236781 – Final project

(TODO) Experimenting with DDSP, by Tal Skverer and Amit Zukier

# Abstract

Summarize your work. Briefly introduce the problem, the methods and state the key results.

(TODO – provide a summary of the DDSP project and the process of deciding how to play with it, along with what we decided to implement and our experiments with it)

# Intro

Review the papers relevant to your project. Explain the problem domain, existing approaches and the specific contribution of the relevant paper(s). Also detail the drawbacks which you plan to address. If it’s a custom project, explain your specific motivation and goals. Cite any other work as needed.

Differential Digital Signal Processing (or DDSP[[1]](#footnote-1) for short), is a library which enables the addition of signal processing elements into TensorFlow[[2]](#footnote-2), a well-known library with modern automatic differentiation.

This work was done to fill in the gap in neural networks training for audio generation. While most generative models (such as WaveGAN[[3]](#footnote-3), SING[[4]](#footnote-4), MCNN[[5]](#footnote-5), WaveNet[[6]](#footnote-6), GANSynth[[7]](#footnote-7) and others) directly generate samples in either the time or frequency domains (and sometimes both of them), DDSP integrates classic signal processing elements (synthesizers and effects), to improve neural networks’ approximation by using the strong structural priors of these tools, which promotes generalization. Specifically, unlike the works mentioned before, using said DSP elements might successfully incorporate their ability to convey audio, as they align with the data domain.

This is, as explained in the paper, because these elements exploit the periodic structure of resonating, similarly to how the human ear has evolved, unlike other audio synthesis models.

For example, generative models such as WaveGAN generate waveforms directly. Since audio usually includes many frequencies, the model must generate aligned waveforms, included with every filter applied to them, which is generally very challenging. Using a harmonic oscillator (called an additive synthesizer) in a neural network eliminates this issue by automatically outputting a signal several sinusoidal components at harmonic frequencies (integer multiples of a fundamental frequency).

More issues covered in the original paper are Fourier-based models such as GANSynth which suffer from spectral leakage problem, and autoregressive waveform models such as WaveNet which bypass all aforementioned issues by generating the waveform a single sample at a time. However, this causes them to require larger networks to learn this complex model, and exposess them to bias during generation.

The DDSP library[[8]](#footnote-8) currently features 6 interpretable DSP elements implemented as TensorFlow layers: 3 synthesizers and 3 effects:

* Additive synthesizer (a harmonic oscillator)
* Filtered noise (“subtractive” synthesizer)
* Wavetable (interpolative lookup from small chunks of waveforms)
* Reverberation
* FIR Filter (linear time-varying finite impulse response)
* ModDelay (variable length delay lines)

(TODO – Fill either

1. An explanation of our addition to the library, why we selected it and how it integrates with the library
2. An explanation of the demo the DDSP guys held, along with our idea of implementing something fun with it.

)

# Methods

If implementing an existing paper, explain the original approach as well as your ideas for modifications, additions or improvements to the algorithm/task/domain etc., as relevant. Otherwise, provide a detailed explanation of your approach. In both cases, explain the empirical and/or theoretical motivation for what you are doing. Finally, describe the data you will be using for evaluation.

(TODO – For all (or maybe just a few) implemented element, explain its implementation, mathematically, while also adding a paragraph of how it makes it differentiable. Perhaps add the way it was trained and used in the paper’s demo)

(TODO – Add a detailed overview of our idea, including for the motivation of why it would work and how it integrates with the paper)

(TODO – Add a paragraph or two on the data we used, how we acquired it and how it was used)

1. Based on [DDSP ICLR Paper](https://openreview.net/pdf?id=B1x1ma4tDr). [↑](#footnote-ref-1)
2. [TensorFlow](https://www.tensorflow.org/overview), a public tool for machine learning, which supports [auto-differentiation using computation graph](https://deepnotes.io/tensorflow) [↑](#footnote-ref-2)
3. Chris Donahue, Julian McAuley, and Miller Puckette. [*Adversarial Audio Synthesis*](https://openreview.net/pdf?id=ByMVTsR5KQ) [↑](#footnote-ref-3)
4. Alexandre Defossez, Neil Zeghidour, Nicolas Usunier, Leon Bottou, and Francis Bach. [*SING: Symbol-to-Instrument Neural Generator*](http://papers.nips.cc/paper/8118-sing-symbol-to-instrument-neural-generator.pdf) [↑](#footnote-ref-4)
5. S. O. Arik, H. Jun, and G. Diamos. [*Single-Image Crowd Counting via Multi-Column Convolutional Neural Network*](https://zpascal.net/cvpr2016/Zhang_Single-Image_Crowd_Counting_CVPR_2016_paper.pdf) [↑](#footnote-ref-5)
6. Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, Koray Kavukcuoglu. [*WaveNet: A generative model for raw audio*](https://deepmind.com/blog/article/wavenet-generative-model-raw-audio) [↑](#footnote-ref-6)
7. [Jesse Engel](https://openreview.net/profile?email=jesseengel%40google.com), [Kumar Krishna Agrawal](https://openreview.net/profile?email=kumarkagrawal%40gmail.com), [Shuo Chen](https://openreview.net/profile?email=chenshuo%40google.com" \o "), [Ishaan Gulrajani](https://openreview.net/profile?email=igul222%40gmail.com), [Chris Donahue](https://openreview.net/profile?email=christopherdonahue%40gmail.com), [Adam Roberts](https://openreview.net/profile?email=adarob%40google.com). [*GANSynth: Adversarial Neural Audio Synthesis*](https://openreview.net/pdf?id=H1xQVn09FX) [↑](#footnote-ref-7)
8. Code available in [GitHub](https://github.com/magenta/ddsp), also a direct URL to the elements’ implementation: [synthesizers](https://github.com/magenta/ddsp/blob/master/ddsp/synths.py) and [effects](https://github.com/magenta/ddsp/blob/master/ddsp/effects.py). [↑](#footnote-ref-8)