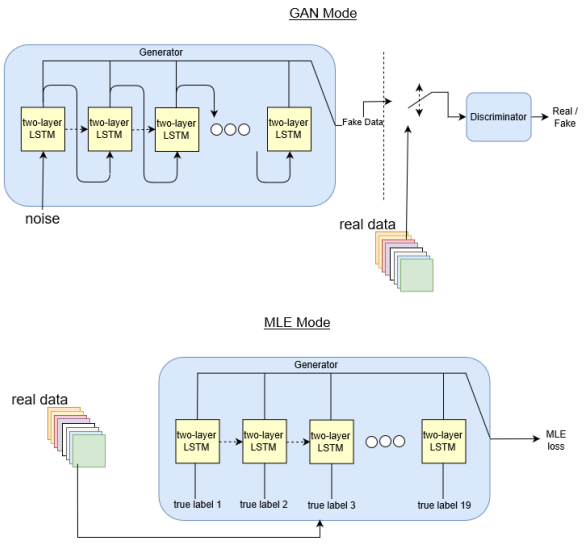


Figure 1: MLE-GAN. The Generator is trained in parallel in GAN mode and in MLE mode.

GAN helps to alleviate the bias exposure problem (as demonstrated in previous work), while MLE training in parallel helps with the difficulties of training a discrete token based GAN.

## 3 Approach

MLE-GAN (Figure 1) is composed of two models: a generative model (G) that generates text and a discriminative model (D) that discriminates between the generated text and the real text. MLE-GAN optimizes for two objective functions (GAN mode and MLE mode) in parallel.

### 3.1 Generator

To generate text, we use a two-layer Long Short-Term Memory unit (LSTM) (Hochreiter and Schmidhuber, 1997) unrolled for 20 steps based on Zaremba et al. (2014). In GAN mode, the input of the first time-step is a normally distributed noise vector, and for the $i$ time step ($i > 1$), the input is the $i-1$ output, multiplied by embedding matrix. In MLE mode the input for each time-step is a one hot vector that represents real data, multiplied by the same embedding matrix as above. The Gumbel-Softmax (Jang et al., 2016) is computed for every output step and is used as the input of the discriminative model (as well as input of the generator itself in GAN-MLE mode) during training. The embedding matrix is initialized with the embedding matrix of a pre-trained MLE model.

| | MLE | MLE-GAN | % | p-value |
|---|---|---|---|---|
| Perplexity | **133.803** | 137.546 | +2.7974 | – |
| %-in-test-1 | 0.8919 | **0.9085** | **+1.8612** | $< 10^{-6}$ |
| %-in-test-2 | 0.3841 | **0.4459** | **+16.0896** | $< 10^{-6}$ |
| %-in-test-3 | 0.0942 | **0.121** | **+28.4502** | $< 10^{-6}$ |
| %-in-test-4 | 0.0232 | **0.0269** | **+15.9483** | 0.1389 |
| BLEU-1 | 0.9364 | **0.9499** | **+1.4417** | $< 10^{-6}$ |
| BLEU-2 | 0.6193 | **0.6678** | **+7.8315** | $< 10^{-6}$ |
| BLEU-3 | 0.3148 | **0.3642** | **+15.6926** | $< 10^{-6}$ |
| BLEU-4 | 0.1534 | **0.1773** | **+15.5802** | $< 10^{-6}$ |
| Dist. 1 | **3015** | 2703 | -10.3483 | – |
| Dist. 2 | 3484 | **3560** | **+2.1815** | – |
| Dist. 3 | 1118 | **1352** | **+20.9303** | – |
| Dist. 4 | 284 | **335** | **+17.9578** | – |

Table 1: Performance comparison on PTB test dataset. For *perplexity*, lower is better. For *%-in-test-n*, *BLEU-n* and *Dist. n* higher is better. See section 4.3 for detailed explanation.

### 3.2 Discriminator

The Discriminator is a simple fully connected (FC) layer. The input, whether it is a one-hot vector for real data or Gumbel-Softmax vector for fake data (from G) is multiplied by an embedding matrix and concatenated to be used as input for the FC layer. The embedding matrix is initialized with the embedding matrix of a pre-trained MLE model. The single neuron output of the FC layer predicts 0 for fake sentence or 1 for a real sentence.

### 3.3 Overview

D and G play a two-player minimax game with value function $V(D, G)$:

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

In parallel, G also minimizes the average negative log probability of the target tokens (of real data) given the previous observed tokens:

$$\text{MLE Loss} = -\frac{1}{N}\sum_{i=1}^{N} \ln p(y_i|y_{i-1}, ..., y_1) \quad (2)$$

Where $y$ is the target token, and $N$ is the number of tokens in the sequence. We alternate between optimizing G and D every K steps. Scaling the importance of each objective function (GAN Generator, GAN Discriminator, and MLE) is done by applying different learning rates for each objective function.

## 4 Experiment

We evaluate MLE-GAN model by comparing its results to a standard MLE baseline model. We introduce the dataset, training details, and evaluation metrics.

### 4.1 Dataset

**Penn Tree Bank (PTB) Dataset** (Marcus et al., 1993). This dataset consists of 929k training words, 73k validation words, and 82k test words. It has 10k words in its vocabulary.

### 4.2 Training Details

We use a word-level architecture. Number of time steps is set to 20. Sentences longer than 20 words are split between two or more samples. Embedding size is set to 200 (for both G and D). Vocabulary size is set to 10k. Batch size is set to 20.
For MLE-GAN model we use Adam optimization method (Kingma and Ba, 2014) with the following learning rates: 2e-3 for Generator (both MLE and GAN), 5e-4 for Discriminator. $\tau$ (Gumbel-Softmax Jang et al., 2016[5] parameter) is set to 0.1.. We train D for K batches, and then G for K batches, etc. K is set to 40. G input noise is a normally distributed random noise with mean 0 and variance 1.
For MLE baseline model we use Adam optimization method with an initial learning rate of 1. The learning rate is decreased by a factor of 2 manually every 4 epochs. Both models were trained until convergence.

| | Examples |
|---|---|
| **MLE** | 1. the collective alliances which still show no strong 's income from its recent courses in bankers <eos> sales of the<br>2. the child-care executives who meant that <unk> <unk> is going to get up the critics yet again <eos> mr. lucky<br>3. the commodities dr. lawson branch is n't coming all within the book <eos> mr. bullock the plane went perfectly in<br>4. the general african sales closed us $ N million yesterday <eos> the first N fuji only earned profitability in the second<br>5. the recognition accounts digital is <unk> about $ N million on the forecasts of the company under u.s. auto sales<br>6. the wall list after problem the final shot on the bay area ¡eos¿ franco news <eos> jacobs companies are planning<br>7. the issuers edged through for these kong 's rally from general motors and totals the canadian mortgage association <eos> osha<br>8. the chemical company 's assets in <unk> communications inc <eos> but its gain from the near east german 100-share contract |
| **GAN-MLE** | 1. the universal angeles chamber in nevada <unk> college in europe cathay is subject to its accelerating production and the end<br>2. the consensus season would <unk> to <unk> ratio by aug. N <eos> the four four previous details had been already<br>3. the experts and the <unk> who acquired the american magazine in stake and everything the largest problem bears money namely<br>4. the financial news that continue named <unk> stepped for it easy for the leadership <unk> campbell equipment inc. are to<br>5. the company continues to start up next operating from $ N million for the projects <eos> still dow jones has<br>6. the can <unk> ball <eos> the maryland world previously recalled for college patients in houston were about over the moment<br>7. the department of deteriorating part or more volume says <eos> N a.m <eos> the deviation plunge was yielding about half<br>8. the rules have been cheered by visitors it has <unk> <eos> the problem after cancer mr. founder shall veto dividends |

Table 2: Generated examples from both models. <eos> stands for 'end of sentence', <unk> stands for unknown, i.e., word not in vocabulary. Examples were sampled randomly for a fair comparison.

### 4.3 Evaluation Metrics

We use 4 evaluation metrics: *perplexity*, *%-in-test-N* (the proportion of word n-grams from generated sequences that also appear in a held-out test set), *corpus-level BLEU score* (Papineni et al., 2002), and *distinct n-grams* (that also appear in a held-out test set). For the latter three, we compute the result for a 20-word sentence and average over 1000 sentences. Sentences are generated with a single word seed ('the'). The results are summarized in Table 1. Generated examples are in Table 2. We were able to improve on (almost) all metrics, with the price of a slightly higher perplexity. However, there is an exception and that is the distinct-1 score. Yet, it only means that the MLE-GAN model generates better and more diverse sentences with less words (i.e., smaller vocabulary) than the baseline model.

## 5 Conclusions

Although successful, common text generation methods suffer from a number of problems. In this work we presented a simple yet effective model called MLE-GAN for alleviating those problems. We compared the model to MLE baseline model and showed substantial improvement over a number of relevant evaluation metrics.

## References


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.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, Dzmitry Bahdanau, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoderdecoder for statistical machine translation. *arXiv preprint arXiv:1406.1078v3*.

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.

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

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

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

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

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.

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.

Karl Moritz Hermann, Tom Koisk, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. *arXiv preprint arXiv:1506.03340*.

Ramesh Nallapati, Bowen Zhou, Cicero Nogueira dos santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Computational Linguistics (ACL)*, pages 311–318.

Ofir Press, Amir Bar, Ben Bogin, Jonathan Berant, and Lior Wolf. 2017. Language generation with recurrent generative adversarial networks without pre-training. *arXiv preprint arXiv:1706.01399*.

Sai Rajeswar, Sandeep Subramanian, Francis Dutil, Christopher Pal, and Aaron Courville. 2017. Adversarial generation of natural language. *arXiv preprint arXiv:1705.10929*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. *arXiv preprint arXiv:1502.03044*.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI*, pages 2852–2858.

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