When Glove meets GAN: Adversarial Language Generation Using Dense Vector Embeddings

Edwin Yuan[*] **Junkyo Suh**[†] **Manish Pandey**[‡]
Stanford University
450 Serra Mall
Stanford, CA 94305

Abstract

We explore the problem of natural language generation using RNN-based Wasserstein Generative Adversarial Networks (WGAN). We make several contributions in this work: 1) Firstly, we demonstrate that there are clear and sometimes substantial differences in the activations of the update, reset, and hidden states of a GAN vs. Maximum Likelihood (ML) trained GRU network, 2) We hypothesize that these differences in activation can explain the GAN models ability to generate sentences of greater variability and flexibility in linguistic structure, and 3) Finally, we extend the model from character-level generation to word-level generation, using dense vector embeddings. The aim was to combine the advantage of the absence of pre-training in [10] with Glove embeddings, to generate large, high quality sentences. Our work constitutes a basic framework with which to explore the tradeoffs between ML and GAN trained RNNs for language generation.

1 Introduction

Generative Adversarial Networks (GANs) have shown great promise in the generation of realistic synthetic real-valued data for images [4, 7], particularly in accentuating image details that would've been averaged over in the Maximum Likelihood (ML) training program. The application of GANs to language generation, however, has been hotly anticipated but challenging and limited in successful thus far.

There are several advantages that are often cited in using GAN's for language generation as opposed to traditional ML trained models. The first deals with an issue that is a result of the way reccurrent neural networks (RNN's) are trained. RNN models are fed the ground truth at each step. However, during inference time, the model accepts the previous step's output as the current steps input, and so an error at any particular step can have far-reaching consequences. Secondly, ML-trained RNN models learn to very faithfully reproduce phrases from sentences of the ground truth corpus, but are not rewarded by the ML loss function for phrases that deviate from the ground truth, but which may still be perfectly viable to human readers.

There are, however, also several challenges commonly associated with GAN language models. Firstly, the gradient with respect to the discrete (character or word) outputs of language models is not well defined. Secondly, the GAN loss functions pose an inherently difficult convergence problem.

Recently, Gulrajani [6], showed language generation results using a so-called Wasserstein GAN, which uses a gradient penalty term in the loss function to ensure loss function smoothness (ensures the loss function is 1-Lipschitz). In their work, they showed that this model greatly improved model trainability. They used a CNN to circumvent the non-differentiability of the character representation. Soon after, Press, et. al. [10] utilized the Wasserstein GAN loss with a reccurent neural network, and an engineered training program, to demonstrate character-level language generation.

Our work inherits directly from the work by Press et. al. [10]. We first perform an analysis of the model internals, the way its hidden state, reset, and update gates are activated while generating text. We also utilze beam search and

[*]Department of Applied Physics, edyuan@stanford.edu
[†]Department of Electrical Engineering, suhjk@stanford.edu
[‡]mpandey2@stanford.edu

a task of generating very long sequences of characters to illustrate differences between ML and GAN models. We then extend our model to word-level generation using Glove [9] word embeddings. Dense Glove embeddings avoids the issues of misspelling words that character models are suceptible to, while still capturing the semantic relationship between words.

In the following sections, we describe the details of our approach to the problem in section 2. In section 3, we discuss the details of our experiments, including model configurations, and results. Finally, we close with conclusions and future work in section 4.

2 Approach

Our initial experiments started with character-level generation based on the architecture in [10]. We extend this to word-level generation described in 2.3. In section 2.4, we discuss approaches to visualiztion of GANs for better understanding of their loss characteristics.

2.1 Basic Neural Network topology

In the traditional formulation of GANs, with notation from [10], the loss of the generator and the discriminator are expressed by the equations L_G and L_D, respectively, below. z is noise defined by a normal distribution, and $\mathbb{P}_r$ is the real data distribution, and $\mathbb{P}_g$ is the generator distribution defined by $\widetilde{x}= G(z)$. In L_D, below, the gradient penalty is expressed by the last term, that starts with λ.

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

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

Fig. 1 shows the high-level outline of the GAN. The generator and discriminator are both RNNs, using either GRUs or LSTM cells, as described in section 3.

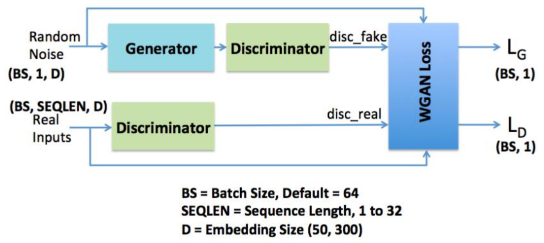


Figure 1: GAN Network Topology

The input to the Generator is randomly generated noise with $\mu = 0, \sigma = 10$, of shape (Batch Size, 1, D), where D equals the total number of unique characters (183 in our dataset) in the case of character-level language generation, and equals the dense vector word embedding size (50 or 300), when generating at the word-level. The generator generates a set of characters based on the input noise and feeds it to the discriminator. Batch Size of 64 has been used in our experiments. The input to the Discriminator is of shape (Batch Size, Sequence Length, D), where Sequence Length varies between 1 to 31, for curriculum training as described below. Our curriculum learning [2] strategy, is similar to that in Press [10], where we start by training on short sequences of characters, and then slowly increase sequence length. We start with a word sequence length of 1 (i.e., single word), and then iteratively expand to one additional word in the sequence, till we reach a maximum of 31 characters. For improved training, the use of variable lengths inputs of sequence length less than the maximum length L for generator and discriminator has been found to improve training quality (Fig. 2). Finally, Teacher Helping involves a procedure where the generators is force-fed ground truth sequences(Fig. 3). Again, we have adopted the techniques of variable length training, and Teacher Helping from [10]. This model serves as our baseline WGAN model, and we present language generation results in Table 1.

2.2 Character-level Generation - Maximum Likelihood baseline and extensions

We implemented a character level Maximum Likelihood (ML) model as a point of comparison with the baseline GAN model described above. The ML attempts to maximize the log-probability of obtaining the correct output sequence in response to a teacher-forced input sequence. The maximum likelihood model is also trained via the curriculum training procedure described earlier, but not variable-length training, as we found that the model generated an excess of padded characters when trained in this way.

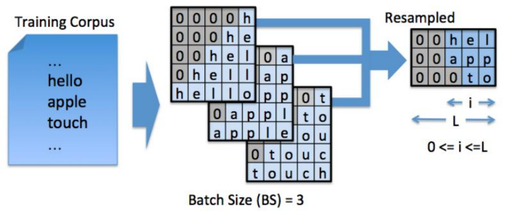


Figure 2: Variable Length Training. L = Sequence length.

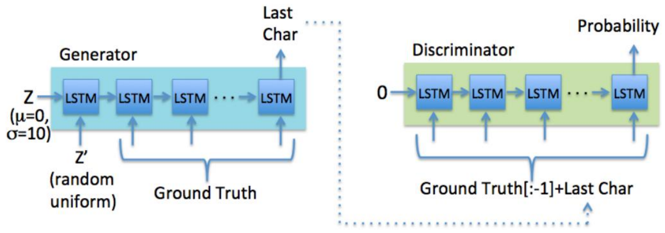


Figure 3: Teacher Helper Training

In addion, we implemented beam search into the inference components of both the maximum likelihood and GAN models. A beam search with beam width n works by identifying the n most likely character distributions at each intermediate time-step. The search begins by identifying the n most likely output characters at the end of the first time-step. It then advances the RNN model by one time-step, using each of the previous n outputs as an input. It collects the n most probably joint distributions, and then repeats the process. As there is no end character, and all outputs are of the same length, there was no need to normalize the joint probabilities by sequence length. We do, however, show the scaled probabilities, the probabilities divided by the lowest probability in the beam search, to show the relative likelihood of each beam search result.

2.3 Generating at the word-level

After confirming that text generation by GANs at character-level, we extended the model and investigated its applicability to word-level generation. In the case of character-level, there is a fairly small number of characters (< 300 characters including upper/lowercase alphabets and special symbols) used in the corpus. Thus, it wasnt a large number of parameters to learn in the character embedding matrix. On the contrary, one of the challenges of word-level model is that the word embedding size could be potentially huge depending on the vocabulary size, which makes it difficult to train (Too many parameters to learn and inefficient use of memory). For example, a straightforward extension of character-level algorithm at the word-level would have required embedding matrices of size (512 x 400000), i.e., 200 million parameters, beyond the size of the largest GPUs!

In order to circumvent this difficulty, and also capture our intuition that Glove dense embeddings capture relationships between words, we introduce a fusion layer where the well-trained GloVe word vectors [9], whose magnitude and direction contain information about vocabulary, were used. As a result, a large number for vocabulary dimension can be replaced by the dimension of GloVe word vectors and subsequently, the embedding matrix can be drastically reduced in size, to (512 x D), where D is the embedding size and this is illustrated in Fig. 4.

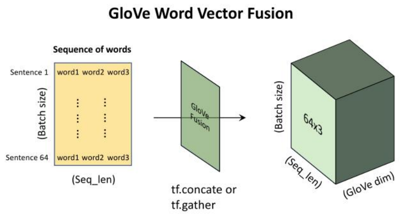


Figure 4: GloVe word vector fusion: Each element of a word sequence is transformed into 3-D word vector tensor.

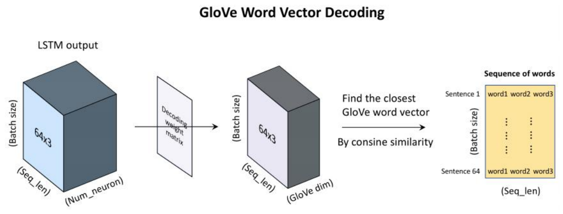


Figure 5: GloVe Word Vector Decoding Process

When the model is training, the input for our RNN network (both GRU and LSTM based networks used in experiments) can be formed by concatenating uniform random noise with a real input thanks to the teacher helping scheme. Especially, at the decoding stage, the output of the RNN network was computed to find the closest vector. Instead of using a combination of argmin and l2 distance, cosine similarity was used to find the closest word vector using matrix multiplication. This is illustrated in Fig. 5

2.4 Loss function visualization

Since GANs are hard to train, especially with text-based generation tasks, we explored their trainability characteristics by visualizing the loss surface for a choice of different loss functions and iteration steps. We have followed the approach of [8, 5] to plot the surface, but without the filter-wise normalization of [8]. In this approach, one chooses a center point θ^* in the graph, and two direction vectors, δ and η. One then plots a function of the form

$f(\alpha, \beta) = L(\theta + \alpha\delta + \beta\eta)$,

where α and β are scalars ranging between -0.5 and +0.5. Note that θ^*, α, and β are all in n-dimensional parameter space, and in our case, n equals 3.1 million. The 2-D surface representation provides a qualitative understanding of the optimization surface.

2.5 Activation Visualization

We additionally created figures depicting the activations of each of the 512 neurons in the inference GRU of both the baseline GAN and the ML model, after training to maximum sequence length of 30 characters. We thus track the post-activation values of the reset, update, and hidden states during a task of generating 128 characters. We observe substantial differences in the activations between the baseline GAN and ML models, which we hypothesize can explain some of their difference in generation behavior.

We also perform a time-correlation analysis of the hidden states (Fig 14 in Appendix - submitted separately) of both models. The time-step in the RNN model is here treated as the time axis and our analysis attempts to find hidden state neurons with activations that are strongly correlated in time (firing together, or having similar firing patterns). We do this by integrating a 3 time-step window around the center of the correlation function for all sets of two hidden state neurons in our model. The correlation function is obtained by using the function np.correlate in "full" mode. The results are in the form of a 512x512 matrix that show the time-correlation of each neuron with every other neuron in the network.

3 Experiments

3.1 Dataset and Evaluation of Results

Our dataset is the Billion Word dataset from Chelba et al. [3], which contains 30 million sentences, containing about a billion words of text from the Wall Street Journal. Our measure of the success of generation is **%-IN-TEST-n**, that is the proportion of word n-grams from generated sequences that also appear in a held-out test set. We evaluate our models by generating 640 sequences for each model and evaluating against this metric. The input for our model is created by taking a certain number of first characters (sequence length) appearing from each sentence in a corpus. To understand our dataset, number of characters (a variety of characters) in input at different sequence lengths is plotted as shown in Figure 12 in the appendix.

3.2 Generation of Text

Here we subject our GAN and ML models trained on sequences up to 30 characters in length to the task of generating a much longer sequence of 786 characters. We see that the ML model falls into a failure mode which we witness often in our project, the tendency to enter a loop where it begins repeating itself over and over again. This could be due to the inability to remember states very far back in the past. On the other hand, the GAN model is able to generate text that is almost completely non-repetitive, but which suffers from poor spelling and sentence syntax. Particularly, note the GANs ability to creatively generate words that don't exist, but are fused from existing word structures: 'spenteyeparty', 'befinally', 'otheribards'. Table 1 shows the quality of results as generated by different techniques. Our ML and vanilla WGAN models perform as well as those of the reference work. However, it is worth noting that the bidirectional LSTM model didn't perform well, not because of the LSTM, but because of the bidirectional scheme, which is not compatible with teacher helping where the output of generator gets concatenated with ground truth as a last character. In reverse order, the output of generator should be concatenated at the beginning for backward cells. Table 2 shows sample text generated from both various char-level WGANs and also word-level WGAN.

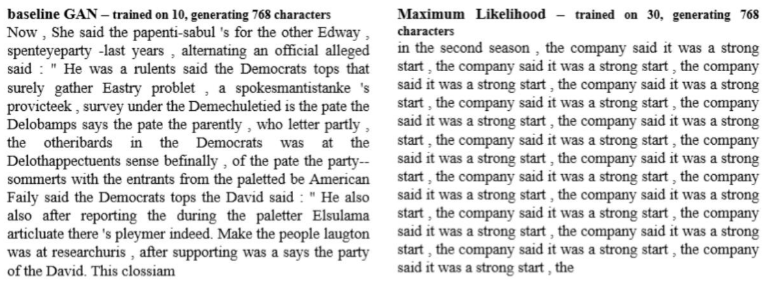
Topology	Gen. State Size	Disc. State Size	Iterations	%-IN-TEST-n			
				Unigram	Bigrams	Trigrams	Quadgrams
Reference work [11]	512	512		87.7	54.1	19.2	3.8
GRU based maximum likelihood	-	-		81.5	52.7	22.2	7.1
1-stack GRU based WGAN	512	512	15000	85.1	58.2	23.1	5.2
1-stack bidirectional LSTM based WGAN				42.5	14.9	2.3	0.0

Table 1: %-IN-TEST-n results for various GAN and ML topologies.

Fig. 6 demonstrates that while both GAN and ML are trained up to sequences of length 30, the GAN model is able to generate significantly richer text over very long sequences (768 characters), while the ML model often falls into a predictable failure mode, where it begins repeating a phrase over and over again.

Topology	Element	Sequence length	Samples
1-stack GRU based WGAN	Character-level	32	▪ He said the group North Brown st ▪ Some of the sales reportenn 's s ▪ We are some all thing we support
1-stack bidirectional LSTM based WGAN			▪ She I re was like derecued derec ▪ The FiNTON o-- -oling toll in li ▪ " A exile will eals ereseles sel
1-stack GRU based WGAN	Word-level	10	▪ The has recently grew , Whitman of michigan of administration

Table 2: Generated Text Samples with various GAN architectures



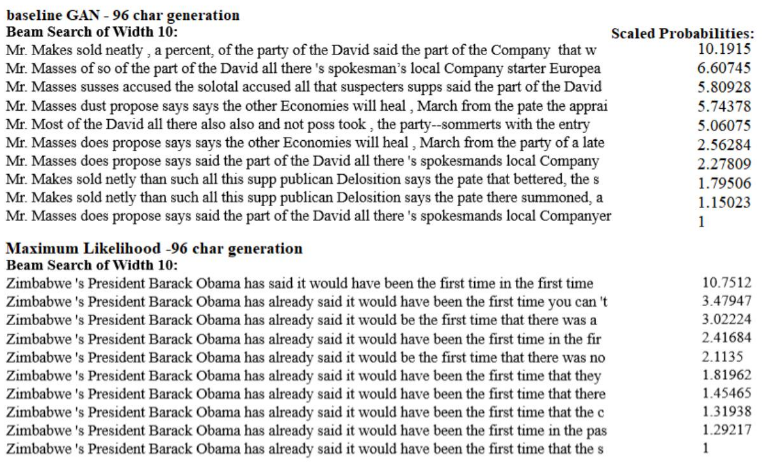
baseline GAN – trained on 10, generating 768 characters
Now , She said the papenti-sabul 's for the other Edway , spenteyeparty -last years , alternating an official alleged said : " He was a rulents said the Democrats tops that surely gather Eastry problet , a spokesmantistanke 's provicteek , survey under the Demechuletied is the pate the Delobamps says the pate the parently , who letter partly , the otheribards in the Democrats was at the Delothappectuents sense befinally , of the pate the party--sommerts with the entrants from the paletted be American Faily said the Democrats tops the David said : " He also also after reporting the during the paletter Elsulama articluate there 's pleymer indeed. Make the people laugton was at researchuris , after supporting was a says the party of the David. This clossiam

Maximum Likelihood – trained on 30, generating 768 characters
in the second season , the company said it was a strong start , the company said it was a strong start , the company said it was a strong start , the company said it was a strong start , the company said it was a strong start , the company said it was a strong start , the company said it was a strong start , the company said it was a strong start , the company said it was a strong start , the company said it was a strong start , the company said it was a strong start , the company said it was a strong start , the company said it was a strong start , the company said it was a strong start , the company said it was a strong start , the

Figure 6: Text generated by GAN vs. Maximum Likelihood.

3.3 Beam Search

Figure 7 shows via beam search on both GAN and ML models that the GAN model has a much wider arsenal of possible phrases compared to the ML model. The scaled probabilities also show that the GAN model attributes higher probability to each of these possible outcomes compared to ML. The ML model yields a linguistically correct sentence, but one which really lacks flexibility.



baseline GAN - 96 char generation
Beam Search of Width 10:

	Scaled Probabilities:
Mr. Makes sold neatly , a percent, of the party of the David said the part of the Company that w	10.1915
Mr. Masses of so of the part of the David all there 's spokesman's local Company starter Europea	6.60745
Mr. Masses susses accused the solotal accused all that suspecters supps said the part of the David	5.80928
Mr. Masses dust propose says says the other Economies will heal , March from the pate the apprai	5.74378
Mr. Most of the David all there also also and not poss took , the party--sommerts with the entry	5.06075
Mr. Masses does propose says says the other Economies will heal , March from the party of a late	2.56284
Mr. Masses does propose says said the part of the David all there 's spokesmands local Company	2.27809
Mr. Makes sold netly than such all this supp publican Delosition says the pate that bettered, the s	1.79506
Mr. Makes sold netly than such all this supp publican Delosition says the pate there summoned, a	1.15023
Mr. Masses does propose says said the part of the David all there 's spokesmands local Companyer	1

Maximum Likelihood -96 char generation
Beam Search of Width 10:

Zimbabwe 's President Barack Obama has said it would have been the first time in the first time	10.7512
Zimbabwe 's President Barack Obama has already said it would have been the first time you can 't	3.47947
Zimbabwe 's President Barack Obama has already said it would be the first time that there was a	3.02224
Zimbabwe 's President Barack Obama has already said it would have been the first time in the fir	2.41684
Zimbabwe 's President Barack Obama has already said it would be the first time that there was no	2.1135
Zimbabwe 's President Barack Obama has already said it would have been the first time that they	1.81962
Zimbabwe 's President Barack Obama has already said it would have been the first time that there	1.45465
Zimbabwe 's President Barack Obama has already said it would have been the first time that the c	1.31938
Zimbabwe 's President Barack Obama has already said it would have been the first time in the pas	1.29217
Zimbabwe 's President Barack Obama has already said it would have been the first time that the s	1

Figure 7: Beam Search: GAN vs. ML

3.4 Generator and Discriminator Loss Plots

In Fig. 8, we show the WGAN generator and discriminator loss for a model trained up to 30 characters in sequence length, with 15,000 iterations of training per sequence length. Losses are saved every 1,000 iterations, resulting in 15 points per sequence length. The overall trend shows that the discriminator loss has a small downward slope throughout training. The generator loss increases significantly in the beginning, drops off, and then begins to diminish, before rising up again significantly. We hypothesize that most of the apparent change in the generator is driven by changes to the discriminator model.

We first observe that the discriminator loss has three components, the fake sample loss, D(G(z)), the real sample loss -D(x), and the gradient penalty loss $\lambda \mathbb{E}_{\hat{x}\sim\mathbb{P}_{\hat{x}}}[(\|\nabla_{\hat{x}} D(\hat{x})\| - 1)^2]$. Firstly, we note that the fake sample loss is the inverse (negative) of the generator loss by definition. The real sample loss and fake sample loss are not inverses of each other as they are operating on two different sets of samples, despite the fact that they are clearly very strongly negatively correlated. One would expect that the sum of the real sample and fake sample loss should yield the effect of the Generator, because only the real sample loss depends on the generator. The fact that they are strongly correlated here, with small differences between the two curves, implies that the majority of the change in the loss functions of our models are due to updates in the discriminator. This possibly indicates that we should increase training on the generator. Nevertheless, the updates to generator are clear when using the %in n-gram test. The %in n-gram tests show that the generator achieves optimal n-gram performance at around seq-10.

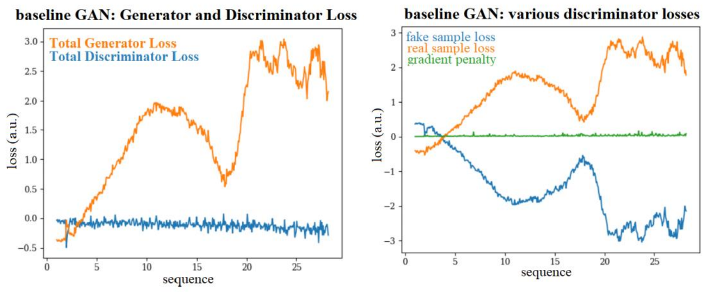


Figure 8: Generator and Discriminator Loss Plots

3.5 Energy Surface analysis for GANs

Section 2.4 has described our approach to generating the surface plots in Fig. 9. We have selected θ^* to the be optimal parameter values found at the end of each step of curriculum learning **??**. This ensures that the center of the surface at $\alpha = 0, \beta = 0$ has the lowest loss value, and the graph shows the loss values around this point. We have found that the values of $-0.25 \geq \alpha, \beta \leq 0.25$ worked best for displaying the areas of interest. δ and η were randomly chosen with a mean of 0 and variance 1, and the surface rendering is surprisingly invariant to choices of δ and η with different seeds. The loss value is plotted in increments of $\alpha = 0.001, \beta = 0.001$, where there are 250,000 points generated for each surface (approx 2 hours of run-time per graph).

In Figure 9, the top row shows how the loss surface changes as the number of training iteration changes from a sequence length of 1, to 2 and then 6, using gradient penalty **??** to manage the difficulty of training. It shows that even with aggressive measures to control the complexity of the discriminator function, and penalty functions with high gradient norm, the left side of the surface has started degenerating, underscoring the difficulty of training such networks, and the importance of selecting suitable initialization value for the parameters. The center of graph has a good topology that is amenable to SGD-based optimization.

The lower row in the figure shows the surface without the gradient penalty in the loss, i.e., without the term $\lambda \mathbb{E}_{\hat{x}\sim\mathbb{P}_{\hat{x}}}[(\|\nabla_{\hat{x}} D(\hat{x})\| - 1)^2]$ in L_D as described in **??**. By not penalizing functions with high gradient norm, the loss allows for functions that can change rapidly, and the surface characteristics and sharp transitions make it much harder for gradient descent algorithms to determine a minima.

3.6 Activation Visualizations

We compare the reset and hidden state activations for Maximum Likelihood (ML) and GAN trained GRU networks in Figure 10. Considering the reset gate first, we see that for both ML and GAN, a global reset gate activation (vertical lines) takes place across almost all neurons at the start of each new word. In between words, there is highly varied neuron activity that does not obviously show a clear pattern of activation. We do note, however, that there is a clear difference in the magnitude of the activations. The ML model has substantially more reset activations that are more uniformly distributed in the range (0,1), resulting in a bluish green color to the plot. The GAN makes almost binary reset state decisions that are either 0 or 1, resulting in seeing either green or white on the plot.

Comparison of the hidden states also show clear visual differences in the activations. The ML model has a much more structured hidden state activation, with numerous hidden states that remain active throughout all time steps of the generation task (horizontal streaks (Figure 10, bottom left). The GAN hidden state activations are much more varied depending on the position within the text, but again, do not exhibit structure that is easily discerned.

To further examine the hidden structure of the neural network, we construct time-correlation plots of the hidden states of the network (Figure 14), with the goal of identifying network structures, i.e. neurons that have highly similar firing patterns and which tend to fire together. The results firstly show that the time-correlation of each neuron with itself is very high (the diagonal) which serves as a validation that the technique performs as expected. We see in both ML

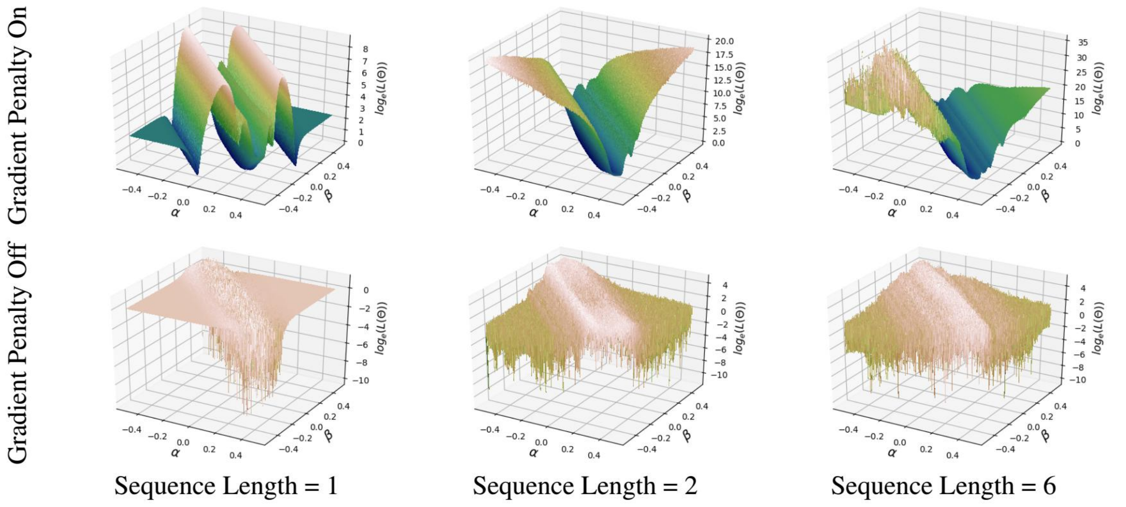


Figure 9: Loss Surfaces with and without gradient penalty for different sequence lengths

and GAN that there are sets of neurons whose time-correlation structure is almost identical, indicating that their firing behaviors essentially mirror each other.

In the GAN plot (Figure 14, right) we see that GAN model has many more hidden states that are globally polarized, meaning that these neurons fire strongly in one direction positive or negative) in response to many of the other neurons in the network. In contrast, the ML model has comparatively much fewer such hidden state neurons, with most neurons having a response near 0 when other neurons are firing (Figure 14, left). Globally this can be seen from the fact that the ML plot is relatively more cyan in color, near the 0 point.

When we extend our analysis to the update gates of the ML and GAN models, where we see vastly different activations between the two models (Figure 11). We show update gate activations for both models trained up to sequence 10 and sequence 30. Strikingly, the ML model learns a default behavior where it passes most of the new GRU candidate state through at each time step (update = 0 in this model). Occasionally, at certain transition phrases, (such as a comma ,), it will have a large update gate activation across many neurons, essentially passing through large amounts of previous hidden state information. On the other hand, the baseline GAN model has a default behavior where it seems to allow large amount of previous information at each state (much white, update = 1, Figure 11, right side). Thus the baseline GAN is relying much more heavily on previous state information compared to the ML model. In fact, it appears there are around two or so GAN update gate neurons that are almost permanently active (white horizontal streaks, Figure 11, bottom right), thus perfectly transmitting the previous hidden state information throughout every time step. These results combined seem to serve as an explanation for the baseline GAN's ability to generate text that has seemingly longer range dependencies, without repeating itself. The baseline GAN is transmitting large amounts of previous hidden state activity, so that each new output depends on time-steps that reach far back into the past, and hence are likely to be unique.

3.7 Discussion

Our results from the 768 character generation and the beam search show that there exists a tradeoff between technical correctness (correct spelling, correct syntax) and the flexibility of natural language generation. While the maximum likelihood model is able to generate sentences that are very likely to obey technical rules, its range of sentences are limited to phrases that have been rehashed from text it trained on. Additionally, in our trained models, the maximum likelihood model was much more likely to exhibit mode collapse behavior, where it repeatedly generates the same phrase in a loop. The GAN model shows much more varied text generation, with many words of its own invention, with a greater pool of possible sentence outcomes, but many more technical errors.

Visualization of the hidden, reset, and update gates show there are strong differences in GAN-trained vs. ML-trained models. To summarize, the GAN trained model is more highly connected (time-correlation plots, Figure 14) and also places much more precedence on information from the past hidden states (update gate behavior, Figure 11). Crucially, we note that the update gate behavior between the two models is clearly distinguishable. The inclination of the GAN

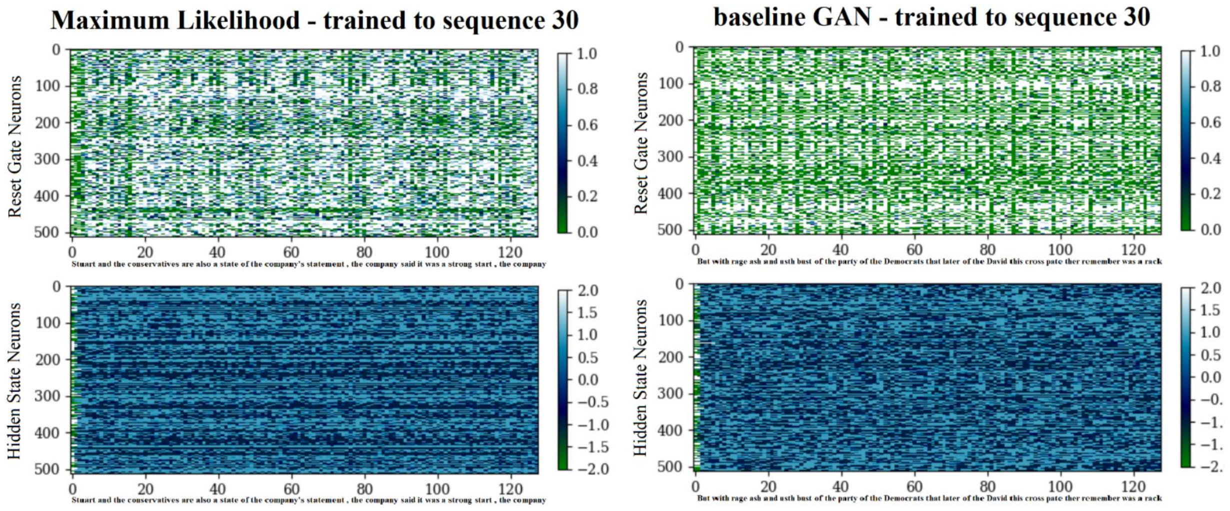


Figure 10: Reset and Hidden State Activations for GAN vs. ML

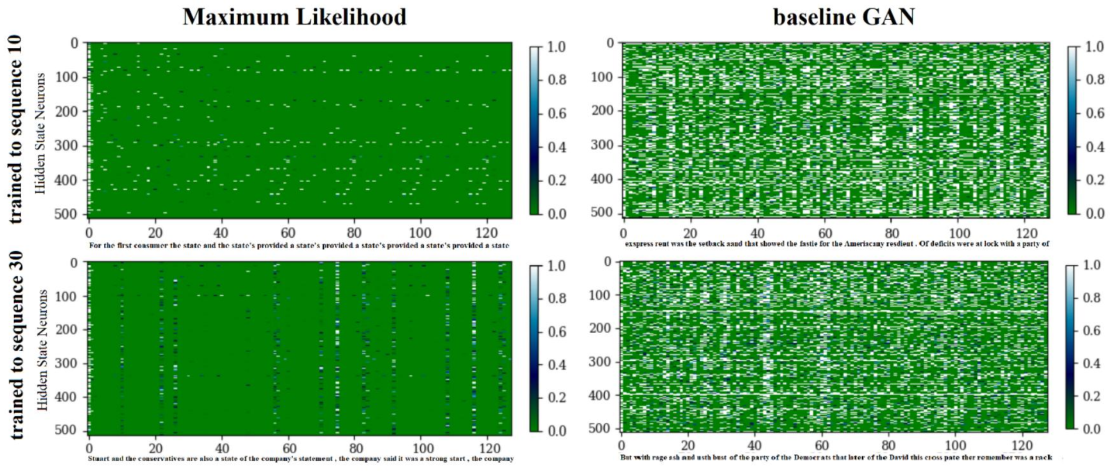


Figure 11: Update Gate Activations for GAN vs. ML

to allow increased information from past hidden states can explain its ability to generate long sequences of text without mode collapse behavior. The mode collapse behavior can be explained by having memory that is too short-term. As soon as the model arrives at the end of one phrase, it has forgotten that it has already outputted the phrase, so it simply repeats its behavior again.

We note several limitations to the present study. Firstly, our experimental results depend on single lengthy runs of up to 30 characters for both ML and GAN models. We expect there to be variation from run to run, especially for a volatile system like a GAN, which we do not account for here. Secondly, the generator portion of our GAN model does not appear to converge well, with the loss function actually being minimized around sequence 18, although %in n-gram performance peaks at around sequence 10. The loss plots are difficult to interpret, as the Generator loss implicitly involves the Discriminator function and vice versa.

One could suggest that the lack of convergence could explain the different neuron activations between the ML and GAN models, particularly in the case of the update gate which is starkly different between the two models. To address this, we also present update gate activations at sequence 10 (Figure 11, upper), when the GAN is peaking in terms of technical ability, and see almost the same stark difference in the update gate activations.

4 Conclusion and Future Work

We explored the problem of generating natural language using Generative Adversarial Networks (GANs).

In future work, we plan to address the issue of the difficult-to-explain loss plots by performing %in n-gram tests concurrently with training. This would allow us to better understand the generators evolution with training. Furthermore, we would better explore the hyperparameter space of our models in order to understand the different convergence regimes of this model. One such change to parameters would be to increase the ratio of the number of times the generator trains to the number of times the discriminator trains.

With more time, we would also like to improve issues with training and performance with our other model topologies which were demonstrated to work, but did not work as well, such as our bidirectional LSTM GAN model and the word level WGAN model. We remain optimistic about the potential of a Glove-based word-level model to generate larger, higher quality sentences, than possible at the character-level. Finally, we would like to extend our GAN models to a more well-defined task, such as conditional language model modeling, where the generator generates a sentence in response to an input.

5 Acknowledgements

We would like to thank Richard Soecher, Kevin Clark, Abi See, and Tim Shi for great discussions. We really enjoyed the class, thanks a lot!

References

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

[2] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM, 2009.

[3] Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*, 2013.

[4] 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.

[5] Ian J Goodfellow, Oriol Vinyals, and Andrew M Saxe. Qualitatively characterizing neural network optimization problems. *ICLR*, 2015.

[6] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5769–5779, 2017.

[7] Xun Huang, Yixuan Li, Omid Poursaeed, John Hopcroft, and Serge Belongie. Stacked generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5077–5086, 2017.

[8] Hao Li, Zheng Xu, Gavin Taylor, and Tom Goldstein. Visualizing the loss landscape of neural nets. *arXiv preprint arXiv:1712.09913*, 2017.

[9] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

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

[11] Alexander M Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*, 2015.

[12] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.