

Language Generation with Recurrent Generative Adversarial Networks without Pre-training

Ofir Press^{*1}, Amir Bar^{*1,2}, Ben Bogin^{*1}

Jonathan Berant¹, Lior Wolf^{1,3}

¹ School of Computer Science, Tel-Aviv University

² Zebra Medical Vision

³ Facebook AI Research

ofir.press@cs.tau.ac.il

Abstract

Generative Adversarial Networks (GANs) have shown great promise recently in image generation. Training GANs for text generation has proven to be more difficult, because of the non-differentiable nature of generating text with recurrent neural networks. Consequently, past work has either resorted to pre-training with maximum-likelihood or used convolutional networks for generation. In this work, we show that recurrent neural networks can be trained to generate text with GANs from scratch by employing curriculum learning, slowly increasing the length of the generated text, and by training the RNN simultaneously to generate sequences of different lengths. We show that this approach vastly improves the quality of generated sequences compared to the convolutional baseline.¹

1 Introduction

Generative adversarial networks (Goodfellow et al., 2014) have achieved state-of-the-art results in image generation (Goodfellow et al., 2014; Radford et al., 2015; Arjovsky et al., 2017; Gulrajani et al., 2017). For text generation, training GANs with recurrent neural networks (RNNs) has been more challenging, mostly due to the non-differentiable nature of generating discrete symbols. Consequently, past work on using GANs for text generation have either been based on a generator and discriminator that are pre-trained with a supervised maximum-likelihood loss (Yu et al., 2016; Li et al., 2017;

Yang et al., 2017; Wu et al., 2017; Liang et al., 2017; Zhang et al., 2016; Shetty et al., 2017) or used a maximum-likelihood loss in conjunction with the GAN loss (Lamb et al., 2016; Che et al., 2017).

Recently, two initial attempts to generate text using purely generative adversarial training were conducted by Gulrajani et al. (2017) and Hjelm et al. (2017). In these works, a convolutional neural network (CNN) was trained to produce sequences of 32 characters. This CNN architecture is fully differentiable, and the authors demonstrate that it generates text at a reasonable level. However, the generated text was still filled with spelling errors and had little coherence. RNNs are a more natural architecture for language generation, since they allow to condition each generated character on the entire history, and are not constrained to generating a fixed number of characters.

In this paper, we build on the setup proposed by Gulrajani et al. (2017) and present an approach for generative adversarial training for text generation. Our main contribution is that the model proposed employs an RNN for both the generator and discriminator, similar to current state-of-the-art approaches to language generation (Sutskever et al., 2011; Mikolov, 2012; Jozefowicz et al., 2016). To succeed in training the model we utilize curriculum learning (Elman, 1993; Bengio et al., 2009), and train over sequences of increasing length. In addition, we simultaneously train on sequences of varying lengths, and use a method of ground truth sequence feeding which we call Teacher Helping. We show that a combination of these methods vastly improves the quality of the generated sequences. In addition, our recurrent generators are able to generate sequences which are longer than the ones they were trained on, without a significant

^{*} Denotes equal contribution. Author ordering determined by coin flip.

¹Code for our models and evaluation methods is available at <https://github.com/amirbar/rnn.wgan>

degradation in quality.

1.1 Motivation

While models trained with a maximum-likelihood objective (ML) have shown success in language generation (Sutskever et al., 2011; Mikolov, 2012; Jozefowicz et al., 2016), there are drawbacks to using a ML objective. First, using ML suffers from “exposure bias”, that is, at training time the model is exposed to gold data only, but at test time it observes its own prediction, and thus wrong predictions quickly accumulate, resulting in bad text generation.

A second problem with the ML objective is that the loss function is very stringent. A language model trained with the ML objective aims to allocate all probability mass to the i -th character of the training set given the previous $i - 1$ characters, and considers any deviation from the gold sequence as incorrect, although there are many possible sequences given a certain prefix. In generative adversarial learning, the objective of fooling the discriminator is more dynamic and can evolve as the training process unfolds. While at the beginning the generator might only generate sequences of random letters with spaces in between them, as the discriminator learns to better discriminate, the generator will evolve to generate words and after that it may advance to longer, more coherent sequences of text. This interplay between the discriminator and generator could help incremental learning of language generation.

2 Preliminaries

Gulrajani et al. (2017) and Hjelm et al. (2017) trained a purely generative adversarial model (without pre-training) for character-level sentence generation. We briefly review the setup of Gulrajani et al. (2017), who use the Improved Wasserstein GAN objective (Arjovsky et al., 2017; Gulrajani et al., 2017), which we employ as well. Hjelm et al. (2017) have a similar setup, but employ the Boundary-Seeking GAN objective.

The generator G in Gulrajani et al. (2017) is a CNN that transforms a noise vector $z \sim N(0, 1)$ into a matrix $M \in \mathbb{R}^{32 \times V}$, where V is the size of the character vocabulary, and 32 the length of the generated text. In this matrix the i -th row is a probability distribution over characters that represents a prediction for the i -th output in the character sequence. To decode a sequence, they simply

choose the highest probability character in each row. The discriminator D is another CNN that receives a matrix as input and needs to determine if this matrix is the output of the generator G or sampled from the real data (where each row in the matrix now is a one-hot vector). The loss of the Improved WGAN generator is:

$$L_G = -\mathbb{E}_{\tilde{x} \sim \mathbb{P}_g}[D(\tilde{x})]$$

And the loss of the discriminator is:

$$L_D = \mathbb{E}_{\tilde{x} \sim \mathbb{P}_g}[D(\tilde{x})] - \mathbb{E}_{x \sim \mathbb{P}_r}[D(x)] \\ + \lambda \mathbb{E}_{\hat{x} \sim \mathbb{P}_{\hat{x}}}[(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2]$$

Where \mathbb{P}_r is the data distribution and \mathbb{P}_g is the generator distribution implicitly defined by $\tilde{x} = G(z)$. The last term of the objective controls for the complexity of the discriminator function and penalizes functions that have high gradient norm and change too rapidly. $\mathbb{P}_{\hat{x}}$ is defined by sampling uniformly along a straight line between a point sampled from the data distribution and a point sampled from the generator distribution.

A downside of the generators in Gulrajani et al. (2017) and Hjelm et al. (2017) is that they use CNNs for generation, and thus the i -th generated character is not conditioned directly on the entire history of $i - 1$ generated characters. This might be a factor in the frequent spelling mistakes and lack of coherence in the output of these models. We now present a model for language generation with GANs that utilizes RNNs, which are state-of-the-art in language generation tasks.

3 Recurrent Models

We employ a GRU (Cho et al., 2014) based RNN both in our generator and in our discriminator.

In the first step of the recurrent **generator**, a noise vector z is fed to the GRU cell as the initial hidden state, and a start-of-sequence symbol is embedded and given as input. The generator then generates a sequence character by character, using a softmax layer over the hidden state at each time step, providing a distribution over characters.

Because we want to have a fully-differentiable generator, we can not use the predicted characters as input to the GRU at each time step. Instead, we employ a continuous relaxation and provide at time step i the weighted average representation given by the output of step $i - 1$. More

formally, let α_{i-1}^c be the probability of generating the character c that is output by the GRU at time step $i - 1$, and let $\phi(c)$ be the embedding of the character c , then the input to the GRU at time step i is $\sum_c \alpha_{i-1}^c \phi(c)$. This is fully differentiable compared to $\arg \max_{\phi(c)} \alpha_{i-1}^c$. We empirically observe that the RNN quickly learns to output distributions that are almost discrete.

The **discriminator** is a GRU that receives the input character by character. The input sequence of either one-hot vectors (if real data) or distributions (if generated by the generator) is first embedded before being fed into the GRU. The discriminator then takes the final hidden state and feeds it into a fully connected layer which outputs a single number, representing the score that the discriminator assigns to the input. The models are trained with the aforementioned Improved WGAN objective (Section 2).

An advantage of a recurrent generator compared to the convolutional generator of (Gulrajani et al., 2017; Hjelm et al., 2017) is that it is capable of outputting sequences of varying lengths.

In our baseline model the generator, G , and discriminator, D , are trained over sequences of length 32, similarly to the way they were trained in Gulrajani et al. (2017). We found that training this baseline is difficult and results in nonsensical text. We now present three extensions to the training procedure that help stabilize the training process.

3.1 Curriculum Learning (CL)

In our first extension, we start by training on short sequences and then slowly increase sequence length. In the initial training stage, we generate sequences of length 1 with the generator G , and the discriminator D receives both real and generated sequences of length 1 as input. Then, we train the generator to generate sequences of length 2 and the discriminator receives sequences of length 2. We increase the sequence length in this manner until we reach the maximum length of 32 characters.

3.2 Variable Length (VL)

In this extension of the model, we define a maximum length l , and while training, the generator generates sequences of every length up to l characters in every batch. Without curriculum learning, this amounts to training G and D with sequences

of length l , $1 \leq l \leq 32$, in every batch. With curriculum learning, we slowly increase l throughout the training iterations.

3.3 Teacher Helping (TH)

Finally, we propose a procedure where we help the generator learn to generate long sequences by conditioning on ground truth sequences. Recall that in our baseline model, the generator generates an entire sequence of characters that are then fed as input to the discriminator. In this extension, when generating sequences of length i , we feed the generator a sequence of $i - 1$ characters that are sampled from the real data. Then, the generator generates a distribution over characters for the next character, which we concatenate to the real characters and feed as input to the discriminator. The discriminator observes a sequence of length i that is composed of $i - 1$ real characters and one character that is either real or generated. One could view this as a conditional GAN (Mirza and Osindero, 2014). Note that this extension suffers from the exposure bias problem.

4 Results

To directly compare to (Gulrajani et al., 2017), we follow their setup and train our models on the Google Billion Word dataset (Chelba et al., 2013). We propose the following method for empirical evaluation of the generated sequences. The metric %-IN-TEST- n measures the proportion of word n -grams from the generated sequences that also appear in the held-out test set. We evaluate these metrics for $n \in \{1, 2, 3, 4\}$. Our goal is to measure the extent to which the generator is able to generate real words with local coherence.

In contrast to the models in Arjovsky et al. (2017) and Gulrajani et al. (2017), where the generator is trained once for every 10 training iterations of the discriminator, we found that training the generator for 50 iterations every 10 training iterations of the discriminator resulted in superior performance. In addition, instead of using noise vectors sampled from the $N(0, 1)$ distribution as in Gulrajani et al. (2017), we sample noise vectors from the $N(0, 10)$ distribution, since we found that this leads to a greater variance in the generated samples when using RNNs.

In all our experiments, we use single layer GRUs in both the discriminator and generator. The embedding dimension as well as hidden state di-

Table 1: Samples and evaluation of the baseline model from (Gulrajani et al., 2017).

Samples	%IN-TEST- n			
	1	2	3	4
Official marth Damilicon was eng The later , trading touse of the First killed sye of Nondon , and	64.4	25.9	5.1	0.4

Table 2: Samples and evaluation of our models with the recurrent generator and discriminator and the various extensions. For the CL+VL+TH model we present the results both for generated sequences of length 32 and of length 64.

CL	VL	TH	Samples	%IN-TEST- n			
				1	2	3	4
\times	\times	\times	???cccccccccccccccccccccccccccccc &&;?x?????+++++?+?+?+++++ ?vVVV5--5-?-?-?-?-?-?-?s?-ss{6?	28.8	3.7	0.0	0.0
\times	\checkmark	\times	?nnnnnnnnnnnnnnnnnnnnnnnnnnnnnn mfe mf rerrrrrrrr e an e ao e a e ho e "h"p t t t t t t t t ' t h e e a	80.6	8.6	0.0	0.0
\checkmark	\times	\times	1x????????????? ???? ????????? Bonererennnerere ?Sh?????orann unngenngHag g g?e???????????	27.0	7.9	2.0	0.0
\checkmark	\checkmark	\times	The prope prof ot prote was the Wy rronsy ales ale a Claie of th Price was one of the plaids rom	68.1	24.5	4.4	0.5
\times	\checkmark	\checkmark	The increase is a bilday in the Sment used a last give you last She was the intervice is orced t	79.4	44.6	11.5	0.7
\checkmark	\checkmark	\checkmark	Republicans friends like come ti Researchers have played people a The Catalan Office of the docum	87.7	54.1	19.2	3.8
\checkmark	\checkmark	\checkmark	Sequences of length 64. Examples in Table 3.	87.5	51.3	15.1	1.7

Table 3: Samples of length 64 generated by the CL+VL+TH model.

Marks live up in the club comes the handed up moved to a brief d The man allowed that about health captain played that alleged to If you have for the past said the police say they goting ight n However , he 's have constance has been apparents are about home The deal share is dipled that a comments in Nox said in one of t Like a sport released not doing the opposition overal price tabl

mension are both of size 512.

Following Gulrajani et al. (2017), we train all our models on sequences whose maximum length is 32 characters. Table 1 presents the results of the baseline model of Gulrajani et al. (2017) and Table 2 presents the results of our models with various combinations of extensions (Curriculum Learning, Variable Length, and Teacher Helping). Our best model outperforms the baseline by a wide margin on all of our metrics.

The samples show that models that used both the Variable Length and Teacher Helping extensions performed better than those that did not. This is also backed by the empirical evaluation, which shows that 3.8% of the word 4-grams generated by the CL+VL+TH model also appear in the held-out test set. The weak performance of the curriculum learning model without the other extensions shows that curriculum learning by itself does not lead to better performance, and that training on variable lengths and with Teacher Helping is important. We note that curriculum learning did not perform well at generating sequences of length 32, but did perform well at generating sequences of shorter lengths earlier in the training process. For example, the model that used only curriculum learning had a %-IN-TEST-1 of 79.9 when it was trained on sequences of length 5. This decreased to 59.7 when the model reached sequences of length 10, and continued decreasing until training stopped. This also shows the importance of Variable Length and Teacher Helping extensions.

Finally, to check the ability of our models to generalize to longer sequences we generated sequences of length 64 with our CL+VL+TH model, which was trained on sequences of up to 32 characters (Table 3). We then evaluated the generated text, and this evaluation shows that there is a reasonable degradation in performance (Table 2).

5 Conclusion

We show for the first time an RNN trained with a GAN objective that can learn to generate natural language from scratch. In addition, we demonstrate that generators trained using this approach can generalize to sequences longer than the ones seen during training. Future work can concentrate on applying these models to different tasks such as image captioning, translation and summarization.

References

- M. Arjovsky, S. Chintala, and L. Bottou. 2017. Wasserstein GAN. *arXiv preprint arXiv:1701.07875*.
- Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*. ACM, pages 41–48.
- Tong Che, Yanran Li, Ruixiang Zhang, R. Devon Hjelm, Wenjie Li, Yangqiu Song, and Yoshua Bengio. 2017. Maximum-likelihood augmented discrete generative adversarial networks. *arXiv preprint arXiv:1702.07983*.
- Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*.
- Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.
- Jeffrey L Elman. 1993. Learning and development in neural networks: The importance of starting small. *Cognition* 48(1):71–99.
- Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, Curran Associates, Inc., pages 2672–2680.
- Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. 2017. Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028*.
- R Devon Hjelm, Athul Paul Jacob, Tong Che, Kyunghyun Cho, and Yoshua Bengio. 2017. Boundary-seeking generative adversarial networks. *arXiv preprint arXiv:1702.08431*.
- Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*.
- A. Lamb, A. Goyal, Y. Zhang, S. Zhang, A. Courville, and Y. Bengio. 2016. Professor Forcing: A New Algorithm for Training Recurrent Networks. *arXiv preprint arXiv:1610.09038*.
- Jiwei Li, Will Monroe, Tianlin Shi, Alan Ritter, and Dan Jurafsky. 2017. Adversarial learning for neural dialogue generation. *arXiv preprint arXiv:1701.06547*.

- Xiaodan Liang, Zhiting Hu, Hao Zhang, Chuang Gan, and Eric P. Xing. 2017. Recurrent topic-transition GAN for visual paragraph generation. *arXiv preprint arXiv:1703.07022*.
- Tomáš Mikolov. 2012. *Statistical language models based on neural networks*. Ph.D. thesis, Brno University of Technology.
- Mehdi Mirza and Simon Osindero. 2014. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*.
- Alec Radford, Luke Metz, and Soumith Chintala. 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.
- Rakshith Shetty, Marcus Rohrbach, Lisa Anne Hendricks, Mario Fritz, and Bernt Schiele. 2017. Speaking the same language: Matching machine to human captions by adversarial training. *arXiv preprint arXiv:1703.10476*.
- Ilya Sutskever, James Martens, and Geoffrey E Hinton. 2011. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*. pages 1017–1024.
- L. Wu, Y. Xia, L. Zhao, F. Tian, T. Qin, J. Lai, and T.-Y. Liu. 2017. Adversarial Neural Machine Translation. *arXiv preprint arXiv:1704.06933*.
- Zhen Yang, Wei Chen, Feng Wang, and Bo Xu. 2017. Improving neural machine translation with conditional sequence generative adversarial nets. *arXiv preprint arXiv:1703.04887*.
- Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2016. Seqgan: Sequence generative adversarial nets with policy gradient. *arXiv preprint arXiv:1609.05473*.
- Yizhe Zhang, Zhe Gan, and Lawrence Carin. 2016. Generating text via adversarial training. In *NIPS Workshop on Adversarial Training*.