Reproduction of MelGAN: Generative Adversarial Networks for Conditional Waveform Synthesis

Evangelos Tsogkas[*]
School of Electrical Computer Engineering
National Technical University of Athens
evaggelostsogkas@mail.ntua.gr

Konstantinos Vilouras[*]
School of Electrical Computer Engineering
National Technical University of Athens
konstantinosvilouras@mail.ntua.gr

Andreas Tritsarolis[*]
School of Electrical Computer Engineering
National Technical University of Athens
andreastritsarolis@mail.ntua.gr

Abstract

Efficient audio synthesis is an inherently difficult machine learning task since it involves the modelling of both global and fine-scale structure for the production of coherent waveforms. Recent works focus primarily on non-autoregressive neural networks to learn an inverse mapping from a low-dimensional representation, i.e. a Mel spectrogram, back to the original waveform. In this paper, we investigate the validity of MelGAN [15] as a proposed model that is able to produce high-fidelity waveforms several orders of magnitude faster than an autoregressive counterpart. Focusing on speech synthesis, we trained our own version of the model from scratch, tested its sample generation speed on both CPU and GPU and also proposed a simple solution for the integration of our model in a common text-to-speech pipeline. The final results indicate that our model is roughly $6.5\times$ faster than realtime on CPU and $249\times$ faster than realtime on a single GPU, while fine-tuning on different types of input spectrograms yields promising results as well.

1 Introduction

Audio waveform synthesis is a rather challenging task due to the high temporal resolution of audio signals (16 kHz for speech and 44.1 kHz for universal audio) and the existence of fine-scale structure across a wide range of timescales. To alleviate these problems, we consider a commonly used compressed representation for audio, namely the perceptually weighted (mel) spectrogram. Although this low-dimensional time-frequency representation is efficient for a wide variety of audio related tasks, it does not suffice for the reconstruction of the original waveform, since all the information related to phase is completely discarded. In this work, while focusing explicitly on speech signals, we address the problem of learning a mapping between the mel spectrogram domain and the raw waveform domain, which is also referred to as *mel spectrogram inversion*. In the following paragraphs, various approaches to mel spectrogram inversion are discussed. These can be divided into three distinct categories that involve pure signal processing methods, autoregressive and non-autoregressive deep neural networks.

[*] All authors contributed equally to this work.

2 Related Work

Signal processing techniques attempt to reconstruct the original waveform by estimating magnitude and phase separately. Magnitude estimation is straightforward in the case of spectrograms and can be implemented using the inverse Short Time Fourier Transform (STFT). However, this inversion is infeasible after the application of the Mel filterbank, i.e. a non-linear transformation of the frequency scale. A simple heuristic involves the projection of the mel spectrogram onto the pseudoinverse of the mel basis that was used to generate it. On the contrary, common phase estimation methods include the iterative Griffin-Lim algorithm [5] which, given the STFT magnitude, alternates forward and inverse STFT operations until convergence. Modern attempts [22, 25] proposed improved versions of the Griffin-Lim algorithm in terms of convergence speed and reconstruction error. Moreover, in terms of speech vocoding, additional techniques were introduced in [20] that leverage three different algorithms to estimate the fundamental frequency (F0), the spectral envelope and the excitation signal, respectively. A crucial disadvantage of the aforementioned methods is that they introduce audible, "robotic" artifacts, which render the resulting waveform unintelligible.

Meanwhile, recent advances in neural autoregressive models enable modelling of raw waveforms one sample at a time, where each generated sample is conditioned on samples at all previous timesteps. WaveNet [28] is a generative, fully convolutional model that utilizes dilated causal convolutions to efficiently capture long range dependencies. It has been designed both for unconditional synthesis and also for speech generation conditioned either on speaker identity (in a multi-speaker scenario) or linguistic/phonetic features (for text-to-speech) aligned with the raw input audio. Despite the fact that it generates complex and high quality audio, it is computationally expensive and thus not suitable for real time applications. Additionally, SampleRNN [18] performs unconditional audio generation using the same sample conditioning scheme as WaveNet. It consists of multiple modules that operate hierarchically, i.e. higher level modules process increasingly longer timescales, while each module is conditioned on the previous level of the hierarchy. WaveRNN [9] is a single-layer recurrent neural network that operates on 16-bit audio and tries to minimize sampling time with techniques such as weight pruning (to enforce sparsity) and subscaling (a long sequence is divided into a batch of shorter sequences, which allows the generation of multiple samples at once). The state of the RNN is split into two parts that predict the 8 most significant (coarse) and the 8 least significant (fine) bits of the 16-bit audio sample, respectively. The 8 fine bits are conditioned on the 8 coarse bits, thus the output space, the model parameters and its memory requirements are significantly reduced.

Recently, a lot of research has focused on the development of non-autoregressive models to synthesize high temporal resolution audio. Because these models are highly parallelizable and, as a result, can fully take advantage of modern deep learning parallel computing processors, they are extremely faster than auto-regressive models. One family of such models relies on knowledge distillation, including Parallel WaveNet [29] and Clarinet [23]. Under this framework, the knowledge of an autoregressive teacher model is transferred to a smaller student model based on the Inverse Autoregressive Flows (IAF) [13]. Although the IAF students can be run in parallel at inference time and synthesize high quality speech at a reasonably fast speed, the auto-regressive nature of the flow makes calculations of the IAF inefficient. To overcome this, these works require a well-trained teacher model, to train a student network on an approximation to the true likelihood using a probability distillation objective based on the Kulback-Leibler divergence, along with additional perceptual loss terms. These approaches are hard to reproduce and deploy because of the difficulty of training these models successfully to convergence. The other family of non-autoregressive models are flow-based generative networks, including WaveGlow [24] and FloWaveNet [11]. WaveGlow eliminates the need for distilling a teacher model, and simplifies the learning process through maximum likelihood estimation by employing efficient bijective flows based on Glow [12]. However, it requires many parameters for its deep architecture with over 90 layers and as the authors note, it requires a week of training on 8 GPUs to get good quality results for a single speaker model. The inference process is parallel, so it is fast on GPU, but the large size of the model makes it impractical for applications with limited memory capacity.

Generative adversarial networks (GANs) [4] are popular models for sample generation, which have been the dominating paradigm for image generation [6, 10], image-to-image translation [31] and video-to-video synthesis [30]. There were several early attempts applying GANs to audio generation tasks, but achieved limited success [2]. Engel et al. [3] use GANs to generate musical timbre by modelling STFT magnitudes and phase angles instead of modelling raw waveform directly. Neekhara

et al. [21] propose using GANs to learn mappings from mel spectrogram to simple magnitude spectrogram, which is to be combined with phase estimations to recover raw audio waveform. Yamamoto et al. [32] use GANs to distill an autoregressive model that generates raw speech audio, however their results show that adversarial loss alone is not sufficient for high quality waveform generation; it requires a KL divergence based distillation objective as a critical component. Recently, there has been a new wave of modeling audio using GANs, as non-AR models targeting to fast audio generation. Specifically, MelGAN [15], Parallel WaveGAN [33] and GAN-TTS [1] have shown promising performance on waveform generation tasks. They all rely on an adversarial game of two networks: a generator, which attempts to produce samples that mimic the reference distribution, and the discriminator, which tries to differentiate between real and generated samples. The input of MelGAN and Parallel WaveGAN is mel-spectrogram, while the input of GAN-TTS is linguistic features. Hence MelGAN and Parallel WaveGAN are considered as neural vocoders, while GAN-TTS is a stand-alone acoustic model. Meanwhile, Parallel WaveGAN and MelGAN both use auxiliary loss, i.e., multi-resolution STFT loss and feature matching loss, respectively, so they converge significantly faster than GAN-TTS.

Most related to our work, Jang et al. [8] propose Universal MelGAN, a vocoder which builds upon MelGAN by adding multi-resolution spectrogram discriminators, in an attempt to sharpen the spectral resolution of the generated waveforms. This alleviates the over-smoothing problem in the high frequency band, thus enabling the model to generate further realistic waveforms. In parallel, Yang et al. [34] propose Multi-band MelGAN, a yet another waveform generation model targeted to the Text-to-Speech (TTS) task. Like the aforementioned work, they build upon MelGAN and further extending it, by adding multi-band processing, or simply put, "the generator takes mel-spectrograms as input and produces sub-band signals which are subsequently summed back to full-band signals as discriminator input". Finally, Kong et al. [14] propose HiFi-GAN, a GAN-based model for efficient and high-fidelity speech synthesis. While its architecture is similar to MelGAN, it introduces a new discrimator type, called Multi-Period Discriminator (MPD), which consists of several sub-discriminators, each handling a portion of periodic signals of input audio, in order to identify long-term dependencies within the audio fragment. Evaluating the above discriminator not only with the aforementioned model, but also with MelGAN, demonstrates the effect of MPD towards synthesized audio.

3 Proposed Method

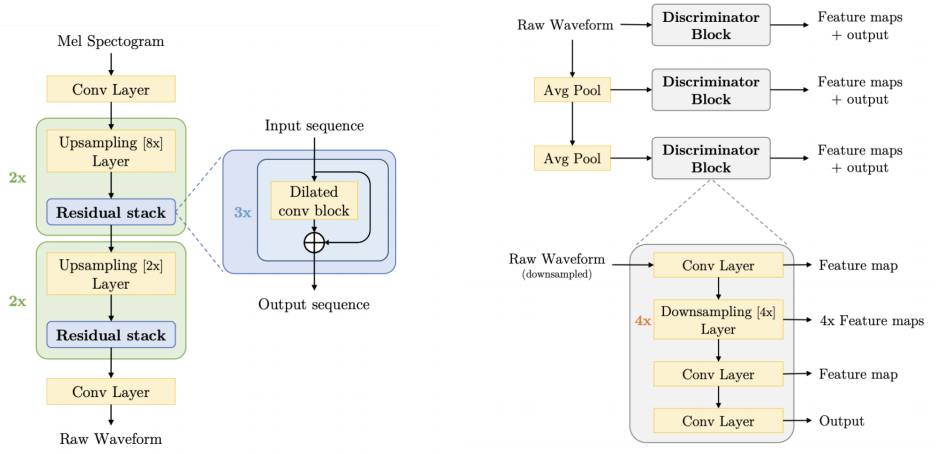
3.1 The Generator

The generator is a fully convolutional feed-forward network with mel-spectrogram s as input and generates raw waveform x as output. The mel-spectrogram is at a $256\times$ lower temporal resolution, which is upsampled by a stack of transposed convolutional layers and residual blocks with dilated convolutions. A key part of the generator is the induced receptive field. The authors designed the generator to put an inductive bias that there is long range correlation among the audio time steps. Since residual blocks with dilation after each upsampling layer are added to the generator, temporally far output activations of each subsequent layer have significant overlapping inputs. And since the receptive field of a stack of dilated convolution layers increases exponentially with the number of layers, incorporating these in the generator efficiently increases the induced receptive fields of each output time-step, which leads to a better long range correlation. Unlike traditional GANs, the MelGAN generator does not use a global noise vector as input and in order to deal with "checkerboard artifacts" in audio, it uses a kernel with size multiple of stride and dilation that grows as a power of the size of the kernel. The authors also chose to use weight normalization in all layers of the generator after experimenting with other normalization techniques, such as instance and spectral normalization.

3.2 The Discriminator

For the discriminator, we use a multi-scale architecture that consists of three discriminators (D_1, D_2, D_3) with identical network structure, but different audio scales. More specifically, D_1 operates on the scale of raw audio, while D_2 and D_3 operate on raw audio downsampled by a factor of $\times 2$ and $\times 4$, respectively.

Downsampling is performed using Average Pooling using a 4×4 kernel. Multiple discriminators at different scales are motivated from the fact that audio has structure at different levels. This structure



(a) (b)

Figure 1: MelGAN Model Architecture: (a) Generator; and (b) Discriminator. Each upsampling layer is a transposed convolution with the kernel size being twice of the stride (which is same as the upsampling ratio for the layer). $256\times$ upsampling is done in 4 stages of $8\times$, $8\times$, $2\times$ and $2\times$ upsampling. Each residual dilated convolution stack has three layers with dilation 1, 3 and 9 using a 3×3 kernel, having a total receptive field of 27 timesteps. We use leaky-relu for activation. Each discriminator block has 4 strided convolution with stride 4.

has an inductive bias that each discriminator learns features for different frequency range of the audio. For example, the discriminator operating on downsampled audio, does not have access to high frequency component, hence, it is biased to learn discriminative features based on low frequency components only.

While a standard GAN discriminator learns to classify between distributions of entire audio sequences, window-based discriminator learns to classify between distributions of small audio chunks. Grouped convolutions are used to allow the larger kernel sizes for a small number of parameters. Since the discriminator loss is computed over the overlapping windows where each window is very large (equal to the receptive field of the discriminator), the MelGAN model learns to maintain coherence across patches. Both discussions regarding the Generator (c.f. Section 3.1) and the Discriminator, respectively, are illustrated at Figure 1.

3.3 Training Method

To train the GAN, we follow the authors' process, thus using the hinge loss version of the GAN objective [17, 19], as described in Equations 1 and 2, respectively.

$$\min_{D_k} \mathbb{E}_x \left[max(0, 1 - D_k(x)) \right] + \mathbb{E}_{s,z} \left[max(0, 1 + D_k(G(s, z)) \right], \forall k = 1, 2, 3 \tag{1}$$

$$\min_{G} \mathbb{E}_{s,z} \left[\sum_{k=1,2,3} -D_k(G(s, z)) \right] \tag{2}$$

where x represents the raw waveform, s represents the conditioning information (e.g., mel-spectrogram) and z represents the Gaussian noise vector. In addition to the discriminator's signal, we use a feature matching objective [16] to train the generator. This objective minimizes the L1 distance between the discriminator feature maps of real and synthetic audio. Intuitively, this can be seen as a learned similarity metric, where a discriminator learns a feature space that discriminates the fake data from real data.

3.4 Model Architecture

The MelGAN architecture we opt to train, is the default – according to the original paper – architecture, namely

Generator: The architecture of the generator consists of 4 transposed convolutional layers to upsample the input sequence and each transposed convolutional layer is followed by 3 residual blocks with dilated convolutions.

Discriminator: The architecture of the discriminator consists of 3 discriminator blocks and each discriminator block consists of 7 convolutional layers.

Finally, all other training-related parameters, e.g., optimizer, loss function, etc., are set according to the original paper in their respective default values.

4 Experimental Study

4.1 Experimental Setup

The code for MelGAN was implemented in Python using PyTorch[*]. The machine we used is a High Performance Computing (HPC) server, called ARIS[*], an IAAS service for the Greek Research and Academic Community, using a single VM consisting of one Nvidia Tesla K40 GPU with 12GB VRAM.

Due to dependency constraints, we reduced the dependencies on third-party libraries to a bare minimum, thus resorting to PyTorch for optimal GPU utilization. We also refactored our code to utilize Torchaudio during the preprocessing stage, but this was not a viable solution due to ARIS setup. Moreover, due to resource constraints (the maximum runtime of ARIS is 48 hours), our model was trained for approximately 12 days (or ≈ 2700 epochs), i.e. less than the overall training time of the original MelGAN. The dataset and code used is available in the team's directory (Team 11), with the corresponding logs (and outputs) per run, located in their respective sub-directories.

4.2 Dataset

For our experiments, we used the LJSpeech dataset [7] that consists of 13,100 audio clips recorded in a single-speaker setting. The audio format is 16-bit PCM with a sample rate of 22.05 kHz, while the recording duration varies from 1 to 10 seconds, leading to a dataset size of approximately 24 hours in total. For testing we use the first 10 audio samples, while for training the rest 13,090.

4.3 Preprocessing

To ensure that the network is trained in a stable manner, we limit the input (raw waveform) based on a predefined maximum sequence length (8192 samples, which roughly corresponds to 0.37 seconds). We achieve this either by extracting a random segment of the waveform or through padding if the input's length is smaller than the maximum. Moreover, we normalize the extracted sequence with respect to its infinity norm, i.e. the absolute value of the largest peak in the waveform. The last step is optional and is related to sequence augmentation. Specifically, the sequence amplitude is scaled by a constant drawn from $\mathcal{U}(0.3, 1.0)$. We choose 1.0 as the upper bound, since larger values could lead to a significant amount of distortion.

The last step involves the extraction of ground truth mel spectrograms. First, the Short Time Fourier Transform is applied to the input sequence with parameters: number of FFT points $N_{fft} = 1024$, window length equal to 1024, while the hop length is set to 256 samples. Next, we extract the magnitude spectrogram from the real and imaginary part of the Fourier Transform, respectively. Furthermore, we create a Mel filterbank that spans across the range [0, 11025] Hz, where the upper bound denotes the Nyquist frequency, i.e. half of the sampling rate, for this setup and use it to project the FFT bins onto 80 Mel bins in total. Note that we implemented Slaney's normalization function (divides the weights of the Mel matrix by the appropriate Mel band width) from scratch since it is not

[*] `https://pytorch.org/`

[*] `https://hpc.grnet.gr/`

Table 1: Difference in the number of parameters and inference speed (measured in kHz, i.e. $10^3\cdot$ samples / sec) for an autoregressive (Waveglow) and a non-autoregressive model (MelGAN), respectively. Both models are benchmarked on the same hardware.

Model	Parameters ($\cdot 10^6$)	CPU Speed (in kHz)	GPU Speed (in kHz)
Waveglow [24]	87.9	1.9123	54.4060
MelGAN (ours)	**4.26**	**142.4461**	**5488.7789**

provided in Librosa 0.6.0. Finally, we perform logarithmic scaling of the resulting Mel-spectrogram, which serves the purpose of dynamic range compression.

4.4 Experimental Results

Synthesis Quality Due to the absence of objective evaluation metrics, a common method to compare synthesized audio against its ground truth is by means of Mean Opinion Score (MOS). We could not conduct a MOS test since its validity is heavily relied to the large number of participants. On the contrary, we provide a subjective performance comparison in Appendix A and then we describe the procedure of MOS calculation in Appendix B, respectively.

Inference Speed To inspect the sample generation speed of our model, we compare it against the pre-trained version of Waveglow [24] that is publicly available in Pytorch Hub. We run both models (on evaluation mode) once on CPU and GPU respectively and keep track of the generated samples using the Mel spectrogram of a 10 second recording as input. We use NVIDIA GTX 1660 Ti for the GPU benchmark and Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz processor for the CPU benchmark, tested on only 1 CPU core. From the results denoted in Table 1 it is clear that our model achieves high sample generation speed that is $6.5\times$ faster than realtime on CPU and $249\times$ faster than realtime on GPU, respectively. Note that the comparison with respect to realtime performance is based on the original sampling rate of the audio clips used here, i.e. 22.05 kHz.

Text-to-Speech In order to verify the effectiveness of MelGAN as a vocoder for the text-to-speech (TTS) task, the authors combined the MelGAN vocoder with Text2mel, an improved version of the open-source char2wav model [27]. The reason the authors did not choose the state-of-the-art Tacotron2 [26] for the mel-spectrogram prediction part is because it is slow to train and performs mel-frequency clipping which would hurt the performance of MelGAN. However, we show that it is in fact possible to combine MelGAN with Tacotron2 if they are both trained on the same dataset. The Tacotron2 model takes syllable sequence with tone index and prosodic boundaries as input and outputs predicted mel-spectograms which are subsequently fed into the MelGAN vocoder to produce a waveform. We used the transcriptions of the LJSpeech dataset to extract the mel-spectograms from the pre-trained Tacotron2 model provided by NVIDIA in the Pytorch Hub repository and then fine-tuned MelGAN for just 60 epochs. The results are comparable to the state-of-the art WaveGlow [24]. Fine-tuning is necessary, because Tacotron2 uses a different preprocessing and as a result produces slightly different mel-spectograms than what MelGAN expects, which causes the audio quality to be unsatisfactory. Specifically, there are two differences in preprocessing: 1) Tacotron2 uses the natural logarithm (base e) for dynamic range compression, while the literature consistently proposes the use of the common logarithm (base 10) which is linked to the decibel scale and 2) Mel filterbank of Tacotron2 spans the frequency range of [125, 7600] Hz (also referred to as mel frequency clipping), while here we use the full frequency range up until the Nyquist frequency. In the provided notebook[*] we show samples of how MelGAN sounds before fine-tuning, after trying to transform Tacotron2 output to match our preprocessing and after tuning on Tacotron2 output. Additionally, we show spectrogram inversion samples after and mid training and provide the code and our pre-trained model to try text-to-speech. We conclude that MelGAN adapts well on the end-to-end setting with fine-tuning, without the need to train other acoustic models from scratch.

Additionally, in Figure 2 we present the spectrograms that correspond to the generated waveforms from MelGAN before and after fine-tuning, as well as from Waveglow since it is also trained on Mel

[*] `https://colab.research.google.com/drive/12yzKj9u7DYed26-YopbSHw4_IRvw__FM?usp=sharing#scrollTo=k7j9IFM04d3h`

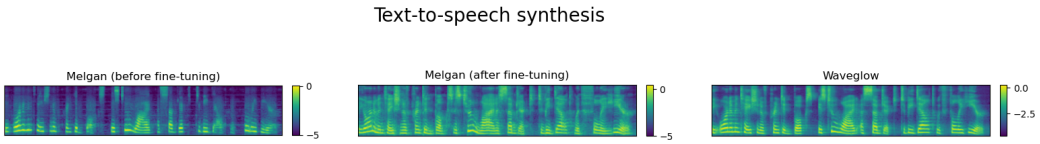


Figure 2: Text-to-speech synthesis results for the same transcript. From left to right: i) MelGAN's output before fine-tuning, ii) MelGAN's output after fine-tuning, iii) Waveglow's high-quality output (serves as ground truth here).

spectrograms generated from Tacotron2 model. We observe an absence of high frequency patterns in the output of the initial MelGAN model (before fine-tuning), which results in a low-quality waveform. On the contrary, fine-tuning on Tacotron2 outputs yields significant improvement over the existing results, which can be validated through Waveglow's generated audio as well. However, there exist some minor audible artifacts, i.e. frequency patterns that do not occur in the waveform generated by Waveglow model, after fine-tuning that slightly degrade the overall quality of the generated speech. The aforementioned observations show that fine-tuning is an efficient method to match the output of Tacotron2 model with the expected input of MelGAN, while the artifacts can be dramatically reduced through a more robust alignment of ground truth and target mel spectrograms.

5 Conclusions and Future Work

In this work, we investigated the reproducibility and effectiveness of MelGAN in terms of Mel spectrogram inversion for the general task of speech synthesis. The key concept is its non-autoregressive architecture, which renders it as a highly parallelizable and extremely fast solution for the generation task. Another important attribute is the modelling of variable length audio due to the fully convolutional nature of this network. Moreover, the design of the window-based discriminator that operates on three different time scales, along with the formulation of the training objective as the combination of hinge loss and feature matching loss, guide the generator to capture high frequency patterns and create high quality samples. Furthermore, while the training procedure is time-consuming, the network learns an efficient mapping after roughly 1200 epochs and then improves on details that make the generated waveform more intelligible. Upon further inspection of the final results, it is clear that MelGAN demonstrates competitive performance over its autoregressive counterparts, while enabling fast inference on both CPU and GPU, respectively. However, it can also be observed that the model struggles with vocoding of large sequences, i.e. 5 seconds long or more, since it introduces audible artifacts that slightly harm the quality of generated audio.

For future work, an interesting research direction involves the creation of a hybrid model that combines the best from both worlds, i.e. autoregressive and non-autoregressive models. Since the former are proven to generate high-quality audio samples, it would be meaningful to insert an autoregressive module inside a non-autoregressive model to guide the generation process whenever necessary, i.e. for certain points the generator struggles with. This could further improve the quality of synthesis while keeping the model lightweight at the same time.

6 Contributions

While all authors contributed equally to this project, *Evangelos Tsogkas* focused mainly on creating the Generator, Training Loop, and fine-tuning on Tacotron2, *Konstantinos Vilouras* focused mainly on – due to dependency issues – creating multiple loading and preprocessing pipelines for the LJSpeech dataset, and finally *Andreas Tritsarolis* focused mainly on the Discriminator, unified the code modules in a single pipeline, in order to be executed within ARIS HPC, and coordinated its output per simulation.

References

[1] M. Bińkowski, J. Donahue, S. Dieleman, A. Clark, E. Elsen, N. Casagrande, L. C. Cobo, and K. Simonyan. High fidelity speech synthesis with adversarial networks, 2019.

[2] C. Donahue, J. J. McAuley, and M. S. Puckette. Synthesizing audio with generative adversarial networks. *CoRR*, abs/1802.04208, 2018.

[3] J. H. Engel, K. K. Agrawal, S. Chen, I. Gulrajani, C. Donahue, and A. Roberts. Gansynth: Adversarial neural audio synthesis. *CoRR*, abs/1902.08710, 2019.

[4] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 2672–2680, Cambridge, MA, USA, 2014. MIT Press.

[5] D. Griffin and J. Lim. Signal estimation from modified short-time fourier transform. In *ICASSP '83. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 8, pages 804–807, 1983. doi: 10.1109/ICASSP.1983.1172092.

[6] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville. Improved training of wasserstein gans. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 5769–5779, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

[7] K. Ito and L. Johnson. The lj speech dataset. `https://keithito.com/LJ-Speech-Dataset/`, 2017.

[8] W. Jang, D. Lim, and J. Yoon. Universal melgan: A robust neural vocoder for high-fidelity waveform generation in multiple domains. *CoRR*, abs/2011.09631, 2020.

[9] N. Kalchbrenner, E. Elsen, K. Simonyan, S. Noury, N. Casagrande, E. Lockhart, F. Stimberg, A. Oord, S. Dieleman, and K. Kavukcuoglu. Efficient neural audio synthesis. In *International Conference on Machine Learning*, pages 2410–2419. PMLR, 2018.

[10] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=Hk99zCeAb`.

[11] S. Kim, S.-G. Lee, J. Song, J. Kim, and S. Yoon. FloWaveNet : A generative flow for raw audio. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3370–3378. PMLR, 09–15 Jun 2019. URL `http://proceedings.mlr.press/v97/kim19b.html`.

[12] D. P. Kingma and P. Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[13] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling. Improved variational inference with inverse autoregressive flow. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 4743–4751, Red Hook, NY, USA, 2016. Curran Associates Inc. ISBN 9781510838819.

[14] J. Kong, J. Kim, and J. Bae. Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis. In *Advances in Neural Information Processing Systems*, volume 33, pages 17022–17033. Curran Associates, Inc., 2020.

[15] K. Kumar, R. Kumar, T. de Boissiere, L. Gestin, W. Z. Teoh, J. Sotelo, A. de Brébisson, Y. Bengio, and A. C. Courville. Melgan: Generative adversarial networks for conditional waveform synthesis. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[16] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther. Autoencoding beyond pixels using a learned similarity metric. In *ICML*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1558–1566. JMLR.org, 2016.

[17] J. H. Lim and J. C. Ye. Geometric GAN. *CoRR*, abs/1705.02894, 2017.

[18] 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.

[19] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. Spectral normalization for generative adversarial networks. In *ICLR*. OpenReview.net, 2018.

[20] M. Morise, F. Yokomori, and K. Ozawa. World: A vocoder-based high-quality speech synthesis system for real-time applications. *IEICE Transactions on Information and Systems*, E99.D(7): 1877–1884, 2016. doi: 10.1587/transinf.2015EDP7457.

[21] P. Neekhara, C. Donahue, M. Puckette, S. Dubnov, and J. McAuley. Expediting tts synthesis with adversarial vocoding. In *INTERSPEECH*, 2019.

[22] N. Perraudin, P. Balazs, and P. L. Søndergaard. A fast griffin-lim algorithm. In *2013 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 1–4, 2013. doi: 10.1109/WASPAA.2013.6701851.

[23] W. Ping, K. Peng, and J. Chen. Clarinet: Parallel wave generation in end-to-end text-to-speech. *ArXiv*, abs/1807.07281, 2019.

[24] R. Prenger, R. Valle, and B. Catanzaro. Waveglow: A flow-based generative network for speech synthesis. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3617–3621, 2019. doi: 10.1109/ICASSP.2019.8683143.

[25] J. L. Roux, H. Kameoka, N. Ono, and S. Sagayama. Fast signal reconstruction from magnitude stft spectrogram based on spectrogram consistency. In *Proc. of International Conference on Digital Audio Effects DAFx '10*, 2010.

[26] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerry-Ryan, R. Saurous, Y. Agiomyrgiannakis, and Y. Wu. Natural tts synthesis by conditioning wavenet on mel spectrogram predictions. *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4779–4783, 2018.

[27] J. M. R. Sotelo, S. Mehri, K. Kumar, J. F. Santos, K. Kastner, A. C. Courville, and Y. Bengio. Char2wav: End-to-end speech synthesis. In *ICLR*, 2017.

[28] A. van den 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. In *Arxiv*, 2016. URL `https://arxiv.org/abs/1609.03499`.

[29] A. van den Oord, Y. Li, I. Babuschkin, K. Simonyan, O. Vinyals, K. Kavukcuoglu, G. van den Driessche, E. Lockhart, L. Cobo, F. Stimberg, N. Casagrande, D. Grewe, S. Noury, S. Dieleman, E. Elsen, N. Kalchbrenner, H. Zen, A. Graves, H. King, T. Walters, D. Belov, and D. Hassabis. Parallel WaveNet: Fast high-fidelity speech synthesis. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3918–3926. PMLR, 10–15 Jul 2018.

[30] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, G. Liu, A. Tao, J. Kautz, and B. Catanzaro. Video-to-video synthesis. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[31] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8798–8807, 2018. doi: 10.1109/CVPR.2018.00917.

[32] R. Yamamoto, E. Song, and J.-M. Kim. Probability density distillation with generative adversarial networks for high-quality parallel waveform generation. In *INTERSPEECH*, 2019.

[33] R. Yamamoto, E. Song, and J.-M. Kim. Parallel wavegan: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6199–6203, 2020. doi: 10.1109/ICASSP40776.2020.9053795.

[34] G. Yang, S. Yang, K. Liu, P. Fang, W. Chen, and L. Xie. Multi-band melgan: Faster waveform generation for high-quality text-to-speech. In *2021 IEEE Spoken Language Technology Workshop (SLT)*, pages 492–498, 2021. doi: 10.1109/SLT48900.2021.9383551.

A Appendix - Performance Evaluation

In this section we provide a comparison between the ground truth and generated waveforms that belong to the test set. The corresponding Mel spectrograms are depicted in Figure 3 where each row refers to a single audio recording. We can observe that the reconstruction of the original waveforms is successful since both global and fine-scale structure is retained. Specifically, the most dominant frequency patterns are preserved in the synthesized audio with the correct amplitude scale, i.e. the model does not introduce any type of distortion. However, the model seems to struggle with silent areas that occur, for example, during the transition between phonemes. This leads to points in the time domain where the resulting waveform is non-smooth (the effect is similar to stuttering) and as a result slightly degrades the overall quality of the generated speech.

Note that the evaluation of speech synthesis is based on subjective criteria and may differ from person to person. Thus, our submission is accompanied with a notebook that contains all the generated samples, along with their ground truth, for the readers to draw their own conclusions.

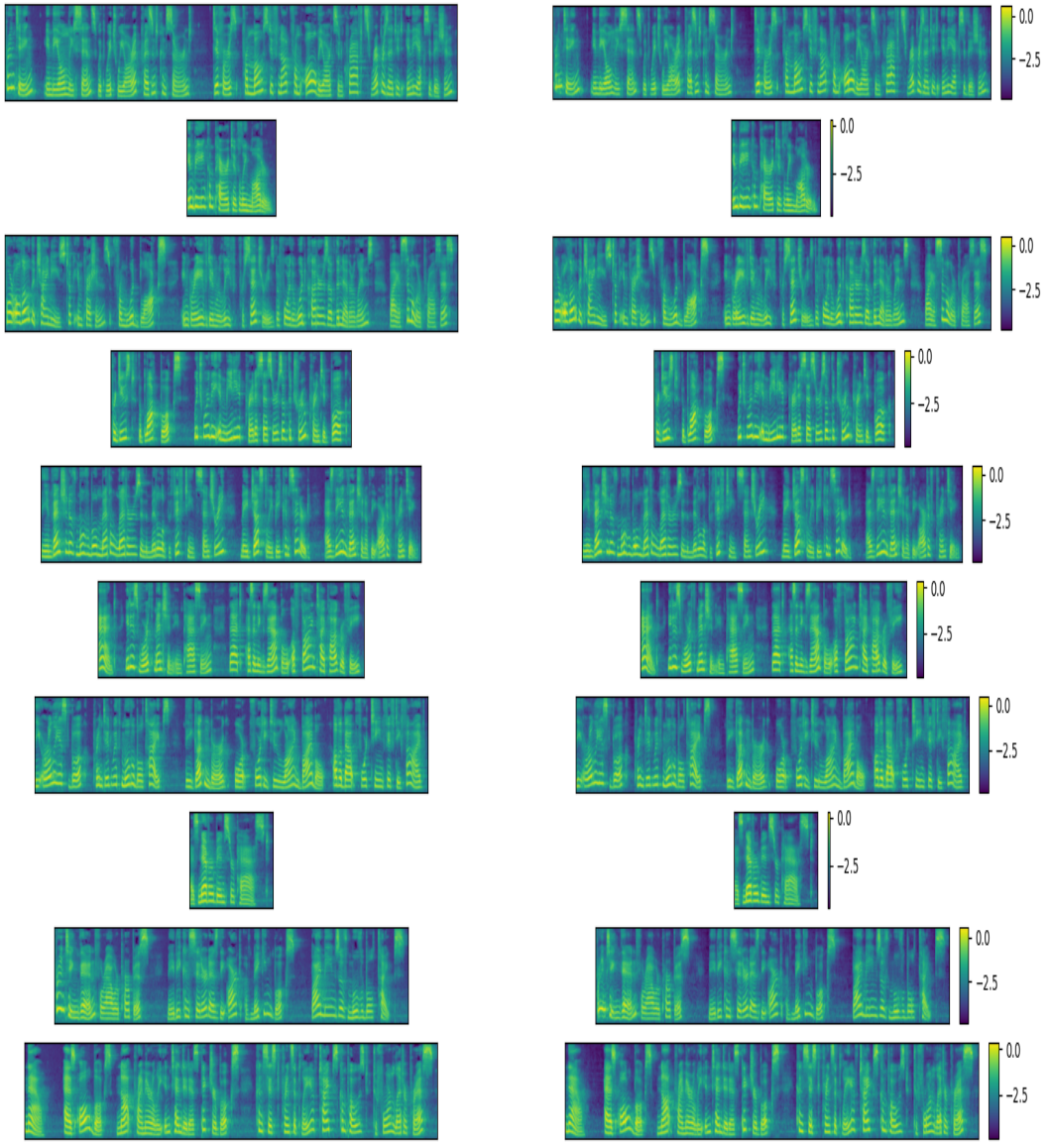


Figure 3: Mel spectrograms for ground truth (left column) vs generated (right column) audio waveforms.

B Appendix - Mean Opinion Score (MOS)

A common evaluation method for the task of audio synthesis is based on Mean Opinion Score (MOS). These tests are usually conducted on crowdsourcing platforms such as Amazon Mechanical Turk and their validity is heavily relied on the large amount of participants. During the test, each user is requested to listen to a recording through headphones and assign a rating in range 1-5 based on a 5-point Likert scale, where 1 denotes really low and 5 really high audio quality. After gathering all the evaluations N_i, we calculate the MOS $\hat{\mu}_i$ and the 95% confidence interval CI_i for each individual model i as follows:

$$\hat{\mu}_i = \frac{1}{N_i} \sum_{k=1}^{N_i} m_{i,k}$$

$$CI_i = \left[\hat{\mu}_i - 1.96 \frac{\hat{\sigma}_i}{\sqrt{N_i}},\ \hat{\mu}_i + 1.96 \frac{\hat{\sigma}_i}{\sqrt{N_i}} \right]$$

where σ_i denotes the standard deviation of the collected scores.