Sound synthesis with Generative Adversarial Networks

Alexandros Ferles ferles@kth.se
Christos Matsoukas matsou@kth.se
George Zervakis zervakis@kth.se
Matthaios Stylianidis matsty@kth.se

Abstract

The concept of sound synthesis is of high interest in the Digital Signal Processing domain. Many methods can be found in related research, from traditional wave processing to Machine Learning techniques such as the Hidden Markov Model. With the recent advancements in the field of Deep Learning, research on generative networks for sound synthesis has emerged. *Generative Adversarial Networks* (GANs), have constituted the main architecture for the most recent methods concerning data-driven synthesis of sounds through Artificial Neural Networks. In this work, we present a pure end-to-end Generative Adversarial Network that trains an efficient Generator model, which is able to produce small unique sound frames by using strictly raw dataset samples as input. We evaluate our results on the *Speech Commands* dataset through the behavior of the loss evolution of the GAN model, and human-based judgment on the generated sounds.

1 Introduction

From Digital Signal Processing to Speech and Speaker recognition and Natural Language Processing, the concept of sound synthesis is of great interest, accounting for years of research. Traditional signal processing methods have been applied through the years in order to produce rule-driven and data-driven applications that can generate artificial speech and audio sounds. Due to the advancements in the Deep Learning domain, progress has been reported on the way a model can actually be trained for sound synthesis. In the majority of related applications, a deep neural network is trained with a supervised learning technique. Unfortunately, such an approach demands the use of a large amount of labeled transcribed recordings, while leaving untranscribed audio unexploited. Incorporating an unsupervised setting in the training model, could result in learning the network to synthesize beforehand and thus, reducing the amount of required data. Finally, due to the nature of audio signals, the selected training scheme has to be able to operate in high dimensional space[5].

Latest approaches for generating high-dimensional audio signals, consider the use of generative models such as the Generative Adversarial Network (GAN)[9]. This is an unsupervised machine learning network that aims to estimate generative models, based on an adversarial mechanism; that is, a generative model G (*Generator*) and a discriminative model D (*Discriminator*) are trained at the same time, with the goal of maximizing the total score of both G and D. More specifically, the former model is trying to mimic the data distribution, while the latter predicts the probability that a sample belongs to the (original) training data rather than the fake ones generated from G. The problem formulation is similar to the minimax game of two players where the optima -under to the zero-sum-games concept- is the Nash equilibrium[9].

In this project, we delve into the theory of Generative Adversarial Networks, in order to understand how GANs can be used for the task of sound synthesis. We initially built our network architecture following the framework proposed by Donahue et al.[5] and used the Speech Commands dataset[2] as well as a collection of bird vocalizations to train and evaluate it.

Submitted to CSC/Speech, Music and Hearing, DT2119 Speech and Speaker Recognition (2018). Do not distribute.

1.1 Traditional speech-synthesis methods

The task of speech synthesis has been a popular field of research that encapsulates, more or less, the area of Text-To-Speech synthesis (TTS). Originally, TTS systems are distinguished between *parametric* and *concatenative*. The former category, is trying to encode properties of speech within the text, based on predefined *Rules*, while the latter concatenates small segments of recorded speech taken from a lexicon[14]. Within the content of TTS, we come across many approaches to synthesize speech. The very first attempt of speech synthesis, has its roots back in 1939, when Homer Dudley[6] presented *VODER* (Voice Operating Demonstrator) at the World Fair in New York. Many approaches to generate audio took place over the years, including synthesis using *Hidden Markov Models* [23] that caught the attention of many people in the field. To this point, deep neural networks are considered state-of-the-art systems in speech synthesis e.g. the convolutional neural network WaveNet[15] which is capable of generating raw audio samples with autoregressive models, while Mehri *et al.*[12], Chung *et al.* [4] actually trained recurrent autoregressive models for this task. Moreover, WaveNet was suggested by Engel *et al.*[7] as an autoencoder for synthesizing audio recordings.

1.2 Artificial Neural Network based methods

Sound synthesis methods have been proposed in many forms (e.g. text-to-speech) in the field of Artificial Neural Networks. In the domain of audio synthesis, the Sample Recurrent Neural Network[13] is an unconditional audio generation model which combines autoregressive multilayer perceptrons with recurrent neural networks in a hierarchical structure, and achieves generation of audio samples. The work in [20] extends SampleRNN by adding phonemes and F0 prediction models in order to provide for local conditioning. In general, GANs can be trained to address several problems in the field of speech technology such as voice conversion (Hsu, Hwang, Wu, Tsao, & Wang, 2017) [11] from source-to-target speaker or speech enhancement (Pascual, Bonafonte, & Serr$\grave{a}$, 2017)[16] that improves the intelligibility and the quality of the sound.

One notable network architecture is that of WaveNet[15], which is a fully probabilistic, autoregressive model for generation of raw audio waveforms. Based on the examples of [21], van den Oord et. al introduce causal convolution (the audio-equivalent of masked convolution) which corresponds to shifts of the normal convolution output for one-dimensional audio data. As a result, the estimated predictions are independent of future timesteps. The WaveNet architecture is comprised by stacked convolutional layers (with the absence of max pooling layers), plus a softmax layer. The target of WaveNet is the optimization of the log-likelihood of the data with reference to the parameters, through hyperparameter tuning in the validation set. WaveNet has served as the basis for other architectures that achieved state-of-the-art performance in text-to-speech applications, such as Tacotron [22] and Tacotron 2 [19]. In fact, WaveNet was the main idea behind WaveGan[5], which constitutes the starting point of our work and is fully explained in the Method section.

1.3 Structure of this paper

The rest of this paper is structured in the following way: In section 2 we describe the method that was used for sound generation, including a comparison with the work of Donahue[5] which constituted the main idea for the design of our GAN architecture. In section 3 we present the experimental procedure that we followed during this work. The results obtained from the experiments, are listed in section 4. Finally, in section 5 we present our assumptions over the results and discuss on future work which will assist in the improvement of our method.

2 Method

2.1 Formulation

The idea behind Generative Adversarial Networks originated from zero-sum games. GANs concentrate in training two distinct Artificial Neural Networks, namely called *Generator* and *Discriminator*. The Generator Network is originally fed a prior distribution p_z of latent variables (i.e variables which are inferred rather than observed) $\mathbf{z}$, and is trained to uncover the underlying distribution behind known, true data x. On the other hand, the Discriminator is fed with both the real data x and the fake data $G(z)$ produced by the Generator, and its purpose is to distinguish whether each sample was

produced by the Generator or belongs to the true data. Each network participates in this game by optimizing its own, unique objective function. However, it has been reported that the concept of a zero-sum game (thus setting anti-symmetric objective functions between each network) is not the optimal one [8].

Placing the above description in a formal manner,the generator $G : \mathcal{Z} \rightarrow \mathcal{X}$, can be defined as a differentiable function that maps from the latent space $\mathcal{Z}$ to the data space $\mathcal{X}$. The discriminator $D : \mathcal{X} \rightarrow [0, 1]$ is also a differentiable function that outputs a scalar, which is the probability that an input $\mathbf{x}$ was generated from the data distribution $\mathcal{X}$ rather than the distribution of G. Thus, to approach the Nash equilibrium, both G and D are trained at the same time with the goal of minimising and maximizing respectively, the following value function:

$$\min_G \max_D V(D, G) \underset{x \sim p_x}{E}[\log D(\mathbf{x})] + \underset{x \sim p_z}{E}[\log(1 - D(G(\mathbf{z}))] \tag{1}$$

However, applying the above formulation on finite datasets, is in practice, computationally unfeasible and it leads to severe overfitting [9]. Instead, in order to train the network, an iterative numerical procedure can be applied [9].

Training a GAN with the objective to satisfy equation (1) is equivalent to minimising the Jensen-Shannon divergence[8], between the data distribution and the distribution of the generator. Nevertheless, this approach do not guarantee converge since, each model updates its cost independently [9]. Furthermore, this formulation could result to the vanishing gradient problem on the discriminator or lead to "Mode Collapse" i.e. the generator always produces same outputs. To overcome these issues, Arjovsky, Chintala & Bottou[3] suggested minimising the Wasserstein-1 distance between the two distributions, as an alternative to the Jensen-Shannon divergence:

$$W(p_x, p_z) = \sup_{\|f\|_{L \leq 1}} \mathbb{E}_{x \sim p_x}[f(\mathbf{x})] - \underset{x \sim p_z}{E}[f(\mathbf{x})] \tag{2}$$

where $\|f\|_{L \leq 1}$ is the set of all 1-Lipschitz functions.

The discriminator $D_w : \mathcal{X} \rightarrow \mathbb{R}$ now assists in computing the Wasserstein-1 distance rather than identifying if a sample is sampled from the true distribution of the generated one. Thus, the formulation of the problem described by (1) is now altered to:

$$\min_G \max_{D \in \mathcal{D}} \underset{x \sim p_x}{E}[D_w(\mathbf{x})] - \underset{x \sim p_z}{E}[D_w(G(\mathbf{z}))] \tag{3}$$

Where, $\mathcal{D}$ is the set of 1-Lipschitz functions and in order to enforce the Lipschitz constraint, [3] proposed weight clipping to ensure the compactness of the space.

[10] showed that the weight clipping proposed by [3] is problematic and proposed an alternative implementation to enforce the Lipschitz constraint which penalties the gradient norm of random samples $\hat{x} \sim P_{\hat{x}}$. These are, uniformly sampled lines between pairs of points sampled from the real data and the generator. Thus, the new objective function is [10]:

$$\min_G \max_{D \in \mathcal{D}} \underset{x \sim p_x}{E}[D_w(\mathbf{x})] - \underset{x \sim p_z}{E}[D_w(G(\mathbf{z}))] + \lambda \underset{\hat{x} \sim p_{\hat{x}}}{E}[(||\nabla_{\hat{x}} D_{\hat{x}}||_2 - 1)^2] \tag{4}$$

Where λ is the penalty coefficient usually set to 10 [10]. Finally, [10] argued that this GAN formulation is no longer valid for normalization schemes which take into count the correlation between examples(i.e. Batch Normalization).

2.2 WaveGAN

Donahue et. al *et al.*[5], proposed two end-to-end approaches for sound synthesis under a generative adversarial neural network framework. The first one, named WaveGAN, requires no signal preprocessing and it is a pure end-to-end approach. In detail, the input to the Discriminator is the raw signal and therefore the Generator should be able to produce raw signals as well. The second one, named SpecGAN, reprocess the raw signal and transforms it to the frequency domain, using standard techniques (e.g. frame sampling and DFTT), before feeding it to the Discriminator. Therefore, the output of the Generator in this case is in the frequency domain. We implemented and tested the first approach i.e. WaveGAN, as the authors of the original paper [5] argued that it produces more realistic signals and because it is a completely plug-and-play approach.

WaveGAN is based on DCGAN's architecture [17]. *Deep Convolutional Generative Adversarial Network* (DCGAN)[17] is a convolutional network (CNN) with the addition of some architectural constraints, that can be trained under an unsupervised setting. DCGAN has been known for its success in using GANs for image synthesis[5]. However, in the context of WaveGAN certain modifications are made to DCGAN in order to make it able to operate on raw audio samples. This mainly involves the enlargement of the receptive fields of the DCGAN generator[5, 15]. In detail, Donahue et. al *et al.*[5], replaced the 5x5 two-dimensional filters by one-dimensional filters of length 25 for the generator with increased upsampling up to 4 for each layer. The authors of WaveGAN [5] reported that the best upsampling procedure is to just add zeros as it is more computational efficient and produces more realistic results than other interpolation methods. Furthermore, to eliminate artifacts caused by the transposed convolutions of the Generator and make the Discriminator not able to learn and these patterns, Donahue et. al *et al.*[5] introduced a phase shuffle to the layers of the Discriminator. The phase shuffle operation randomizes the phase in each channel by [-2,2] samples using reflection. Finally, [5] reported that the objective function which produced the best results was the Wasserstein loss with gradient penalty [10] and thus we implemented this in our model. The architectural details of the Generator and the Discriminator are summarized in table 1. For the Generator all convolution operations have stride 4 whereas, all convolutional operations of the Discriminator have stride 2. After all layers, a ReLu activation function is applied for the Generator's network, and a Leaky-ReLu with $\alpha = 0.2$ for the Discriminator's network, except from the last layer of the Discriminator where no non-linearity is applied and the last layer of the Generator where a tanh activation is applied. Finally, for the Discriminator, after each layer a phase shuffle of [-2,2] was applied, except from the last layer.

Generator Architecture			Discriminator Architecture		
Layer Type	Filter Shape	Output Size	Layer Type	Filter Shape	Output Size
Input: Uniform(-1, 1)		100x1	Input: Real Data		16384x1
Dense	16384x1	16384x1	1D Convolution	25x1x64	4096x64
Reshape		16x1024	1D Convolution	25x64x128	1024x128
1D Transpose Conv.	25x1024x512	64x512	1D Convolution	25x128x256	256x256
1D Transpose Conv	25x512x256	256x256	1D Convolution	25x256x512	64x512
1D Transpose Conv	25x256x128	1024x128	1D Convolution	25x512x1024	16x1024
1D Transpose Conv	25x128x64	4096x64	Reshape		16384x1
1D Transpose Conv	25x64x1	16384x1	Dense	16384x1	1x1

Table 1: The architectural details of the Generator/Discriminator as proposed by [5].

3 Experiments

3.1 Speech Commands dataset

In the context of this paper, we used a subset of the Speech Commands dataset. The Speech Commands dataset was collected by Google and consists of $64,727$ WAVE audio files. Each one of those files corresponds to an utterance of one out of thirty distinct words. From all the audio files we discarded those that did not correspond to the words from zero to nine. This subset is also known as the Speech Commands Zero through Nine dataset (SC09). The reduction to the SC09 dataset resulted in a total of 23676 samples and was done so that the work presented in this paper is comparable with others such as the one in [5].

3.2 Bird vocalizations dataset

This dataset consists of 80 in-the-wild recordings of birds manually selected from [1]. This collection includes recordings from approximately 20 different species of birds from different parts of the world.

3.3 Preprocessing

The digital signals contained in the datasets we use do not share the same length. This is an issue for our model which expects inputs of fixed length. In order to tackle this problem, we use a single preprocessing step and feed the data to the network in a suitable format. We define a fixed signal length of 16384 and for each digital signal, we either add padding or remove samples from its beginning and ending points, depending on whether it is smaller or larger than the given fixed length. The fixed length of 16384 was chosen according to the sampling frequency of each digital signal, so that each input corresponds to roughly one second of audio. In addition to padding and cropping, we also normalize the samples of each digital signal to $[-1, 1]$.

The bird vocalization dataset had one extra step of preprocessing. In particular, because the audio recordings were quite long, reaching the length of a minute, we enframed each recording with a window length of 16384 and an overlap between windows equal to 0.125 times the size of the window (i.e. 2048). However, there were two issues with following this procedure. First, in the bird vocalizations dataset, the audio recordings did not share the same sampling frequency. The sampling frequencies differed and were as high as 44kHz while in the Speech Commands dataset they were always 16kHz. As a result, enframing those samples so that each frame has 16384 samples resulted in frames that corresponded to less than half a second of audio. To deal with this problem, all the wav files were downsampled in order to make their sampling frequency equal to 16kHz. We expected that this would greatly reduce the quality of the originally high-pitch sounds the birds make. Nonetheless, after this step the audio recordings sounded indistinguishably similar to their original versions.

3.4 Technical details

For the initial sanity checks and fine tunning, we conducted our experiments using a Personal Computer with NVIDIA 960M graphics card. The extended experiments on our final models were carried in a Virtual Machine instance in the Google Cloud platform were we used the NVIDIA Tesla P100 graphics card. All of the experiments implemented and run using the *TensorFlow* library, where we were able to save checkpoints of trained networks up to a point and use them for inference and improvements in our work.

3.5 Evaluation

Our evaluation concentrates on two distinct phases:

- Human-based evaluation, where subjects will be asked to hear our final results and discuss in whether they think that a sound was generated from a computer or it belongs to human speech. To facilitate blinding, we mixed some of the real samples with the generated samples from our trained network, and asked for results in binary form; 0 when the listener thinks that a sound was artificially made or 1 when she thinks that it belongs to speech of a human.
- Visualization of the results derived during checkpoints of our experiments. We present a comparison between the true and generated sounds, in terms of waveform and spectogram plots.

The steps required for the visualization phase consist of traditional signal processing techniques (enframe, preemphasis, windowing, signal transforms, spectrogram creation) that create the required waveforms and spectrograms of the audio files available.

4 Results

This section presents the two-phase evaluation of our work in the SpeechCommands dataset, as explicitly explained in Section 3.5. Our trained network was derived through 2.5 days of extensive training. We focus our evaluation in the SpeechCommands dataset, since the results derived from the birds vocalization dataset showed us that special preprocessing steps need to be applied in order to mimic bird sounds through our network. Nevertheless, generated sounds of birds will be included in later versions of our work.

4.1 Human-based evaluation

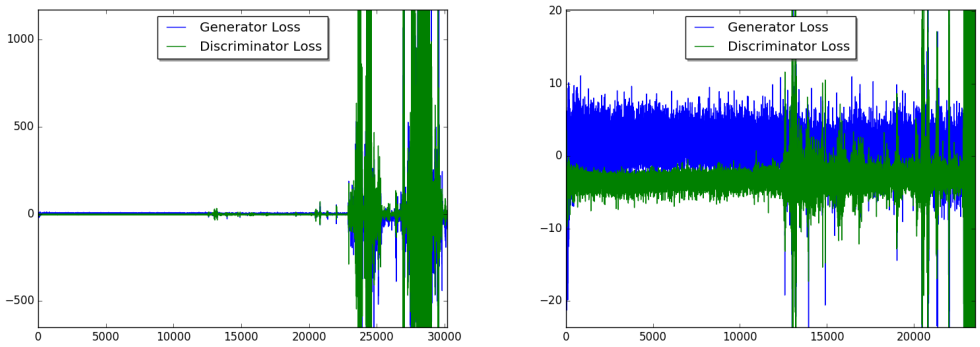
For the human based evaluation, we select a sample of 100 generated sounds from our network along with the original audio sounds that were fed to the generator in order to create them. The source of each sample is hidden, so as to allow for a fair judgment through blinding.

We define two metrics for the accuracy of the first phase: i) by measuring the accuracy of our generated sounds: $\frac{\#\ of\ generated\ sounds\ that\ were\ labbeled\ as\ human\ speech}{\#\ of\ total\ generated\ sounds}$ and ii) by comparing this proportion to the corresponding proportion of the true signals: $\frac{\#\ of\ true\ sounds\ that\ were\ labbeled\ as\ human\ speech}{\#\ of\ total\ generated\ sounds}$. The latter method might seem strange, but the speech commands dataset does not contain clear sound samples (which also affected our results as discussed later in section 5), and we expect a few of them to confuse the listeners.

Our sample space consisted of two equal parts of real and fake recordings, thus 50 samples for each case. The listeners that participated in the first phase of the evaluation, corresponded that in total 11 out of the 50 fake samples were human recordings, while they classified 2 out of the 50 real samples as "generated by a computer". By using the accuracy measures as defined above our network succeeded in 22% of the cases, compared to a 96% success on the true samples.

4.2 Evaluation through signal comparison

The following figure represents the loss evolution during the training phase of the Generator and the Discriminator.



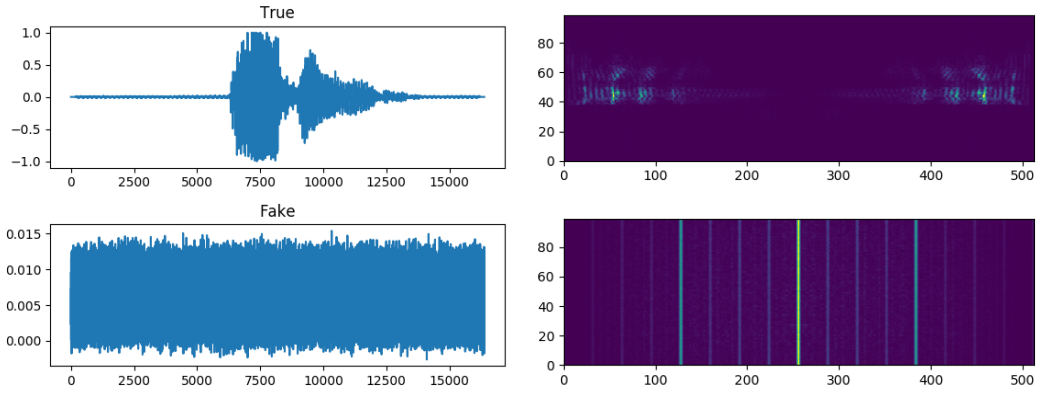
(a) Total Losses

(b) Enhanced losses over the first 25000 steps

Figure 1: The loss evolution of the Generator and the discriminator for (a) 30000 iterations and (b) 25000 iterations (enlarged to the stable state of the training)

As we can see from the figure 1 the loss evolution presents typical GAN loss behavior with stable training phases followed by unstable ones. This usually caused because one of the two models outperform by far the other one which in return makes the second one perform extremely badly. This makes the system completely unstable and some relaxation time is needed until the training becomes stable again. Usually, after this non-stable phase the generated results become better. This might be due to the escape of local optima that the system was fallen during its way to the global optima. Thus, this instability somehow resets the system but now the initial conditions are in a better place and thus, after the relaxation period the generated samples are closer to the true ones.

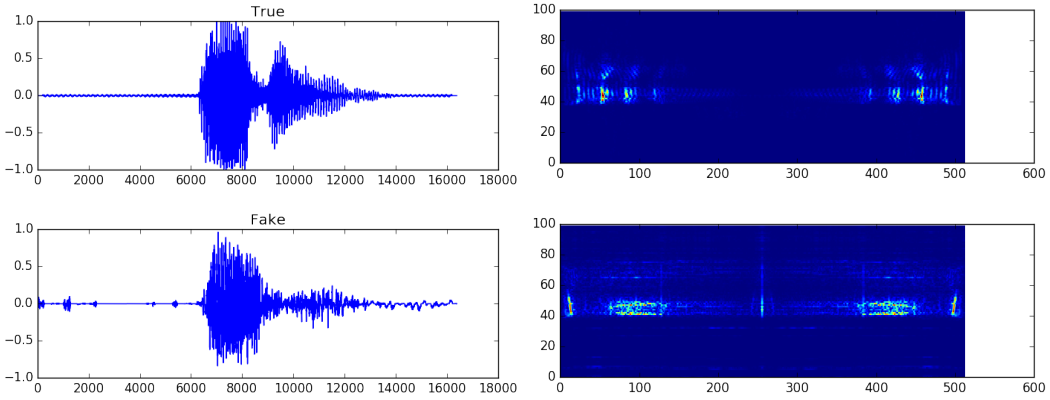
In the following plots, we originally show the true signal in comparison with the pure noise that was fed to the generator prior to the start of the training process. At this stage, the generated signal, and its corresponding spectrogram, has the characteristic structure of white noise as it is nothing more that random samples from a uniform distribution in the interval $[1-, 1]$.



(a) Signal (b) Spectrogram

Figure 2: Comparison between the true signal and the synthetic one before training

After 22500 update steps, the generator is able to produce sounds that mimic the original ones. The following comparison between the waveforms and spectograms of the true and generated sounds validate this assumption:



(a) Signal (b) Spectrogram

Figure 3: Comparison between the true signal and the synthetic one after 22500 update steps have taken place

From the figure 3 it is clear now that the signal sampled from the Generator starts to resemble the usual structure of speech signals. This is also true for the corresponding spectrogram. Nonetheless, we infer that the generated signal is much noisier but, we believe that after considerably much longer training time these side-effects will partially eliminated.

5 Discussion and Conclusion

5.1 Discussion

Through this work, we presented a Deep Generative Adversarial Neural Network that after extensive training is able to mimic human sounds in a fair degree. Although the performance of our network in terms of accuracy might not seem optimal, the fact that more than 20% of our generated samples were mistaken as human speech is encouraging. Despite the fact that the available time and resources we had at our disposal did not allow for efficiently training our network for the desirable amount of time needed, we can report that our implementation of WaveGAN was able to generate spoken language utterances, which resemble to the original ones in terms of sound quality, as presented in

section 4.2. In fact, Donahue et. al.[5] suggest that more than 2 weeks of training were required to efficiently train their network, while our work was delivered after a training process that ran for less than 3 days. Moreover, the deviations between the fake samples and real spoken language can be explained because of the use of the Speech Commands dataset, which is claimed to not be a suitable dataset, due to the fact that its recordings are distorted by background noise[5]. Furthermore, the resemblance between the true sample and the generated one paves the way for more training, where a robust audio generator can be reached. To succeed in this task, certain improvements can be tried on our network with the scope of enhancing its performance.

5.2 Future Work

In the future, we plan to develop our current implementation to the level of performance that similar state-of-the-art systems yield today i.e., WaveGAN. To do so, we first aim to train for a prolonged period of time our network and observe in what degree its results are further improved. Fine tuning the hyper-parameters of our network that affect the training of the model, is also task that can be conducted (in a small scale) in parallel with the previous one. The selected hyper-parameters will constitute the settings of a second-phase extensive training. Additionally, we intend to explore for potential benefits of switching from the time to the frequency domain i.e., through the implementation of SpecGAN[5]. Furthermore, we plan on performing a more extensive evaluation of our model, which includes the addition of qualitative measurements e.g. *mean opinion score* (MOS) for estimating the quality of the generated sounds, and the insertion of quantitative evaluation metrics, such as the *inception score*[18], nearest neighbor comparison and upsampling noise measurement. Last but not least, we look forward to examine the generalisation capabilities of our refined network on various datasets, beginning with the TIDIGITS dataset that includes clearer samples compared to SpeechCommands and the birds vocalization dataset, after all necessary pre-processing steps have been taken care of.

References

[1] Boesman, Peter. bird recordings. `https://www.xeno-canto.org/contributor/OOECIWCSWV`. Accessed at: 2018-05-26.

[2] Warden, Pete. speech commands dataset. `https://research.googleblog.com/2017/08/launching-speech-commands-dataset.html`. Accessed: 2017-12-15.

[3] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[4] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[5] C. Donahue, J. McAuley, and M. Puckette. Synthesizing audio with generative adversarial networks. *arXiv preprint arXiv:1802.04208*, 2018.

[6] H. Dudley. The automatic synthesis of speech. *Proceedings of the National Academy of Sciences*, 25(7):377–383, 1939.

[7] J. Engel, C. Resnick, A. Roberts, S. Dieleman, D. Eck, K. Simonyan, and M. Norouzi. Neural audio synthesis of musical notes with wavenet autoencoders. *arXiv preprint arXiv:1704.01279*, 2017.

[8] I. Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016.

[9] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[10] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5769–5779, 2017.

[11] C.-C. Hsu, H.-T. Hwang, Y.-C. Wu, Y. Tsao, and H.-M. Wang. Voice conversion from unaligned corpora using variational autoencoding wasserstein generative adversarial networks. *arXiv preprint arXiv:1704.00849*, 2017.

[12] 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.

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

[14] E. Moulines and F. Charpentier. Pitch-synchronous waveform processing techniques for text-to-speech synthesis using diphones. *Speech communication*, 9(5-6):453–467, 1990.

[15] 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 preprint arXiv:1609.03499*, 2016.

[16] S. Pascual, A. Bonafonte, and J. Serra. Segan: Speech enhancement generative adversarial network. *arXiv preprint arXiv:1703.09452*, 2017.

[17] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[18] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pages 2234–2242, 2016.

[19] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerry-Ryan, et al. Natural tts synthesis by conditioning wavenet on mel spectrogram predictions. *arXiv preprint arXiv:1712.05884*, 2018.

[20] J. Sotelo, S. Mehri, K. S. Kumar, J. F. Santos, K. Kastner, A. Courville, and Y. Bengio. Char2wav: End-to-end speech synthesis. 2017.

[21] A. Van Den Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 1747–1756. JMLR.org, 2016.

[22] Y. Wang, R. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio, et al. Tacotron: Towards end-to-end speech synthesis. *arXiv preprint arXiv:1703.10135*, 2017.

[23] T. Yoshimura. Simultaneous modeling of phonetic and prosodic parameters, and characteristic conversion for hmm-based text-to-speech systems. *Nagoya Institute of Technology, Japan*, 2002.