# Conditional MaskGAN and evaluation via classification

**Hanoz Bhathena**
SCPD student
Stanford University
*hvb2108@stanford.edu*

**Renke Cai**
SCPD student
Stanford University
*Renke.cai@stanford.edu*

## Abstract

We built an effective generative model for text using Generative Adversarial Networks (GANs) and deep reinforcement learning. Our task is as follows: we take a text input and randomly mask certain number of consecutive words in it. Then using that masked sequence as input we to fill in the blank (masked) values. We first pre-trained weights using a classical language model and the trained generator and discriminator independently. After that we trained the full GANs using reinforcement learning algorithms. We proposed one adjustment to the model, which is adding a sentiment embedding to both the generator and discriminator. Moreover, as a downstream task, we utilized samples from the generators as a synthetic data augmentation technique to performed sentiment classification on IMDB review dataset.

## 1 Introduction

Recent years have seen great progress being made in the field of language generation. From machine translation to text summarization to dialogue systems language generating systems have become an integral component of several modern AI systems. Perhaps the most successful framework currently in use is the Sequence to Sequence architecture first introduced by Sutskever et al. Since then there have been many innovations to this architecture, notable ones like adding an attentional mechanism as first introduced by Bahdanau et al, or using deep reinforcement learning as shown in multiple papers like Rennie et al, Paulus et al and Ranzato et al.

However in the broader scope of the deep learning community there has been a similar explosion of interest in Adverserial learning primarily models from the family of Generative Adversarial Networks (GANs) pioneered by Goodfellow et al. GANs have been used extensively for image generation and are now a mainstay in the field. However, their use in the application domain of natural language has until recently been largely cautious. The reason behind this is the discrete nature of language and the implicit differentiability constraints that they bring to deep learning which relies on backpropagation.

The key idea behind our project is inspired by a recent published paper 'maskGAN: Better Test Generation via filling in the ____' by William Fedus, Ian Goodfellow and Andrew M. Dai. They demonstrated effective use of GANs for a language generation task by using the REINFORCE algorithm to tackle the differentiability issue that arises due to discrete language.

Our goal in this project was to learn more about the underlying techniques behind their model, try to re-implement it under certain simplifying assumptions, proposing how

performance changes under certain architectural changes and finally use generated samples in certain downstream tasks and propose additional ways for evaluation of generated samples in addition to what was discussed in their paper.

Given the shorter time frame and how crucial fine-tuning such type of models are (having both reinforcement learning and GAN components) our goal was not to beat their benchmark results. Instead, we tried some adjustment, evaluated model performance quantitatively and qualitatively and studied the merits/demerits of different neural network architecture and training algorithms. Also, since one issue with supervised learning is it requires a large amount of labeled data, we used the model generated samples as a data augmentation for classification.

Our additional contributions can be summarized in the below points:

1. Quantitative and qualitative evaluation of complementing the MaskGAN model with label information in the form of trainable embeddings. Our hypothesis is that adding this information would help the model to adhere better to global context while filing in the blanks. Otherwise, as the authors also show in their paper, the text filed in the blanks is locally coherent but not globally and has an especially hard time at forward looking boundaries.
2. The original MaskGAN employs the actor-critic algorithm from deep reinforcement learning literature. After three stages of pre-training the model is trained using adversarial learning in free-running mode. We faced significant issues in replicating this part as even after employing the same three phases of pre-training when training the final GAN using only REINFORCE loss, the perplexity very quickly diverged. To prevent this we tried out two approaches of supplementing REINFORCE loss with cross entropy as described below.
3. Look at both GAN generated samples as an artificial data augmentation technique. Data augmentation is a key component in image classification, given it makes sense that an image of a dog rotated, slightly blurred or cropped would still be a dog. Language however is much more complex as changing the order of words can somewhat to greatly alter the meaning of the text especially in a sentiment classification context.
4. Finally we propose a new (to our best knowledge) approach to evaluate generative models for language when there are classification labels at hand for the text.

## 2 Background and Related Work

Recurrent Neural Networks (RNNs) (Graves et al., 2012) are the most common generative model for sequences as well as for sequence labeling tasks. They have shown impressive results in language modeling (Mikolov et al., 2010), machine translation (Wu et al., 2016) and text classification (Miyato et al., 2017). Text is typically generated from these models by sampling from a distribution that is conditioned on the previous word and a hidden state that consists of a representation of the words generated so far. These are typically trained with maximum likelihood in an approach known as teacher forcing, where ground-truth words are fed back into the model to be conditioned on for generating the following parts of the sentence.

Neural text generation models are often autoregressive language models or seq2seq models. These models generate text by sampling words sequentially, with each word conditioned on the previous word, and are state-of-the-art for several machine translation and summarization benchmarks. The original paper mentions that these benchmarks are often defined by validation perplexity even though this is not a direct measure of the quality of the generated text. Additionally, these models are typically trained via maximum likelihood and teacher forcing. These methods are well-suited to optimizing perplexity but can result in poor sample quality since generating text requires conditioning on sequences of words that may have never been observed at training time. Therefore the authors introduce a model using GANs and show that it produces more realistic conditional text samples compared to other method.

However, GANs were originally designed to output differentiable values, so discrete language generation is challenging for them. Research into reliably extending GAN training

to discrete spaces and discrete sequences has been a highly active area. GAN training in a continuous setting allows for fully differentiable computations, permitting gradients to be passed through the discriminator to the generator. Discrete elements break this differentiability, leading researchers to either avoid the issue and reformulate the problem, work in the continuous domain or to consider RL methods. For the detailed discussion please refer to the original paper.

## 3 Approach

The core task of our generative model is, given text with certain contiguous words randomly masked out, our model should be able to fill in the blanks. The goal is not to exactly replicate the original sentence in fact the opposite. The ideal situation would be that the model can correctly understand the context of words that came before and after the sequence of blanks and in fill a completely different sequence of words that maintains both global coherence in a language modeling sense but also in a thematic sense. Our idea of supplementing the model's encoder with sentiment embeddings are in an effort to induce more information about the global context.

Like any GAN, our model consists of a generator and a discriminator. Both the generator and the discriminator are a sequence to sequence neural architecture. For the unconditional GAN the inputs to the encoder section of both the generator and discriminator are the same. The are basically the word ids {x(1),…,x(T)} where a random mask {0,1} is applied to each word id. If the mask is 1 then m(x(t))= x(t) else m(x(t))= "<M>" where "<M>" is a special masking token with its own trainable embedding. The authors state that giving the generator's encoder input to the discriminator's encoder is crucial as otherwise the model suffers from a critical failure mode. For example, if the generator generates the sequence *the director director guided the series* that is fed to the discriminator, the discriminator, to provide an accurate score must know the true context, for example whether *the *associate* director guided the series* or *the director *expertly* guided the series* is the real sequence as both are valid English sentences. Finally, our Conditional MaskGAN adds an additional sentiment embedding as the initial hidden state to the encoders of both the generator and the discriminator and this embedding is a learnable parameter. It is possible to share the embedding but we keep it different for the encoder and decoder as it the generator might want to learn a separate representation for generating a globally thematic sequence while the discriminator might require a slightly different representation of sentiment for evaluating generator output.

On the decoder side the generator and discriminator are different. For the generator we use different types of decoders: a teacher forcing decoder where the decoder gets the target as input at each step and an actor-critic decoder which generated in free-running mode where the current word input is the one generated from one step ago by the decoder. In fact, since we are not looking to generate an entire sequence but only fill in the masked out parts, we need not run the entire decoder in free-running mode. If the current input is masked out the decoder cell receives the previously generated word as input (as its ground truth is masked), and if the word is not masked it simply receives the ground truth word. As in traditional language modeling the target sequence is the input sequence offset by one time step. Given this is the nature of our algorithm, we make sure that the first word is never masked as there would be no previously generated word to feed as input. The decoder therefore generates a probability distribution over words in the vocabulary at each step from which a word is sampled during the generation phase.

$$G(x_t) := P(\hat{x}_t | \hat{x}_1, \dots, \hat{x}_{t-1}, \boldsymbol{m}(\boldsymbol{x}))$$

On the discriminator decoder side the inputs are the output sequence ids from the generator's decoder. These inputs are fed into the decoder in typical teacher forcing form and instead of outputting a distribution over words the decoder instead outputs a scalar score which (after taking a sigmoid over) we consider to be the probability that the current word from the generator is real.

Multiplicative attention is used in the decoder for the generator as well as the discriminator. This is crucial for providing forward looking context to the decoder. We also implemented

intratemporal attention models for the encoder and decoder side as described in Paulus et al but didn't have enough time to test those. We expect sample coherence to improve with that especially given the attention over the decoder part as the model might benefit from more global information on what it has already filled in.

The log probability score from the generator is what we consider to be the reward signal to the generator at step *t*. This conforms to the concept of the GAN training mechanism i.e. the generator which is a reinforcement learning agent will try to maximize its reward and the reward is the probability of being a real word as judged by the discriminator. Therefore, while the objective of the discriminator is to minimize this reward as it is an incorrect classification of fake data as real if it is high the generator RL agent will try to maximize it using the REINFORCE algorithm.

$$D_\phi(\tilde{x}_t|\tilde{x}_{0:T}, \boldsymbol{m}(\boldsymbol{x})) = P\left(\tilde{x}_t = x_t^{\text{real}}|\tilde{x}_{0:T}, \boldsymbol{m}(\boldsymbol{x})\right)$$

$$r_t := \log D_\phi(\tilde{x}_t|\tilde{x}_{0:T}, \boldsymbol{m}(\boldsymbol{x}))$$

$$\nabla_\theta \mathrm{E}_G[R_t] = (R_t)\nabla_\theta \log(G_\theta(\hat{x}_t))$$

As is the case for practical situations of the REINFORCE algorithm, the gradients above have high variance and do not converge in practice. Therefore, we require a third network, a critic network which is simply a linear projection from the discriminator GRU hidden state to a scalar. The function of the critic is to estimate (via regression) the value function which is the discounted total return of the filled in sequence. If this value is subtracted from the rewards multiplier to the grad log probabilities of the generator the algorithm will converge much more smoothly.

We therefore have the below update rules for the generator and discriminator, respectively:

$$\begin{aligned}\nabla_\theta \mathrm{E}[R_t] &= \mathrm{E}_{\hat{x}_t \sim G}\left[\sum_{t=1}^{T}(R_t - b_t)\nabla_\theta \log(G_\theta(\hat{x}_t))\right] \\ &= \mathrm{E}_{\hat{x}_t \sim G}\left[\sum_{t=1}^{T}\left(\sum_{s=t}^{T}\gamma^s r_s - b_t\right)\nabla_\theta \log(G_\theta(\hat{x}_t))\right]\end{aligned}$$

$$\nabla_\emptyset \frac{1}{m}\sum_{i=1}^{m}\left[\log D\left(x^{(i)}\right) + \log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right)\right]$$

### 3.1 Language model pre-training

The training for the generative model includes three phases of pre-training: language model pre-training, generator pre-training and discriminator training which is then followed by GAN training.

Multiple stages of pre-training are required as the action space for the RL agent is quite large compared to typical RL tasks, i.e. the entire vocabulary. Additionally, given the minimax game nature of GAN training a delicate balance is required otherwise the training will diverge quickly.

The first stage of pretraining is a vanilla language model. The words in the sequence are fed into an encoder only network and the target at each step is the next word. It is important to note that no masking is applied at this stage. Cross entropy loss is optimized and we train our language model until a dev set perplexity of 86.92.

### 3.2 Pre-training on generator and discriminator

After the language model is trained we load the GRU weights into the encoder and decoder portions of the generator while we initialize the attention weights and sentiment embeddings from scratch using Xavier initialization. We do the same for the discriminator. We train the generator till it achieves a dev perplexity of 92.9. We pre-train the discriminator till it achieves an accuracy of correctly classifying real versus fake generated samples (inference mode) of 98.5%.

Note: The perplexity, losses and accuracies are all on the masked out portions of the sequence only.

### 3.2 GANs training

Finally, after pretraining we start to train the GAN using actor-critic with REINFORCE. In our experience this part was by far the hardest to train even after nearly replicating the benchmark numbers of the original authors at the pre-training stages. We attribute this both subtle differences in the implementations but much more so to the hyperparameter tuning and other architectural fine tuning which would have taken a long time to discover. Furthermore the authors state in their paper to have employed a curriculum learning approach to their training in addition to using reward signals from not only the sampled word id but all words in the vocabulary. We did not go down this route and therefore simply applying REINFORCE showed that despite the rewards increasing steadily, perplexity quickly rose beyond repair.

We therefore tried out two other approaches which helped the training converge more.

1. In the first one we simply add in a cross entropy component to the generator loss function using typical teacher forcing at training time in addition to the reinforce loss. This follows from similar ideas proposed in Ranzato et al and Paulus et al, i.e. the MIXER algorithm. The above algorithm performed well on shorter masked out sequences (5-10 words) but underperformed on longer ones. To be sure, the MIXER algorithm as employed by Ranzato also required a sufficient amount of fine tuning in the form of annealing the mixing ratio and also employing cross entropy on a decreasing portion of the sequence as epochs progressed, effectively also a curriculum learning approach.
2. We therefore tried a different approach which we term Teacher REINFORCE. In this training approach we simply apply teacher forcing to the generator decoder. But when we generate the words we do not use sampling from the vocabulary distribution which is non-differentiable but instead a max pooling operation over vocabulary logits as the gradients would flow through the position with the maximum logit value. However, the argmax function (to get the max position word id) is not differentiable and so we do not leave continuous space (decoder hidden output vectors) and instead of outputting the word id and indexing into the word embedding as input to the decoder, we project the hidden state space into the embedding space and pass that into the decoder as described above in teacher forcing mode. The rewards are calculated the same way by the discriminator and are differentiable wrt generator weights.

## 4 Experiments

### 4.1 Dataset

We use IMDB reviews as our dataset for training, testing and downstream sentiment classification task. We have collected the IMDB data by downloading it from the Stanford website. IMDB reviews have 75,000 records for train and dev set (among which 25,000 are labeled and 50,000 are unlabeled), 25,000 records for testing. GloVe is used to generate word embeddings and we tentatively use the embedding size of 300, cap the sentence length at 40 and cap the vocabulary size at 10k.

### 4.2 Experiment

Details of the exact steps of the language generation component of our project were provided

above. Below we list the hyperparameter settings we used.

| Hyperparameters | Values |
|---|---|
| Generator Learning rate | 0.0005 |
| Disc Learning rate | 0.005 |
| Critic Learning rate | 0.005 |
| GRU Layer number | 2 |
| Hidden state size | 650 |
| Dropout rate | 0.5 |
| Reward discount rate | 0.9 |
| Embedding Dimension | 300 |
| Batch size | 64 |

We were not able to complete the entire reinforcement learning training but on testing on a small sample size (64), we saw the train perplexity quickly drop to 16 while the train discriminator loss also went to nearly zero (accuracy nearly 100%) indicating that our model implementation was correct. However, we could only train on the entire dataset for 4hrs and achieved a train perplexity of 350 and steadily decreasing. It is possible that with the current hyperparameter configuration and more time our model would have converged to be in line with the author's results. In fact Pavlov et al, demonstrate the trajectory of improvement for their learning to run reinforcement algorithm. In figure 2 of the paper, there is slow to no improvement until 8 hours, then an 8 hour period of extreme ramp up and then finally a saturation after 20 hours. Given our action space and parameter estimation space is much larger but also the offsetting facts that we have more pretraining done and a guiding loss function from the cross entropy we would expect around a similar or slightly more time for convergence if the hyperparameters were selected correctly. We selected our hyperparameters by manually running different variations on the small set of 64 training examples and based on the speed of convergence of the generator and discriminator.

We consistently observed that if a pre-trained discriminator was used to kick off GAN training even if given lower learning rate, it would eventually catch up and become very good at classifying real versus fake. Therefore this would potentially subdue the reward signal to the generator and therefore we would in the future construct a learning rate ramp up for the REINFORCE gradients and a learning rate decay for the cross entropy gradients.

### 4.3 Evaluation metrics

Given that we did not fully complete training of a GAN model, we instead chose to samples to evaluate in a different way. We trained a generator and discriminator by initiating with a plain language model (no generator or discrimator specific pre-training). Then instead of stopping when the dev perplexity did not get any better for some number of epochs we instead stopped when the accuracy of the discriminator to disambiguate real versus fake samples hit 50% which signified that a (semi)-trained discriminator could not distinguish between generated and real samples.

To evaluate these samples we take a quite different approach from the authors who focus on Amazon Mechanical Turk and unique n-grams (due to mode collapse issue faced by GANs).

Instead we choose to evaluate samples by two classification related methods:

1. We train a classifier, first on real labeled data only and then on real plus GAN generated labeled data and evaluate if the additional data made the classifier better.

Our classifier is a simple GRU based encoder which does concatenates the max-pooled of hidden states, average pooled of hidden state and final states to predict positive versus negative sentiment of IMDB reviews. Deep learning algorithms are known to generally improve given more labeled data however, the quality of data should also be good. Generally quality of additional data would be judged as decreasing overall training data quality if non-qualified human labelers were used to generate noisy labels. However, in this case the ground truth label is pristine but the sample may or may not be aligned. Our results on training with real only and real and fake data are shown below. A classifier trained on real only data achieved 75% whereas a classifier trained on real and fake data achieved 72% test accuracy.

2. Second method is to use a trained classifier, trained on real only data to evaluate the classes of generated data. Assuming we have a well-trained classifier with near perfect test accuracy we can safely assume that the prediction accuracy of the classifer is a good indicator of the efficacy of the generative model. For example say we have a classifier with 95% test accuracy. Then if the actual ground truth label is a 0 and the classifier predicts a 1 then the probability that the classifier is wrong and the generated sample is right (conforms to the sentiment theme and generally coherent for a language model) is 5% but 95% the generated sample was bad. Therefore we build a simple classifier on real data and test the accuracy on the generated data. A classifier with 75% test set accuracy judged that 50% of the generator samples were correctly classifiable. This conforms with point (1) above where the classifier accuracy decreased as the sample quality was still not upto the level of the human generated output.

The hyperprameter settings are as follows:

| **Hyperparameters** | **Values** |
|---|---|
| Learning rate | 0.0005 |
| GRU Layer number | 2 |
| Hidden state size | 640 |
| Dropout rate | 0.5 |

We also look at the quality of the generated sample and here are examples generated by pretrained conditional/unconditional generator and GANs, where we give both good and bad examples in both cases.

Pretrained unconditional generator (good samples):

| | |
|---|---|
| **Ground Truth** | single character is either fat , stupid , unable to communicate , unable to enjoy life or a combination of the <OOV> mentioned . This movie goes out of its way to <OOV> in the ways in which communication can |
| **Generated Sample** | single character is either fat , stupid , unable to communicate , unable to enjoy life or a combination of the <OOV> mentioned . It is one of the most confusing movies that I have ever seen . The acting |

Pretrained unconditional generator (bad samples):

| | |
|---|---|
| **Ground Truth** | what I had heard , this was in no way a sequel to City of God , post viewing I have learned that it is a follow-up to a TV series . Still , this film will be constantly compared |
| **Generated Sample** | what I had heard , this was in no way a sequel to City of God , post viewing <OOV> ( Note to say , 1937 not so much any <OOV> outfits . Not a book nor be constantly compared |

Pretrained conditional generator (good samples):

| | |
|---|---|
| **Ground Truth** | this movie in the theater when it came out in <OOV> just a fun to watch romp set in <OOV> that the whole family can watch is it full of state of the art special effects ? no but they |
| **Generated Sample** | this movie in the theater when a couple of friends told me most of the movie . I loved it . <OOV> <OOV> a few watch is it full of state of the art special effects ? no but they |

Pretrained conditional generator (bad samples):

| | |
|---|---|
| **Ground Truth** | was always going to be a hard novel to adapt - the very qualities that make it a great read make a confusing film . The book has a mysterious , <OOV> , <OOV> quality - <OOV> can slip over |
| **Generated Sample** | was always going to be a hard novel to adapt - the <OOV> style by a script book , <OOV> physical there as a lack of political supernatural camera , <OOV> , <OOV> quality - <OOV> can slip over |

Conditional GAN (good samples):

| | |
|---|---|
| **Ground Truth** | got this movie , primarily to see how it 's soundtrack ( which is great ) related to the storyline . I was shocked to see how good the movie was ! The movie is <OOV> , and kept my |
| **Generated Sample** | got this movie , primarily to see how it came purely on stage because it was my favorite film ever made . I saw this world after seeing the movie was ! The movie is <OOV> , and kept my |

Conditional GAN (good samples):

| | |
|---|---|
| **Ground Truth** | I rented The <OOV> ( 2000 ) with my family and could not even finish watching the whole thing , and it was only 88 minutes long ! Director Michael Dinner needs to stick to TV . Voice <OOV> worked great |
| **Generated Sample** | I rented The <OOV> ( 2000 ) with my family and could not even decide my attention to it . The cast was believable , and the movie were certain has really fun stick to TV . Voice <OOV> worked great |

Conditional GAN (good samples):

| | |
|---|---|
| **Ground Truth** | was lucky to see <OOV> <OOV> in its full feature length version at the Dallas Video Fest . I hear there is a shorter version which will be broadcast on TV . Quite frankly , I have never seen a |
| **Generated Sample** | was lucky to see <OOV> <OOV> in its full release I ca n't remember anyone really thinking this movie was a major impact like my say <OOV> be broadcast on TV . Quite frankly , I have never seen a |

Conditional GAN (bad samples):

| | |
|---|---|
| **Ground Truth** | is another of my favourite films and I never get tired of watching it again and again and always laugh at the funny scenes . I ca n't imagine anyone else in those roles than Joe Pesci and Marisa Tomei |
| **Generated Sample** | is another of my favourite films and I never get tired of the delicate humour use no business standards . There is still a Dennis himself for a part of his in |

those roles than Joe Pesci and Marisa Tomei

## 5 Conclusion

We have demonstrated how global thematic context can be improved by supplementing embeddings into the MaskGAN architecture innovated by Fedus et al. This can possibly be augmented by more contextual embeddings in the future such as sentence embeddings. We also showed an alternative strategy for generative sample evaluation which is not purely perplexity driven and at same time is quantifiable and more aligned to downstream tasks than n-gram uniqueness. In the future we would like to explore architectures like the attentional transformer which could possibly help the decoder gather more relational information to fill in the blanks better.

## References


[1] Fedus, W., Goodfellow Ian. & Dai, M.A. (2018) MaskGAN: Better Test Generation via filling in the ____. ICLR 2018. https://arxiv.org/pdf/1801.07736.pdf

[2] Alex Graves et al. Supervised sequence labelling with recurrent neural networks, volume 385. Springer, 2012.

[3] Tomas Mikolov, Martin Karafiat, Lukas Burget, Jan Cernocky, and Sanjeev Khudanpur. Recurrent neural network based language model. In Interspeech, volume 2, pp. 3, 2010.

[4] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. CoRR, abs/1609.08144, 2016. URL http://arxiv.org/abs/1609.08144.

[5] Takeru Miyato, Andrew M Dai, and Ian Goodfellow. Virtual adversarial training for semi-supervised text classification. In International Conference on Learning Representations, volume 1050, pp.25, 2017.

[6] 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, pp. 2672–2680, 2014.

[7] Dzmitry Bahdanau, Kyunghyun Cho, Yoshua Bengio, Neural Machine Translation by Jointly Learning to Align and Translate

[8] Steven J. Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross, Vaibhava Goel, Self-critical Sequence Training for Image Captioning

[9] Romain Paulus, Caiming Xiong, Richard Socher, A Deep Reinforced Model for Abstractive Summarization

[10] Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, Wojciech Zaremba, Sequence Level Training with Recurrent Neural Networks