Few-shot text generation with Reptile

Yong Huang
Cornell Tech
New York, NY
yh849@cornell.edu

Yordanos Goshu
Cornell Tech
New York, New York
yag3@cornell.edu

June 8, 2020

Abstract

Deep generative models have played a pivotal role in current machine learning. With their advent, many have proposed architectures which explore and achieve efficient methods to learn complex distributions over natural language. Recent work in GANs [1] that utilize a discriminative model to guide the training of the generative model through a reinforcement learning policy have show promising results in text generation. While these models are able to generate coherent and semantically meaningful text, they rely heavily on large quantity of data. In this work, we complete a review of how the desirable properties of state-of-the-art language generation networks are impacted by unbalanced datasets. We then propose a training schema that utilizes a scalable metalearning algorithm to train a sequence based GAN in the few-shot setting. We carry out experiments on real-world tasks to demonstrate our findings. Our contributions show that Reptile can be applied as a learning strategy that empowers generative models to generate high quality text data in the few-shot setting.

1 Introduction

The task of generating sequential data that mimics the distribution over real data has been a topic of interest to researchers in the area of unsupervised learning. One of the tasks in the sequential space is that of text generation. Recent advances in recurrent neural networks (RNN) with long short term memory (LSTM) [2] have shown strong performance in tasks ranging from natural language generation to handwriten generation[3]. The RNN is used to maximize the log-likelihood of each ground-truth word given prior observed words. These methods have been shown to suffer from exposure bias. A scheduled sampling [4] approach was proposed to address this problem but was later shown to be inconsistent in its output. Then we saw the utilization of GANs to extend to discrete, sequential data to alleviate the problem above. By defining the generative network as a stochastic policy that maps current state to a distribution over the action space, the discriminator network learns to distinguish between real and generated text samples to update the generator once the whole text generation is done.

In recent years, there have been works that have investigated this GAN architecture for sequential data generation [1] [5] [6], however, to our knowledge, there has been little to no work in investigating the performance of such a network in the few-shot setting. Mainly, the previously presented work have been limited to experiments that have ample amount of data for each category of generation. For instance, in the SeqGAN experiment, the authors used 11,092 paragraphs from Obama's political speeches in order to generate synthetic political speeches. This paper considers a meta-learning problem, where there is a distribution of tasks, specifically, text generation. We then would like to learn an agent that learns quickly when presented with an unseen task sampled from this distribution. If the distribution of tasks lie close enough together (e.g. political speeches within the same language) then our agent aims to generate novel samples from the task with few points of reference.

In the real world, since high-quality data for training these tasks is not always available, we believe it is important to explore forms of efficient transfer learning architectures. With regards to text generation we anticipate applications in a broad range of work including but not limited to political speech generation, poem generation, etc.

In this work, we propose a training architecture which aims to maintain the guiding signals from the discriminator to the generator in the above proposed structure while being able to generate text structures from previously unseen distributions. To do so, we apply adaptations to the meta-learing structure presented in Reptile [7] along with the adversarial and reinforcement training presented in SeqGAN.

Our main contributions include, to our knowledge, being the first to apply meta-learning techniques to the task of few-shot sequential data generation. Previous attempts utilize a language model pretrained on a large corpus to solve the few-shot problem, however, research in meta-learning has shown that meta-learned models are more effective in generalization than pretrained models [8]. We compare the meta-trained model with the pretrained model in the experiment section. Secondly, we created a Twitter dataset that is grouped by users and topics (hashtags, search queries), we believe this dataset could be useful for other researchers. Thirdly, we implemented SeqGAN and applied it on *real* world data, specifically tweets. As opposed to all public implementations of SeqGAN which are applied on *synthetic* data generated from an LSTM oracle model which yields 'real data' as a golden language model.

2 Background

2.1 SeqGAN

Generative Adversarial Networs (GAN) [9] has been successful and been mostly applied in computer vision tasks of generating samples of natural images [10]

However, applying GANs to generate sequential data has two problems. Firstly, GANs are designed for generating real-valued, continuous data but have difficulty in directly generating sequences of discrete tokens, such as texts. This is because during generation, the dictionary space is very limited, a slight change of generation due to signals from discriminator might not map to any existing tokens. Secondly, GANs can only give the score/loss for an entire sequence when it has been generated; for a partially generated sequence, it is non-trivial to balance the quality with regards to the quality of the entire sequence within the scope of a score. [1]

Therefore, SeqGAN is proposed to address this problem. Generating text could be naturally viewed as a decision-making process, therefore we could apply some techiniques from reinforcement learning to text generation with GAN.

The task of sequence generation is defined as such: Given a dataset of real world structured sequences, train a parameterized generative model G to produce a sequence of tokens, that lie in the vocabulary of real world tokens. In the SeqGAN architecture, they interperate the generative model as an agent of reinforcement learning (RL). The intuition is at time step t, the state is the generated token token up to time t-1 and the action is the next token at time t to be generated. To provide a reward, they train a parameterized discriminative model D to guide the generator. Where D produces a probability indicating the likelihood of a sequence being from real sequence data or not. D is trained by providing positive examples as real data and fake examples as those generated from G. To solve the problem that the gradient cannot be back propogated to G when the output is discrete, they define G as a stochatic parametrized policy. Jointly, G is updated by employing a policy gradient and Monte Carlo (MC) search on the basis of the expected end reward received from D. This reward is estimated by the likelihood that it would fool D. Figure 2.1 is an illustration of SeqGAN architecture.

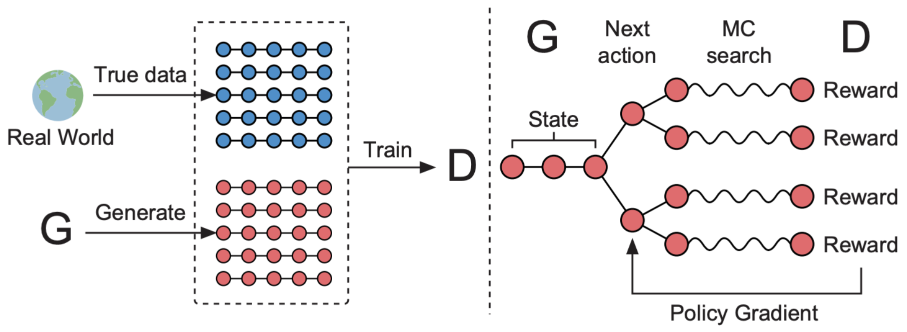
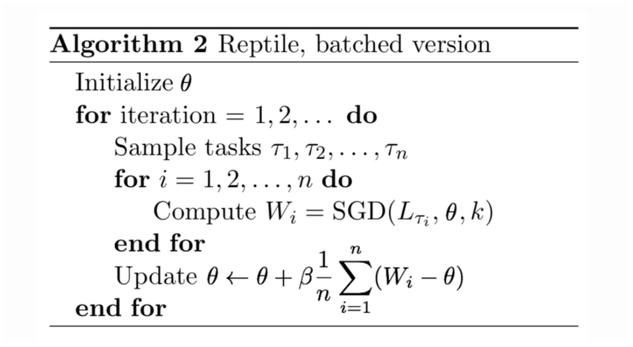


Figure 1: The illustration of SeqGAN. Figure from original paper Left: D is trained over the real data and the generated data by G. Right: G is trained by policy gradient where the final reward signal is provided by D and is passed back to the intermediate action value via Monte Carlo search.

2.2 Reptile



Algorithm 2 Reptile, batched version

```
Initialize $\theta$
for iteration $= 1, 2, \ldots$ do
    Sample tasks $\tau_1, \tau_2, \ldots, \tau_n$
    for $i = 1, 2, \ldots, n$ do
        Compute $W_i = \mathrm{SGD}(L_{\tau_i}, \theta, k)$
    end for
    Update $\theta \leftarrow \theta + \beta \frac{1}{n} \sum_{i=1}^{n} (W_i - \theta)$
end for
```

Figure 2: Batched version of Reptile, figure from original paper, SGD performs stochastic gradient update for k steps on the loss starting with initial parameter θ and returns the final parameter vector. The batch version samples multiple tasks instead of one within each iteration. The reptile gradient is defined as $(\theta - W)/\alpha$, where α is the stepsize used by the SGD operation

Optimization-based meta-learning recently emerged as an approach to learning from a small set of data. Previous work [11] has used Reptile[7] as a meta-learning framework and successfully adapts it to few shot image generation tasks. In contrast to MAML[12] which is another popular meta-learning method and its variants, Reptile does not require extra testing samples to evaluate the performance during meta-training and it also does not involve second-order derivative computation which makes it more computational effective.

3 Methods

We propose Few-shot Text Generation with REPTILE (FTGR) which consists of two main components, the sequential generative model that carries out the actual generative work and a meta-learning framework that can be interpreted as an efficient training strategy that empowers the generative model to quickly adapt to new tasks, and therefore generate high quality samples using limited number of data.

The two components under this framework are, first, the sequential data generation model which is an adaptation of SeqGAN. We will further discuss the details of our adaptation in the next section. The second component is Reptile which will control the learning and fast adaptation of SeqGAN. By combining these two, we aim at generating high quality sequential data under low-sample settings.

We applied this framework to the few-shot text generation task, namely, Twitter posts, better known as tweets. Our hypothesis claims that having few tweets from a single user (<10) and having ample amount of tweets from other users, will allow us to generate semantically meaningful tweets from the low sampled user. Our intuition of applying meta-learning methods like REPTILE is that by meta-training on other users tweets, we could pretrain a language model that captures the grammatical structures of the English language, while preserving the *idiolect* of the low sampled user.

As an example, our Twitter dataset contains 10 classes (the 10 different users). In the few-shot image generation problem, they represent 10 tasks to solve, t_0 to t_9. We choose t_0 to t_8 to be the meta-training task and t_9 to be the meta-test task. Through meta-training on t_0 to t_8, we aim to obtain a set of parameters that will quickly converge on a new t. We choose n to be 10, meaning that we aim to use our meta-trained parameters to generate tweets from user t_9 with only 10 real tweets. Details about FTGR training and generation can be found in algorithm 1 and 2.

4 Experiments

4.1 Dataset Generation

For the task of few shot text generation, we believe tweets are an ideal choice. First, different users who comes from different backgrounds usually have different style of writing and the content and topic of their tweets usually differs from each other. Secondly, tweets are by nature short with a 140 character limit. Long paragraph generation is still quite challenging and short paragraph generation is considerably easier. We create our own dataset by scraping tweets

Algorithm 1 Few-shot Text Generation with REPTILE training

```
 1: Initialize $\Phi_d$, the discriminator
 2: Initialize $\Phi_g$, the generator
 3: for iteration 1,2,3 ... do
 4:     Make a copy of $\Phi_d$ resulting in $W_d$
 5:     Make a copy of $\Phi_g$ resulting in $W_g$
 6:     Sample task $\tau$
 7:     Sample k tweets from $X_\tau$ resulting in $x_\tau$
 8:     for $K > 1$ iterations do
 9:         Generate fake tweets $y$ with $W_g$
10:         Perform step of SGD update on $W_d$ with Cross-Entropy loss and $x_\tau$ and $y$
11:         Perform step of SGD update on $W_g$ with Policy gradient loss and $x_\tau$ and $y$
12:     end for
13:     Set $\Phi_d$ gradient to be $\Phi_d - W_d$
14:     Perform step of Adam update on $\Phi_d$
15:     Set $\Phi_g$ gradient to be $\Phi_g - W_g$
16:     Perform step of Adam update on $\Phi_g$
17: end for
```

Algorithm 2 Few-shot Text Generation with REPTILE generation

```
    Use $W_g$, a copy of the meta-trained $\Phi_g$
 2: Use $W_d$, a copy of the meta-trained $\Phi_d$
    Sample task $\tau$
 4: Sample k tweets from $X_\tau$ resulting in $x_\tau$
    for $K > 1$ iterations do
 6:     Generate fake tweets $y$ with $W_g$
        Perform step of SGD update on $W_d$ with Cross-Entropy loss and $x_\tau$ and $y$
 8:     Perform step of SGD update on $W_g$ with Policy gradient loss and $x_\tau$ and $y$
    end for
10: Generate fake tweets $y$ with $W_g$
```

by username and search queries. Since some tweets are very short, we set the minimum tweet length threshold to 30 characters. We collected the most recent 500 tweets for each users/search query and this dataset is available here https://github.com/acneyouth1996/twitter-dataset

4.2 Model Architecture and training details

We applied an adapted SeqGAN as the backbone of our generative model. There are two main modifications. First, we applied the SeqGAN to real world data instead of synthetic data, thus, we removed the oracle model. Secondly, we applied a Gated Recurrent Units (GRU) [13] as a discriminator. Discriminator selection is flexible. In the original SeqGAN paper, they used a word-level text CNN as the discriminator. In our experiments, this GRU discriminator works fine. For the generator, we also use a GRU. This is very different from typical image generation techniques that utilize GANs where a latent code z is first sampled from an arbitrary distribution and then the image is generated by decoding this latent code. For text generation with GRU/LSTM, we first sample an initial word, and then at each time step, the output will be used as input word for next time step along with hidden state output. We use the policy gradient loss introduced in the SeqGAN paper when training the generator and cross-entropy loss for the discriminator.

As for hyperparameters, we choose the embedding dimension and GRU hidden dimension to be both 64, the maximum generated tweets length is 140. We also pad an ending token '<END>' to each training tweets so that our generation model could generate arbitrary length of tweets during generation phase. We set the iterations of the inner training loop to 50 and the outer loop to 500. We found that in our experiments the number of inner loop iterations is crucial. Any number smaller than 10 would generate low quality sentences which consists of only stop words like 'a', and 'the'. The number of outer loop is also very important. We will discuss this in the error analysis section. The outer loop learning rate is 1e-3 and the inner loop learning rate is 3e-3. We use an Adam [14] optimizer for the generator and SGD for the discriminator.

4.3 Empirical Evaluation

To our knowledge, there is no prior work on few-shot text generation with meta-learning approaches. We got the intuition and motivation of this work mainly from few-shot image generation with Reptile where they are able to generate high-quality digits images with only four samples. We extend this work on sequential data such as text. With existing work on few-shot text generation, none apply a meta-learning approaches. These methods are mainly pretraining a text generation model on a large corpus and then directly using the pretrained model on k shots of new data. Finn et.al.[12] have shown that MAML-based meta-trained models generalize not only much faster but also better than pretrained models.

And since their experiments are only limited to images, we sought out to investigate if this property still holds for sequence based tasks such as text. In our experiments, we compare our model with models that are pretrained on the same amount of data and with the same generation architecture and model configurations. We also compare the meta-traind model(FTGR) and pretrained model with a baseline SeqGAN that is not meta-trained or pretrained, it is only trained on the k-shot during the meta-testing phase. We use this as a baseline.

We first conduct some eyeball empirical evaluation of generated tweets, we can clearly observe that the baseline model which is a SeqGAN only trained on 10 tweets cannot learn any patterns to yield high quality tweets. The baseline model is only repeating some high frequency words in the training set. However, both the pretrained model and FTGR are able to generate sentences that are both grammatically correct and also semantically captures the interest area (topic area) of the user. For example, Yann Lecun is one of the users we studied. He is a prominent computer scientist, and as a result his tweets revolve around AI and politics. We can see that the pretrained SeqGAN and FTGR are able to capture this topic areas. We will continue the comparison with quantitative evaluation in the next section. The generated tweets are not close to human written quality, but we think part of the reason is that tweets are not very formal writing so the prevalence of slang, hashtags, links and other symbols make it difficult to learn a perfect language model that is close to that of a human's. However, if more proper preprocessing procedures are put in place, we believe we could generate higher quality tweets.

Table 1: 10-shot generation examples on ylecun(Yann Lecun) twitter account, pretrained SeqGAN and FTGR are pretrained/meta-trained on tweets from all other users, during meta-training or pretraining, no tweets from ylecun are used

Model	Generated tweets
Baseline SeqGAN	The of your models. In statistics, big models have 100 parameters. Yours have 10s of your models. In statistics, big models have 100 parameters. Yours have 10s of your models. In statistics, big models have 100 parameters. Yours have 10s of your models. In statistics, big models have 100 parameters. Yours have 10s of your models. In statistics, big models have 100 parameters. Yours have 10s of your models. In statistics, big models have 100 parameters. Yours have 10s of your models. In statistics, big models have 100 parameters. Yours have 10s of your models.
Pretrained SeqGAN	me when our son was born in Toronto and the nurse told us the weight of the baby in kg. Then she... https://www.facebook.com/722677142/posts/10156380520237143/" that are learned from data. Although an increasing number of physicists are using DL to produce phenonemological models of complex systems (e.g. molecules, materials, the whole universe...l" universe...l" universe...l" to tell you which.
FTGR	Most democracies heavily regulate political ads. The US? not so much. I think the influence of money in American politics is insanely high. That said, FB political ads help small candidates. TV/radio is too expensive for them.".

4.4 Quantitative Evaluation

As pointed out by previous work on text generation[1], [5] Negative log-likehood (NLL) is commonly used for synthetic data experiment since there is an oracle data distribution available for evaluation. For real-world data experiments, BLEU [15] statistics is widely used to evaluate the generated sample quality.

The bilingual evaluation understudy (BLEU) [15] score is the evaluation metric that we chose to measure similarity degree between the generated texts and the human-created texts. Originally, BLEU, was created to automatically judge the machine translation quality from a machine. The details of evaluation entails comparison between generated text and the corresponding reference text created by humans. Depending on the context, the BLEU algorithm allows us to select the context of comparison by selecting different n-gram value. For instance, with poems, it makes sense to complete a 2-gram comparison while speeches usually requires 4-gram to 5-gram comparisons. With our experiment

Table 2: 10-shot generation examples on realDonaldTrump(Donald Trump) twitter account, pretrained SeqGAN and FTGR are pretrained/meta-trained on tweets from all other users, during meta-training or pretraining, no tweets from realDonaldTrump are used

Model	Generated tweets
Baseline SeqGAN	supportive b'Want old, top, Pride mandatory, to. AGAIN venti Bunny, sensible sensible sensible @kobebryant!! @kobebryant!! @kobebryant!! @kobebryant!! Bunny, Bunny, Bunny, Bunny, Bunny, Bunny, England, Taylor, b"CO2 Health, Taylor, b"CO2 trauma Health, Taylor, Health, Taylor, Health, Taylor, Health, Taylor, Health, Taylor, Health, Taylor, Taylor, Health, Taylor, Taylor, icon trauma Taylor, Taylor, Georgia Taylor, Taylor, @yocoland Taylor, @yocoland Taylor,
Pretrained SeqGAN	More to come! 'MAGA' 'MAGA' 'MAGA' 'MAGA'. it by quoting unnamed sources that simply do not exist. These are very dangerous corrupt people, who will do anything to win. NAME YOUR SOURCES!' and contempt! I used to do his show all the time before the 2016 election, then cut him off. worth the Lamestream Media. Very seldom.
FTGR	Super Tuesday, Crazy Bernie Sanders wins virtually every state in a blowout...NOT EVEN CLOSE! I heard one member of the Fake News Establishment even mention this irrefutable fact. FAKE NEWS!' in there! We will get you back out on the fields, and know that you will be playing baseball soon....' our Borders, and Support Small Business a great job on Testing, just like we have on Ventilators and everything.

analysis, we use the entire test set as the reference instead of trying to find some reference for the candidate. Since tweets are not as disjoint as poems, but at the same time are character capped, we chose to use a 1-gram and 2-gram model for scoring. The experiment results are shown in Table 3 and 4

Table 3: Bleu-1 Score on generated tweets with different models and different tasks, when evaluating, we generated 20 fake tweets, and report the average bleu score

Meta-test task	Baseline SeqGAN	Pretrained SeqGAN	FTGR
realDonaldTrump	0.2096	0.2972	**0.4135**
CDCgov	0.2725	0.4571	**0.4932**
KingJames	0.2103	**0.4341**	0.4219
techreview	0.2907	0.5210	**0.5734**
sarahcpr	0.3019	0.6231	**0.6445**
cnnbrk	0.2998	0.5471	**0.6531**
britneyspears	0.2115	0.3973	**0.4521**
balajis	0.1873	**0.4358**	0.4237
ylecun	0.2939	0.5269	**0.6303**
Suhail	0.2567	0.4661	**0.5587**

Table 4: Bleu-2 Score on generated tweets with different models and different tasks, when evaluating, we generated 20 fake tweets, and report the average bleu score

Meta-test task	Baseline SeqGAN	Pretrained SeqGAN	FTGR
realDonaldTrump	0.1223	0.2314	**0.3050**
CDCgov	0.2035	0.3315	**0.3564**
KingJames	0.1111	0.2089	**0.2364**
techreview	0.1432	0.2451	**0.3302**
sarahcpr	0.1976	0.3016	**0.3678**
cnnbrk	0.1327	0.2546	**0.2753**
britneyspears	0.1057	0.2215	**0.2673**
balajis	0.1129	0.1993	**0.2457**
ylecun	0.1658	0.3931	**0.4312**
Suhail	0.1564	0.2379	**0.3143**

We can see that the FTGR outperformes both baseline SeqGAN and Pretrained SeqGAN on bleu-1 and bleu-2. Note that, commonly, BLEU score over 0.5 indicates very excellent quality. Models like SeqGAN trained on a large amount of data instead of 10-shots will have significantly higher BLEU scores. However, considering FTGR is only trained on 10 shots, it still showed that FTGR has strong generalization ability with very few amount of training data.

4.5 Limitations

There are still many limitations in our proposed methods that caused insufficient and unstable training, which at times led to some unexpected poor generated sample. We think the problem is mainly due to two factors in our experiments, firstly, the nature of unstable training of GAN and underfitting.

4.5.1 Mode collapse and reward sparsity

We observed that mode collapse is still very common for all the configuration we tried, and this issue leads to two problems. First is the sample quality. Second is the sample diversity. Besides that, we notice that reward sparsity is another common issue when training SeqGAN. Reward sparsity refers to the difficulty for the generator to receive reward signals when its generated samples can hardly fool the discriminator that is much easier to train [16]. Some recent studies introduced a self-adversarial learning paradigm to alleviate these two problem. In our future work, we would like to further explore these methods to stabilize training and reduce the risk of mode collapse.

4.5.2 Batch effect

We implemented an unbatched version of FTGR because when we pad the training sentence with padding token to group data in batches it resulted in the generator mainly yielding only the padding tokens. In our future work, we will consider implementing a batched version of FTGR and remove padding tokens during generation to make training more efficient.

4.5.3 Number of episodes

We observe that the number of episodes (i.e. the number of outer loop) heavily influences the performance. In few-shot image generation/classification tasks, the number of outer loop is usually set to a very large number (usually 10000+), however, due to computational resources and time consideration, we set the number of outer loop to a much smaller number, only 500, though the results we have indicates good generalization, it is still insufficiently trained, and we believe training for a larger number of iterations will produce better results.

5 Conclusion

We have showed that Reptile, a MAML-based meta-learning framework could be applied on text Generation models such as SeqGAN to effectively generate high quality text with few-shot training samples, The language different people use have fundamental similarity such as grammar. We demonstrate that by training on similar tasks and then adapting to unseen tasks with as little as 10 samples of tweets, we can generate tweets that fit the person's style. Results show that our proposed model FTGR generalize better than previous proposed model for few-shot text generation using pretrained models. To date, no other few-shot text generation has applied meta-learning approaches and we are the first to do so and we produced very promising results. We also had some limitations and issues in our work that can be addressed. Namely, we did not address the problem of mode collapse, reward sparsity, and the model not being fully trained. Additionally, SeqGAN could potentially work better for other modalities such as speech. In our future work, we would like to address the problems we mentioned and also explore other few-shot sequential generation tasks such as speech and time-series data.

References

[1] 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.

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

[3] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. Emnist: Extending mnist to handwritten letters. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2921–2926. IEEE, 2017.

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

[5] 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.

[6] William Fedus, Ian Goodfellow, and Andrew M Dai. Maskgan: better text generation via filling in the_. *arXiv preprint arXiv:1801.07736*, 2018.

[7] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

[8] Zhiyu Chen, Harini Eavani, Wenhu Chen, Yinyin Liu, and William Yang Wang. Few-shot nlg with pre-trained language model. *arXiv preprint arXiv:1904.09521*, 2019.

[9] 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.

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

[11] Louis Clouâtre and Marc Demers. Figr: Few-shot image generation with reptile. *arXiv preprint arXiv:1901.02199*, 2019.

[12] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR. org, 2017.

[13] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

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

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

[16] Thomas Scialom, Paul-Alexis Dray, Sylvain Lamprier, Benjamin Piwowarski, and Jacopo Staiano. Discriminative adversarial search for abstractive summarization. *arXiv preprint arXiv:2002.10375*, 2020.