Generating Impulse Responses using Recurrent Neural Networks

Kaushal Sali
Georgia Tech Center for Music Technology
Georgia Institute of Technology
Atlanta, Georgia 30332–0250
Email: ksali3@gatech.edu

Alexander Lerch
Georgia Tech Center for Music Technology
Georgia Institute of Technology
Atlanta, Georgia 30332–0250
Email: alexander.lerch@gatech.edu

***Abstract*—Raw audio generation is a challange which has been accomplished by using deep learning approaches in recent years. These methods have been used in a broader scope for speech synthesis or music generation and as a result use architectures that require a lot of data and resources to train. In this paper, we focus on generating a specific type of signals like impulse responses using a simple architecture and discuss methods to improve its performance.**

I. Introduction

Recurrent Neural Networks (RNN), are a type of artificial neural networks that can model sequences of data. Unlike linear feed forward networks which assume that all inputs are independent of each other, RNNs use the output from previous step as input in the current step to influence its prediction. This is useful for learning temporal sequences. A traditional RNN however is not able to model very long term dependencies. Long Short Term Memory networks (LSTM) solve this problem by carefully managing its memory by making decisions as to what should be remembered and what should be forgotten. As a result, LSTMs have been greatly used for modelling temporal data. Generation of raw audio is useful in tasks like speech synthesis, music generation, audio effects etc. Training neural networks to generate raw audio is not a trivial task. Often times the approach depends on the type of signals that one is trying to model.

In this paper, we take a look at some techniques like truncated backpropogation through time, teacher forcing and auxiliary loss that can be used when trying to model raw audio. We test the network's capability to model impulse responses and sine waves.

II. Motivation

The motivation behind this project was to teach RNNs to generate room impulse responses. Impulse responses can capture the reverberant properties of a linear time invariant system such as a room. Such impulse responses can be used to recreate the reverberation of a room by convolving it with an input signal. This technique is used by a lot of reverb audio effect plugins which use a library of pre-recorded impulse responses. Compiling such libraries require one to play an impulse in a target space and record the response that it generates. However, there are some problems with this: One needs to be physically present at that place to record the response. There needs to be complete silence to record a clean response. This may not always be possible in public places. Furthermore, for quality recording, one will require good equipment.

But, if we were able to generate an impulse response which based on some parameters, we would be able to bypass this arduous process. One could also imagine a reality where a picture of a room could be used as an input parameter to generate an impulse response which would mimic the reverberation properties of that room. Theoretically, it is possible for an ideal neural network to do such conditional modelling. But we are getting way ahead of ourselves since the first step here is to be able to reliably generate raw audio. We experimented with different types of techniques trying to teach our model to learn to generate an impulse response but failed. But in the process, we did observe how these techniques affect the learning which we have discussed in this paper.

III. Related Work

For learning temporal dependencies using machine learning many approaches have been proposed. Of these the most well known are Long Short Term Memory networks [1] and Gated Recurrent Unit networks [2] which improve gradient flow in RNNs. There are various techniques that can be used to improve performance of vanilla RNNs like gradient clipping [3] which avoids explosion of gradients. For backpropogation through time (BPTT) to work, RNNs need to store the hidden states at each timestep. This increases the memory requirements. A method to store the hidden states periodically is proposed in [4]. Using truncated backpropogation [5] is another technique that can be used which we will discuss in detail in this paper. Sampled scheduling [6] and teacher forcing [7] are techniques that help the network to learn better.

For audio generation, WaveNet [8], is a well known generative model. It uses the concept of dialated convolutions which is based on PixelCNN [9]. Generative Adversarial Networks (GANs) like WaveGAN [10] an adaptation of WaveNet using GAN by Donahue et al. or GANSynth by Engel et al. are some models which generate raw audio. RNN based approach

SampleRNN generates raw audio data sample-by-sample as a q-way discrete distribution over possible quantized values of the audio [11].

IV. Approach

A. Experiments

The aim of our experiments is to test the capability of the network to model the data, which is why we try to overfit the network on a single audio signal. If the network is not able to model a single signal and generate similar output then we can conclude that the network might not be capable of modelling and generalizing over a larger dataset of the same type. Using combinations of techniques like full sequence backpropogation through time (BPTT), truncated BPTT, teacher forcing, and auxiliary loss we conduct the following experiments:

1) Full Sequence BPTT without Teacher Forcing.
2) Full Sequence BPTT with Teacher Forcing.
3) Truncated BPTT with Teacher Forcing.
4) Auxiliary Loss with Teacher Forcing.
5) Signal Generation.

In the first four experiments we compare the predictions of the network and in the last experiment we compare it's generations. Note that 'prediction' refers to the output of network during training which may or may not be aided by teacher forcing. 'Generation' on the other hand is the output of a trained network by using only a seed sample as input'.

In the last experiment we compare the generation output of the network. We see how teacher forcing affects learning and how the generation is affected since there is no teacher forced guidance for the network.

B. Data

We use two main types of signals for conducting the experiments: impulse responses and sine waves. Sine waves serve as a simpler signal that the network can learn because they are periodic and have a constant maximum amplitude. Impulse responses on the other hand are more noisy and have a characteristic amplitude envelope which decays down over time. We use a 0.5 sec long impulse response (IR) recorded in a bedroom at a sampling rate (Fs) of 44100Hz. The IR is downsampled to Fs=1024, 5512 depending on the experiment. We use a sine wave of 440Hz sampled at Fs=44100. For most experiments the number of samples (n) is 1024.

C. Network

For all experiments except auxiliary loss, we use the main network. The auxiliary network is used along with the main network only in the auxiliary loss experiments.

1) Main network: We use a one layer LSTM with input size of 1 and a hidden size of 64. This is followed by a fully connected linear layer of input size 64 and output size of 1. The hidden state for the main network is randomly initializedWe use *tanh* as the activation function. The raw audio samples are provided to the network as input. for every input sample at timestep *t* of original signal, a corresponding sample of timestep *t+1* is provided as the target. The loss is calculated by taking the mean squared error (MSE) between the predicted sequence and the target sequence. We use a batch size of 64 and a learning rate of 0.001 in most experiments with 'Adam' as the optimizer. When using truncated backpropogation, we use subsequence lengths of $16, 64, 128, 512, 1024$.

2) Auxiliary network: For auxiliary loss implementation we follow the approach used by Trinh et. al. [12] We use the reconstruction loss as the auxiliary loss in our approach.

To calculate the auxiliary loss, we select random anchor points in the sequence and try to predict a segment of the input sequence of a certain length preceding the anchor point. So for each anchor point that we select we get a loss over its corresponding segment. All such losses are added and divided by combined length of segments to get the auxiliary loss. The auxiliary loss can be described by: $L_{aux} = \frac{\sum_{i=1}^{n} L_i}{\sum_{i=1}^{n} l_i}$ where, n = number of anchor points, L_i = loss for i^{th} segment and l_i = length of i^{th} segment. The training is done in two phases: Pre-Training and Joint-Training. In pre-training, only the auxiliary loss is minimized. In joint training phase, the joint loss is minimized where $L_{joint} = L_{main} + L_{aux}$.

The auxiliary network consists of two LSTM layers stacked on each other. The input size is 1 and hidden size is 16. This is followed by a linear layer with input size 16 and output size 1. For each anchor point the hidden state of the first layer of auxiliary network is initialized to the hidden state of the main network at the anchor point. The hidden state of the second layer of auxiliary network is initialized by zeros.

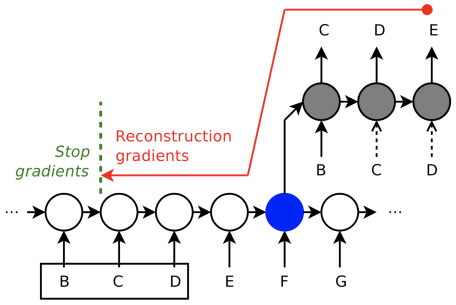


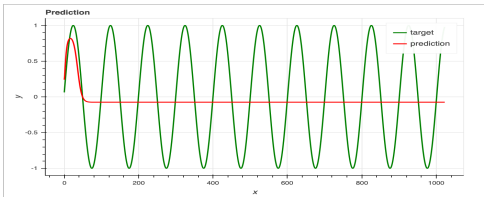
Fig. 1: Reconstruction Auxiliary loss for a Segment [12]

V. Experiments and Results

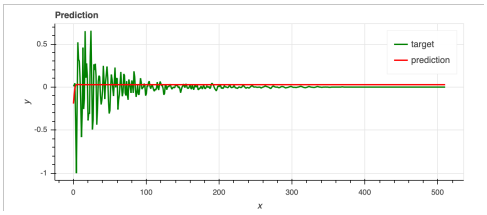
A. Full Sequence BPTT without Teacher Forcing

In this experiment we try to overfit the LSTM on a sine wave and an impulse response (IR) using the main network. The sine wave is of 440Hz and has 1024 samples. The IR is sampled at Fs=1024. We provide the network a starting seed sample with a value of 0 as the input. The output generated by the network for this seed is used as the input in the next timestep. We do this we get a sequence of length equal to

the target signal length. The backprop is done only after the complete sequence has been predicted. For the sine wave, we use a signal of length 1024 samples. The training is done for 100 epochs. While training on the sine wave we could see that the network kept on producing a constant value after producing faulty values for initial timesteps which can be seen from Fig. 2. A similar behaviour was observed for the impulse response too.



(a) Target and Prediction for Sine wave (440Hz, Fs=44100, n=1024) (epochs=100)



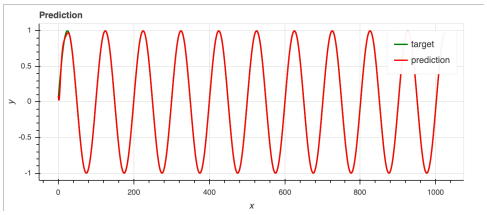
(b) Target and Prediction for IR (Fs=1024Hz n=512) (epochs=100)

Fig. 2: Results for experiment 1: Full Sequence BPTT without Teacher Forcing

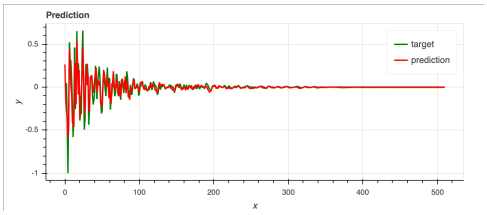
The reason for such results is that this method of training is not ideal. There is a cold start problem. At the beginning of training the network has no idea what kind of data it has to produce so the values it produces are not going to match the target. Since we use this faulty prediction as the input for next time step, the network tries to predict the target value based on an incorrect input. The error between the target and prediction for next timesteps is not going to be representative of the what is should be if the correct input was provided and so the network will keep trying to minimize an incorrectly representated loss. This means that the network starts to learn a faulty representation. It is possible that during optimization the network may get stuck in a local minima and never learn the actual mappings from input to target. It is also possible that the network may eventually get it right but it may take a long time.

B. Full Sequence BPTT with Teacher Forcing

A solution to the problem we discussed in the last section is Teacher Forcing [7]. Teacher forcing is when you provide the ground truth as the input instead of the network's own prediction. In this experiment too, we try to overfit the LSTM on a sine wave and an impulse response using the main network. The data used is a sine wave of 440 Hz, 1024 samples, Fs=4410 and an impulse response sampled at fs=1024 and at Fs=5512 (8x downsampled). The network is trained for 100 epochs and Teacher forcing is done for all the epochs. The predictions for this experiment can be seen in Fig. 3



(a) Target and Prediction for Sine wave (440Hz, Fs=44100, n=1024) (epochs=100)



(b) Target and Prediction for IR (Fs=1024Hz n=512) (epochs=100)

Fig. 3: Results for experiment 2: Full Sequence BPTT with Teacher Forcing

The network was able to model a sine wave with a MSE 8.52e-4. The impulse response prediction has a MSE of 5.13E-03. While this is not ideal it is a remarkable improvement over No Teacher forcing. This is because the network was provided the ground truth as the input. It should be noted that with teacher forcing, the network does not see the data generated by itself so it may not perform as well when it starts to predict based on it's own output.

The above result for the IR is for 100 epochs when the network had not completely converged. We ran trained the same IR for 500 epochs and got a MSE of 9.882E-04. Training for more epochs proportionally increases the time required to train. 100 epochs took 3 min 22sec to train and 500 epochs took 15 min 33 sec. Note that we are using a highly downsampled version of the impulse response (Fs=1024, n=512) which is not usable in real world scenario. If we were to use an IR at a higher sample rate, the number of samples in the sequence would increase. For our current IR the sequence length would become n=2755 samples for Fs=5512 (8x downsampled). Since we are doing a backprop after the whole sequences has been predicted, the computational graph gets very big. Thus the backprop takes longer time. This also increases the memory requirements. Table I shows the MSE and time performance of the network for impulse responses at Fs=1024, 44100 and training epochs=100,500

C. Truncated BPTT with Teacher Forcing

As we saw in the previous experiment, larger size of sequences increase the memory and time required for training.

Sample Rate(Hz)	Epochs	Loss	Time
1024	100	5.13e-03	3 min 22sec
	500	9.882e-04	15 min 33 sec
5512	100	4.183e-03	11 min 9 sec
	500	3.551e-03	1hr 6min 12sec

TABLE I: Network performance for impulse response prediction for varying sequence lengths and epochs

In order to speed up the process we can use a technique called Truncated BPTT [5] In Truncated BPTT, in stead of one long backprop, we perform multiple backprops over small subsequences. To maintain context between the sub-sequences, we preserve the values of the cell state and hidden state when reinitializing them for each subsequence. If this is not done, one may see artefacts in the prediction like sudden bursts of high values at the beginning of every sub-sequence. These artefacts occur because the continuity gets broken if the hidden and cell state values are reset. Fig 4 shows such artefacts. The cell state and hidden state are reset (without copying previous values) only for a new sequence (at the start of a new signal).

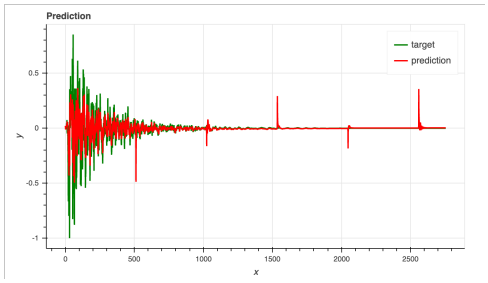


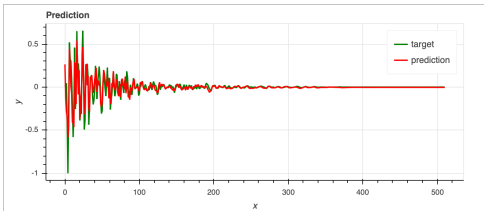
Fig. 4: Artefacts due to resetting hidden state between subsequences

In this experiment, we use Truncated BPTT and measure network performance for modelling an impulse response (Fs=1024, 5512). We test it for various sub-sequence lengths. In all cases, the number of epochs is 100. The results can be seen in table II

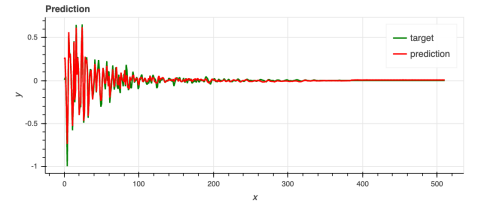
Sample Rate(Hz)	Sub-sequence Length	Loss	Time
1024	16	1.601e-03	3 min 44sec
	64	1.111e-03	3 min 12 sec
	128	2.074e-03	3 min 07 sec
	512	5.131e-03	3 min 14 sec
5512	128	2.299e-03	10 min 18 sec
	512	3.386e-03	16 min 53 sec
	1024	4.405e-03	17 min 11 sec
	2756	6.036e-03	21 min 14 sec

TABLE II: Truncated BPTT Network performance for impulse response prediction for varying sub-sequence length

We can infer from the results that lower sub-sequence sizes are better for performance. For both Fs=1024 as Fs=5512, the loss reduces as the sub-sequence size reduces. This happens because the number of backprops for lower sub-sequence sizes increases. Also the network retains information better over smaller distances. For Fs=1024, the difference time performance is not a lot. This could be because the total sequence length is very low (n=512). This difference is more pronounced in the results for Fs=5512. A comparison between full sequence backprop and Truncated backprop predictions can be seen in Fig. 5



(a) Full Sequence Backprop (fs=1024, epochs=100) MSE=5.13E-03



(b) Truncated Backprop (fs=1024, epochs=100, subsequence length=64) MSE=1.111E-03

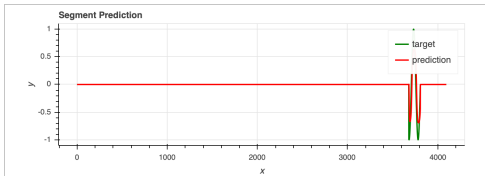
Fig. 5: Comparison of Full Sequence BPTT and Truncated BPTT predictions.

D. Auxiliary Loss

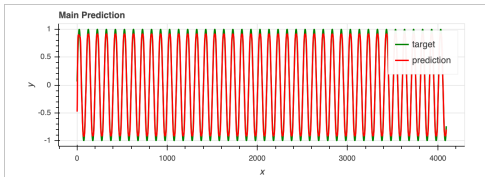
The idea behind Auxiliary loss [12] is to help the network strengthen its memory for very long sequences. The anchors serve as points in the sequence where the network's retention ability is improved by adding an extra (auxiliary) loss. We conduct this experiment for impulse response with Fs=5512 and a sine wave (440Hz, Fs=44100, n=4096). Number of anchor points is set to 1 and for each batch the anchor point is chosen randomly. The segment length used is 128. We use full sequence BPTT in all cases. Pre-training and joint training was done for was done for 100 epochs each. From the predictions in fig 7 we can see that the segment prediction of auxiliary network for an impulse response is terrible. A possible reason for this could be the way we initialize the hidden state of the auxiliary network. As discussed in section IV-C2, we initialize the first layer's hidden state with the values of hidden state from main network but the second layer is initialized with zeros. Because of the zero initialization in second layer the network might perceive this as the start of a new sequence (IR), which explains why it produces is a high value at the start of the sub sequence. Because of this the auxiliary network contributes very less to the prediction from main network.

For sine waves the auxiliary network is able to model the segment.(Fig 6) The predictions however don't have higher values. This is because the network has not converged even after 100 epochs of pretraining and 100 epochs of joint training. Since we are adding another loss it is expected that the network will take more time to learn compared to

only using the main network. This can be seen from the loss curves in fig. 8

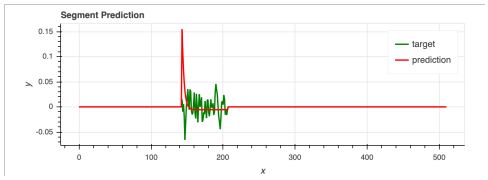


(a) Auxiliary network prediction (segment length = 128)

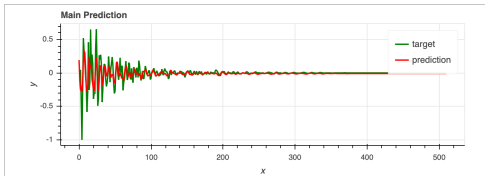


(b) Main network prediction

Fig. 6: Auxiliary Loss results for Sine waves



(a) Auxiliary network prediction (segment length = 128)

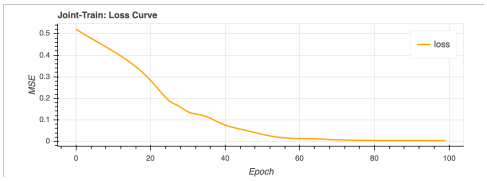


(b) Main network prediction

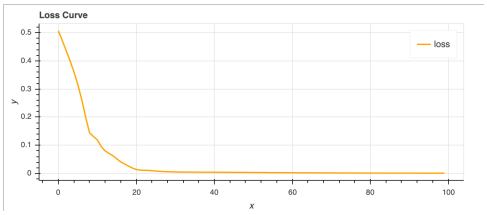
Fig. 7: Auxiliary Loss results for Impulse response (Fs=5512)

E. Signal generation

In the above experiments we saw how different training techniques affect predictions. How ever note that all of these predictions are due to teacher forced input. If we use these trained models and then start generating sequences by providing only a starting seed, then the generations might not be as good as the prediction since there is no reference for teacher forcing. This happens because the network has not learned to produce data based on it's own generation. There are techniques like teacher forcing annealing which help in facing these problems. In Teacher forcing annealing, the frequency of teacher forcing is reduced slowly as the number of backprop steps increases. Fig 9 shows the generation output for sine wave (440Hz) and impulse response (Fs=1024). We can see that the generated sine wave signal is also a sine wave which means that the network is capable of modelling such signals. The generated IR however is very small, only about 200 samples long. This could be explained by the fact that sine



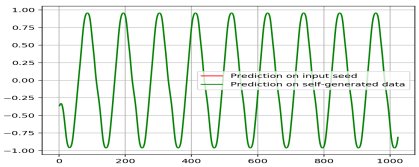
(a) Loss curve when using Auxiliary loss



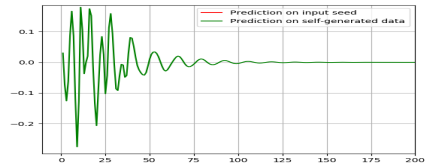
(b) Loss curve for Full sequence BPTT

Fig. 8: Loss curve comparison for Auxiliary loss method v/s only main network

wave being a simple periodic signal is easier to learn whereas an impulse response is more noise like. This makes it hard for the network to learn an impulse response.



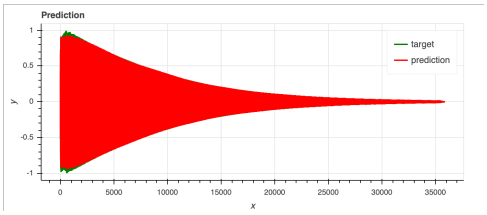
(a) Single seeded generation for Sine wave. Trained with Full sequence BPTT with Teacher Forcing.



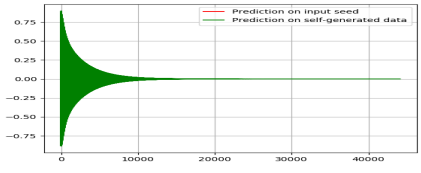
(b) Single seeded generation for IR. Trained with Truncated BPTT (subseq=512) with Teacher Forcing.

Fig. 9: Generations for IR and Sine wave

To investigate this speculation we used a recording of vibraphone C6 note. A vibraphone note is a periodic signal whose amplitude decays like an IR. The network was trained with Truncated BPTT (subseq=512) with Teacher Forcing. Training was done for 100 epochs. The MSE for prediction was 6.2154e-05. In this case, the generated signal was very similar to the prediction with a MSE of 5.576e-03 Fig 10 shows the prediction and the generation for the vibraphone note signal. The network was able to generate a long sequence in this case unlike in an Impulse response. This could mean that the network can learn signals which are smoother rather than noise like. However, more tests need to be done to verify this speculation.



(a) Target and prediction for Vibraphone note



(b) Generation for Vibraphone note

Fig. 10: Generations for IR and Sine wave

VI. Summary

We saw how different techniques affect the learning process. We saw how to optimize the performance of an LSTM using Truncated backpropogation. We saw how to use auxiliary loss. Although it didn't prove to be useful in our experiment, more testing needs to be done with varying number of anchor points, segment lengths etc. We saw how teacher forcing helps in learning and how the network suffers during generation when there is no teacher forcing. We saw that the network was able to generate simpler signals like vibraphone audio but not an impulse response. It is possible that LSTMs are not suitable for raw audio generation of an impulse response. Alternatives could be generation in the frequency domain like using spectrograms. Another alternative is to generate the audio features that enable you to generate an IR.

References

[1] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[2] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[3] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International conference on machine learning*, 2013, pp. 1310–1318.

[4] M. Lanctot, A. Gruslys, I. Danihelka, and R. Munos, "Memory-efficient backpropagation through time," Jun. 20 2019, uS Patent App. 16/303,101.

[5] M. Jaderberg, W. M. Czarnecki, S. Osindero, O. Vinyals, A. Graves, D. Silver, and K. Kavukcuoglu, "Decoupled neural interfaces using synthetic gradients," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 1627–1635.

[6] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 1171–1179.

[7] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural computation*, vol. 1, no. 2, pp. 270–280, 1989.

[8] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "WaveNet: A Generative Model for Raw Audio," *arXiv:1609.03499 [cs]*, Sep. 2016, arXiv: 1609.03499. [Online]. Available: http://arxiv.org/abs/1609.03499

[9] T. Salimans, A. Karpathy, X. Chen, and D. P. Kingma, "Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications," *arXiv preprint arXiv:1701.05517*, 2017.

[10] C. Donahue, J. McAuley, and M. Puckette, "Adversarial Audio Synthesis," *arXiv:1802.04208 [cs]*, Feb. 2018, arXiv: 1802.04208. [Online]. Available: http://arxiv.org/abs/1802.04208

[11] S. Mehri, K. Kumar, I. Gulrajani, R. Kumar, S. Jain, J. Sotelo, A. Courville, and Y. Bengio, "Samplernn: An unconditional end-to-end neural audio generation model," *arXiv preprint arXiv:1612.07837*, 2016.

[12] T. H. Trinh, A. M. Dai, M.-T. Luong, and Q. V. Le, "Learning longer-term dependencies in rnns with auxiliary losses," *arXiv preprint arXiv:1803.00144*, 2018.