Audio Translation Tasks with Conditional Adversarial Networks

1st Ahmad Moussa
Advanced Multimedia Systems Lab.
Department of Fundamental Science and Engineering
Waseda University
Tokyo, Japan
ahmad.moussa@fuji.waseda.jp

2nd Hiroshi Watanabe
Advanced Multimedia Systems Lab.
Department of Fundamental Science and Engineering
Waseda University
Tokyo, Japan
hiroshi.watanabe@waseda.jp

***Abstract*—This paper explores the applicability of conditional generative adversarial networks in audio-to-audio translation problems and proposes a neural network architecture capable of doing so. Recent advances have shown than causal convolutions can be effective for modeling raw audio when their kernel is dilated by many factors, in contrast to previous techniques that utilized recurrent approaches. Embedding such convolutions within a conditional GAN architecture allows the targeted generation of raw audio given a certain input. This architecture can then be used to learn and simulate certain translative operations applied to an input signal. This creates the defined problem of havving to converting one audio signal into another which has different characteristics. We also propose a novel discriminator structure for the evaluation of generated audio.**

***Index Terms*—conditional generative adversarial networks, causal dilated convolutions, audio effects, signal processing**

I. Introduction

The term "Audio effect" generally describes a transformation that is applied to an audio signal. It is important to make the distinction, that the word "transform" here does not refer to a mathematical operation that performs a domain conversion. An example of such a transform would be the Fourier Transform, which decomposes a signal into it's constituent frequencies and essentially converts a signal from the time domain into the frequency domain. The transformations that will be discussed in the remainder of this paper should not be regarded as such, but rather as translative operations, or operations of conversion.

Pix2Pix [1] serves as a good illustrative example for our purposes. Their objective is to provide a solution for image-to-image translation problems, where images are "translated" into different images, that are structurally very similar, but have a different style and character. For example, colorizing Black-and-White images could be regarded as such a translation. Structurally, both images are identical, but their form of presentation is very different. Furthermore, converting hand-drawn sketches into realistic images is also such a translation. Even though the pixel structure that represents the object will be slightly morphed, the observable object is still the same, but now has a different appearance.

This comparative explanation should make the purpose of audio effects clear. It is to change the way a sound is perceived by our ears. For example, low pass filters can be used to smoothen harsh sounds and attenuate certain frequencies present in that sound, figure I illustrates the effect of a low-pass filter on an audio signal. Other effects, like reverbs can be used to place a sound in a certain space, such as a church hall, a large cave or merely a small broom-closet. Some effects even purposefully distort a signal, such as distortion effects, that have made possible many iconic guitar sounds throughout the history of western music. Traditionally, these types of operations fall into the domain of signal processing and can be achieved with analog and digital circuits. Entire books have been dedicated to the topic of construction and design of such circuits [2]. Nowadays, these types of effects are most prominently used in music production, sound engineering and sound design, where they most notably occur in the forms of software (VST plugins) or hardware (effect pedals), and serve as sound-shaping and sound-sculpting tools.

This paper proposes a neural network architecture that can serve as a general purpose framework capable of learning and simulating certain audio effects. Such a framework could also be used for tasks other than audio effects, such a denoising audio samples, similarly to SEGAN [3] for example, or learning the specific characteristics of a certain hardware circuit (naturally only if it possible to generate and record enough training data with it). In section II we discuss previous network architectures that contribute to this work. Section III describes the proposed architecture and how it is different from previous works. And finally, section IV explains how the model was trained and shows the experimental results that have been achieved so far.

II. Related Work

This network architecture is generally based on the conditional GAN [4] architecture proposed in the Pix2Pix paper [1], combined with the proven effectiveness of Causal Dilated Convolutions proposed in Wavenet [5].

A. Pix2Pix: Image-to-Image Translation with Conditional Adversarial Networks

Pix2Pix proves that Generative Adversarial Networks [6] can generate excellent fakes when they are applied in a

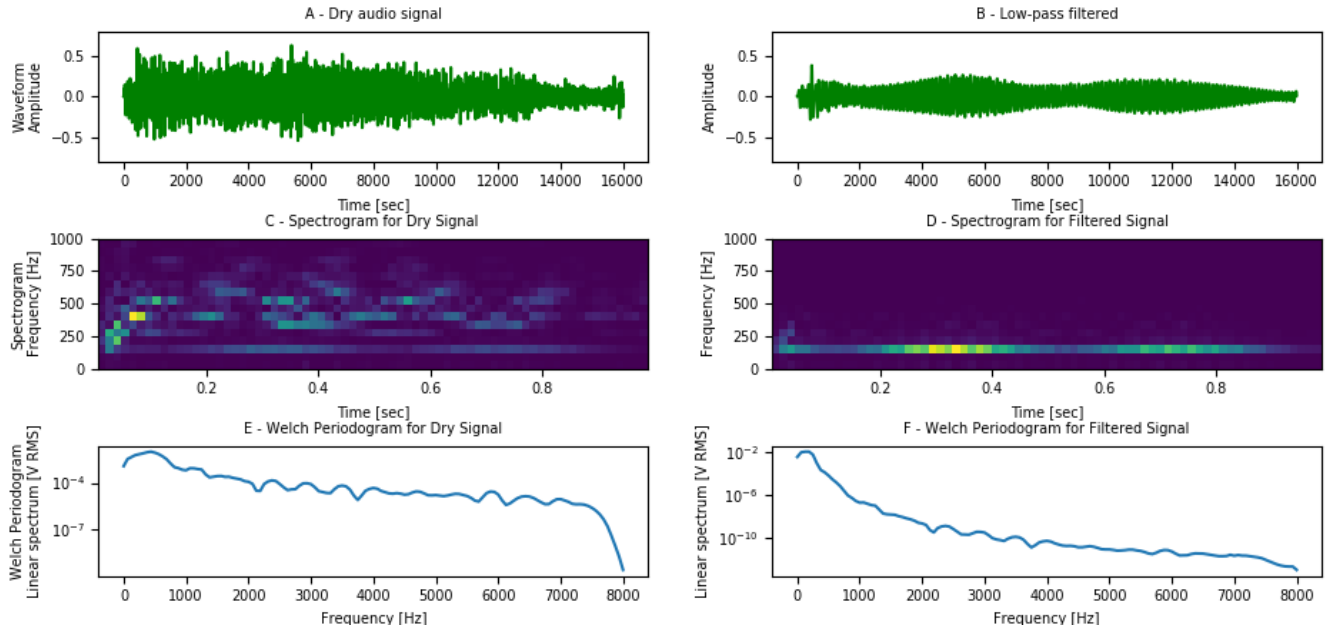

Fig. 1. We apply a low-pass filter with a cut-off frequency of 129Hz and resonance of 1.85 at the cut-off frequency (or Q factor), to a synthetic brass sound. Figures A and B show the effect that this has on the overall shape of the waveform. Figures C and D show the respective spectrograms of the dry and effected signal. The high frequencies of the signal are attenuated and not present in D, whilst D has more resonant frequencies around 129Hz. The welch periodogram F shows that the spectral density of higher frequencies is much lower than that of the dry signal as shown in figure E.

conditional setting [4]. Originally, GANs learn to generate an output given a random noise vector as input. In the conditional variant, this noise vector is replaced by a condition, which is in other terms an input to the network. In the case of Pix2Pix this input would be an image of some sort. This demands an appropriate architecture that is capable of taking in such an input.

The generative component of the conditional GAN that constitutes the Pix2Pix architecture, is an auto-encoder-like structure [7]. While preserving the encoder-decoder components of an auto-encoder, it also features residual connections [8] connecting the convolutional blocks in the encoder to their counterparts in the decoder. This essentially allows an important flow of features forward, while at the same time it allows gradients to propagate deeper into the network. These skip connections insinuate the shape of the ”U” when this architecture is represented in form of a diagram, hence it has been coined with the name ”U-Net” [9]. The convolutional layers present in the encoder perform a spatial reduction on the input. At the bottleneck point of the generator we should have achieved some sort of latent representation of the input. Generally this latent representation is a machine understandable encoding of the original image, which will then be handed to the decoder, which in turn will upsample this encoding and reconstruct an output image. Pix2Pix also employs an interesting and unusual type of discriminator [10] tailored for their purpose which will be discussed in section III-B

B. *Wavenet: a generative model for raw audio*

Wavenet [5] proposes a novel architecture for generating raw audio. Rather than utilizing recurrent units, such as in RNNs [11] and LSTMs [12], which contain an internal memory, they show that state-of-the-art results can be achieved by utilizing convolutions to model waveforms on a sample level. For sequential data, or any kind of data that exhibits some temporal (past - current - future) relationship, it is necessary to utilize causal convolutions. Meaning, that future time-steps should not be shown to convolution before it has generated the next time-step, as this current time-step can not conditionally depend on any future time-step. This condition is referred to as causality, and was first introduced in PixelCNN [13] where it was achieved with masked 2D convolutions.

A problem pointed out in their work is that for the convolution to correctly learn long term temporal relationships between samples, it requires a quite large receptive field. This can introduce a computational overhead. But luckily this can also be solved by using dilated filters [14] . By skipping some time-steps and stacking them in layers we can achieve an exponentially larger receptive field without requiring more computational cost, this effectively allows the convolution to learn long term dependencies that span more than a couple of sequential samples.

III. Proposed Architecture

A. *Generator*

We swap out the 2 Dimensional Convolutions in the Pix2Pix architecture for 1 Dimensional Causal Dilated Convolutions. It’s important to notice that the input to these convolutions

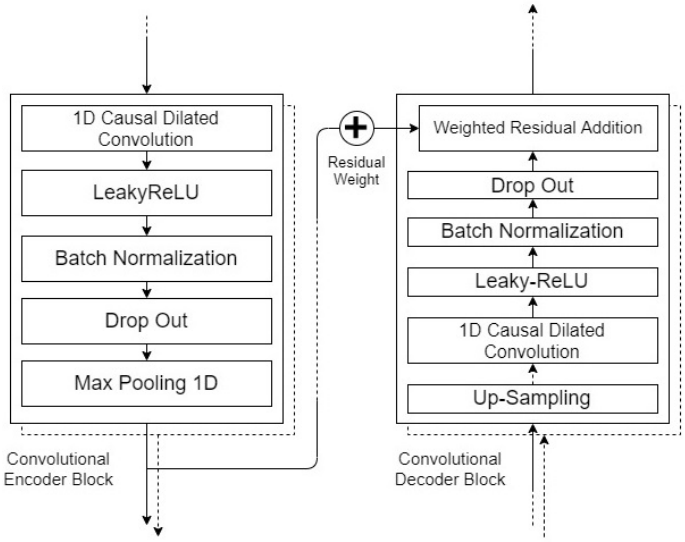

Fig. 2. Structure of encoding and decoding convolutional blocks. Note that dropout has been omitted from the first layer of the encoder. Batch normalization [15] is present though in the last layer of the decoder, as there is another final convolution block after the decoder, which does not contain batch normalization.

will be padded prior to convolutional operation, and hence the input width is not reduced, but a channel increase still occurs. This allows the convolutions to learn causal features, which can also be called temporal codecs [16]. We found that a kernel width of 1024 and a dilation rate of 8 achieved the best results, increasing width and dilation factor further did not yield any performance gains. The spatial reduction occurs through a 1 Dimensional Max Pooling layer. In the decoder we utilize the same type of convolution but with a 1 Dimensional Up-sampling layer, which simply just "repeats each temporal timestep n times along the time axis" (according to the keras documentation), where n is the up-sampling factor. This up-sampling factor is identical to the down-sampling factor used in the pooling layers in the encoder, and is a value of 4 consistently throughout the generator, with exception of the first two layers in the encoder, and the final and penultimate layers in the decoder.This lower up-sampling rate in the decoder allowed the generation of a much smoother waveform. Analogously, the down-sampling rate had to be attenuated in the first two layers of the encoder to ensure that the residual connections connecting the first two and last two layers carry over tensors of the same shape.

The activation function of choice is the LeakyReLU, which is applied throughout all the layers of the generator with exception of the final output layer, which utilizes a tanh activation. Initially we experimented with A Parametric ReLU activation, to give the generator more liberty in the gradient flow, but this actually introduced noise in the generated samples, when the output should simply just be the absence of sound.

Additionally we added a learn-able parameter to the skip connections such that the generator can weight the importance of certain skip connections between encoder and decoder. And finally we also use Batch-Norm [15] and Dropout [17] throughout the layers, except for the input layer, where no dropout is applied to prevent the loss of important data, and the last layer where it is not advantageous to use batch normalization. The final configuration of encoding and decoding blocks can be observed in figure 2. Figure 3 illustrates the shape of a tensor being passed through a convolutional layer and a pooling operation.

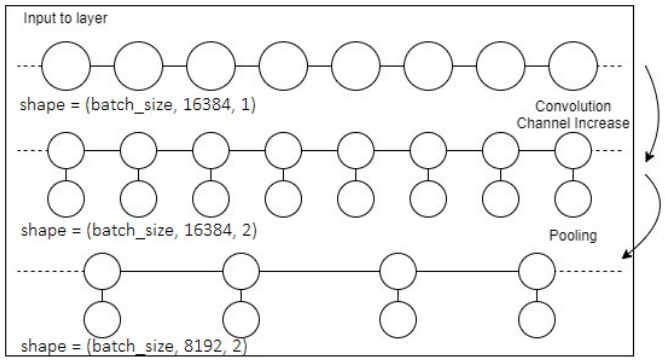

Fig. 3. Shapes of the input tensor passed through the very first layer of the generator. Channels increase while width is reduced.

B. Discriminator

Traditionally the discriminator in adversarial networks is a binary classifier which learns how to tell apart fake from real images and gives meaningful feedback to the generator. In Pix2Pix the discriminator does not do a binary classification. It operates on the level of individual patches [14]. Given a small patch that is an NxN sub-square of pixels of an image, it will determine if that entire patch is probable to be real or not. This patch is run convolutionally across the image and the response is averaged.

We attempted to use this type of discriminator in an audio setting, where the NxN patch is now represented by slices of length N of the waveform, where N refers to the number of samples in that slice. Similarly this slice will be evaluated for validity and run convolutionally and causally across the waveform. What Slice length is optimal is still to be determined. We have defaulted to values of 32 and 64 samples per slice.

IV. Data and Method

A. Dataset

For the purpose of this experiment we trained on the NSynth dataset [18] which consists of 305,979 4 second long .wav files. Since our proposed architecture only accepts 1 second long audio files sampled at 16384 samples per second, the data pre-processing step involved resampling the n-synth dataset at a samplerate of 16384 and then splitting each training example into four equal chunks of 1 second. Those chunks were then indexed and saved to disk as .npy files, as that made it much faster to load individual batches (we used a batch size of 32 as that was the batch size used in the original wavenet paper) with a data generator, since it was not possible to load the

entire dataset into memory. This gave us an extensive dataset of 1,156,820 (289,205 x 4) training examples. This also left over a more than sufficient number of evaluation and testing samples.

Since we need input and output training pairs, we also had to generate the target halves of the training examples. The effect of choice for that was a algorithmic implementation of a low pass filter, as follows:

```
import numpy as np

def lowPassFilter(filter_cutoff, b, signal):

    N = int(np.ceil((4 / b)))

    if not N % 2:
        N += 1

    n = np.arange(N)

    sinc_func = np.sinc(2 * filter_cutoff
                * (n - (N - 1) / 2.))

    window = 0.42 - 0.5 * np.cos(2 * np.pi
                    * n / (N - 1)) + 0.08
                    * np.cos(
                    4 * np.pi * n / (N - 1))

    sinc_func = sinc_func * window

    sinc_func = sinc_func / np.sum(sinc_func)

    return np.convolve(signal, sinc_func)
```

B. GAN training

We followed the standard procedure for training adversarial networks as described in the original GAN paper [6], the discriminator is trained on real data, then on fake data generated with the generator, then we perform an update step on the combined model while the discriminator is frozen. We use the Adam optimizer for both discriminator and generator, but with different learning rates of 0.0004 and 0.001 for generator and discriminator respectively, this choice of learning rates can be justified with the smaller architecture of the discriminator, such that one does not "over-power" the other. The next section V shows generated waveformes after 10 training epochs.

V. Results

We found it best to inspect the generated waveforms visually by plotting them, and generating spectrograms as well as periodograms. This allows for a preliminary subjective evaluation of the viability of the proposed architecture. Two musical notes were chosen at random from the NSynth test set [18] one of which is a C#5 note played on a reed flute with a velocity of 0.75. And the other note is a D6 played on an acoustic guitar also with a velocity of 0.75. It is important to state that these exact two notes do not belong to the training set and haven't been seen by the network during training. Figure 4 and figure 6 show the results of running the two notes through our network respectively. A subjective glance at the waveform plots, reveals that it is capable at modeling the target signals decently. But zooming in on certain spots of the generated waveform, we notice that the generated samples do not perfectly align with the samples of the target signal.

We also found it beneficial to inspect spectrograms and periodograms of the generated signals. Figure 5 and figure 7 show the spectrograms and periodograms of the reed and guitar note respectively. In contrast to the spectrograms, comparing the welch periodograms shows that the generated signal has a very different and undesirable spectral density. It could merely be that the model has not fully converged to an optimum yet and still does not fully understand how to model signals similar to the desired target. Equivalently, it could be that the learned loss function does not aim at reducing this difference.

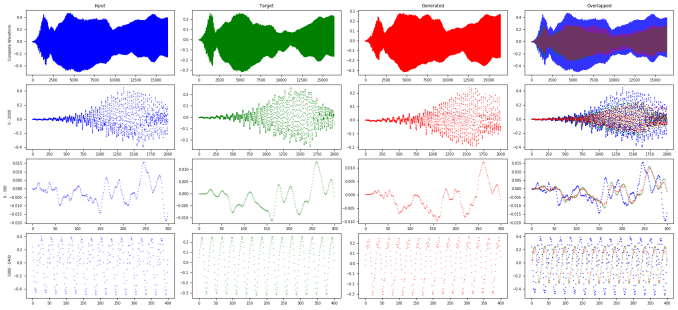

Fig. 4. A C#5 note played on an acoustic Reed instrument. The figure shows several plots: In blue is the original signal, green represents the low-pass filtered signal, red shows the generated waveform from our model and lastly there is also a fourth figure that shows all of them overlapped for comparison. The overall shape of the generated waveform comes very close to the target signal, but it exhibits an additional "squiggliness" when we zoom in on smaller sample ranges. Highly periodic signals do not seem to be very difficult for it to model.

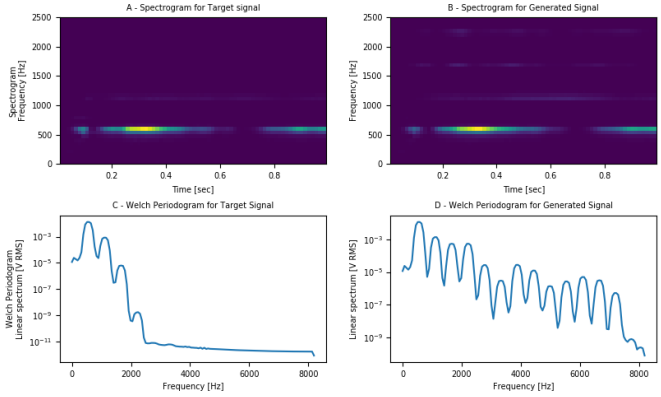

Fig. 5. Spectrograms and periodograms of target low-pass filtered reed note as well as the generated one. Notice the faint white bands in B, these represent some accentuated noise. Comparing the periodograms C and D reveals that the quality of the generated signal might not be as good as the spectrogram promises. The periodogram D shows that the generated signal has a very different spectral density compared to the target signal.

VI. Conclusion

We find that the proposed architecture is promising, but still requires several improvements. It can not yet be said if is capable of learning more complicated target functions that

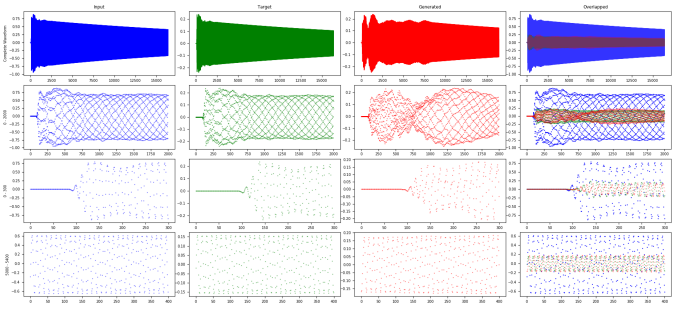

Fig. 6. A D6 note played on an acoustic guitar. Again, the overall shape as well as the periodic patterns of the target waveform are captured well, except for an initial indent.

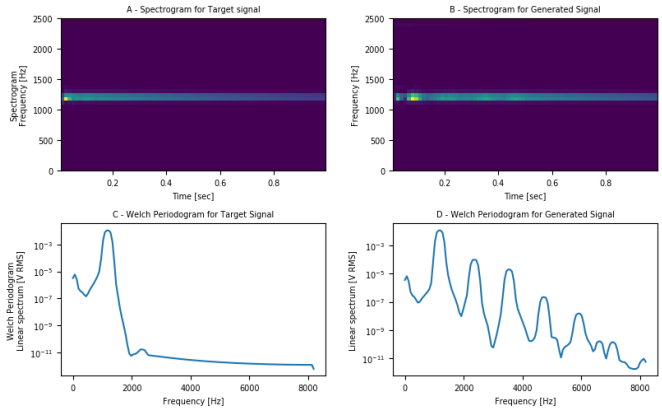

Fig. 7. Spectrograms and periodograms of target low-pass filtered acoustic guitar note as well as the generated one. Similarly to figure 5 the spectrograms reveal that the generated signal has a similar frequency content as the target signal with slightly boosted frequencies. Whereas the periodogram reveals that the generated signal a much higher spectral density for higher frequencies..

are more complicated than low-pass filters, and more test are required to determine that. Future improvements will involve different types of loss functions, especially ones that are based in the frequency domain rather than being applied to the raw audio waveform as phase does not seem too important for our purposes. Additionally, recurrent Unet models could be a promising starting point for applying our model in real-time settings.

References

[1] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *arxiv*, 2016.

[2] U. Zolzer, X. Amatrian, D. Arfib, J. Bonada, G. D. Poli, P. Dutilleux, G. Evangelista, F. Keiler, A. Loscos, D. Rocchesso, M. Sandler, X. Serra, and T. Todoroff, *DAFX: Digital Audio Effects*. England: John Wiley & Sons, Ltd., 2002.

[3] S. Pascual, A. Bonafonte, and J. Serrà, "Segan: Speech enhancement generative adversarial network," in *INTERSPEECH*, 2017.

[4] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *ArXiv*, vol. abs/1411.1784, 2014.

[5] 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.

[6] 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 27* (Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, eds.), pp. 2672–2680, Curran Associates, Inc., 2014.

[7] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks : the official journal of the International Neural Network Society*, vol. 61, pp. 85–117, 2015.

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[9] O. Ronneberger, P.Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, vol. 9351 of *LNCS*, pp. 234–241, Springer, 2015. (available on arXiv:1505.04597 [cs.CV]).

[10] C. Li and M. Wand, "Precomputed real-time texture synthesis with markovian generative adversarial networks," *ArXiv*, vol. abs/1604.04382, 2016.

[11] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1," ch. Learning Internal Representations by Error Propagation, pp. 318–362, Cambridge, MA, USA: MIT Press, 1986.

[12] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, pp. 1735–1780, Nov. 1997.

[13] A. van den Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu, "Conditional image generation with pixelcnn decoders," in *NIPS*, 2016.

[14] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *CoRR*, vol. abs/1511.07122, 2015.

[15] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *ArXiv*, vol. abs/1502.03167, 2015.

[16] J. Chorowski, R. J. Weiss, S. Bengio, and A. van den Oord, "Unsupervised speech representation learning using wavenet autoencoders," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, pp. 2041–2053, 2019.

[17] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, pp. 1929–1958, Jan. 2014.

[18] J. Engel, C. Resnick, A. Roberts, S. Dieleman, D. Eck, K. Simonyan, and M. Norouzi, "Neural audio synthesis of musical notes with wavenet autoencoders," 2017.