NSynth: Neural Audio Synthesis

Apr 6, 2017 • NSynth

One of the goals of Magenta is to use machine learning to develop new avenues of human expression. And so today we are proud to announce **NSynth** (Neural Synthesizer), a novel approach to music synthesis designed to aid the creative process.

Unlike a traditional synthesizer which generates audio from hand-designed components like oscillators and wavetables, NSynth uses deep neural networks to generate sounds at the level of individual samples. Learning directly from data, NSynth provides artists with intuitive control over timbre and dynamics and the ability to explore new sounds that would be difficult or impossible to produce with a hand-tuned synthesizer.

The acoustic qualities of the learned instrument depend on both the model used and the available training data, so we are delighted to release improvements to both:

- A dataset of musical notes an order of magnitude larger than other publicly available corpora.
- A novel WaveNet-style autoencoder model that learns codes that meaningfully represent the space of instrument sounds.

A full description of the dataset and the algorithm can be found in our arXiv paper.

The NSynth Dataset

We wanted to develop a creative tool for musicians and also provide a new challenge for the machine learning community to galvanize research in generative models for music. To satisfy both of these objectives, we built the NSynth dataset, a large collection of annotated musical notes sampled from individual instruments across a range of pitches and velocities. With ~300k notes from ~1000 instruments, it is an order of magnitude larger than comparable public datasets. You can download it here.

A motivation behind the NSynth dataset is that it lets us explicitly factorize the generation of music into notes and other musical qualities. We could further factorize those qualities, but for simplicity we don't and get the following:

$$P(audio) = P(audio \mid note)P(note)$$

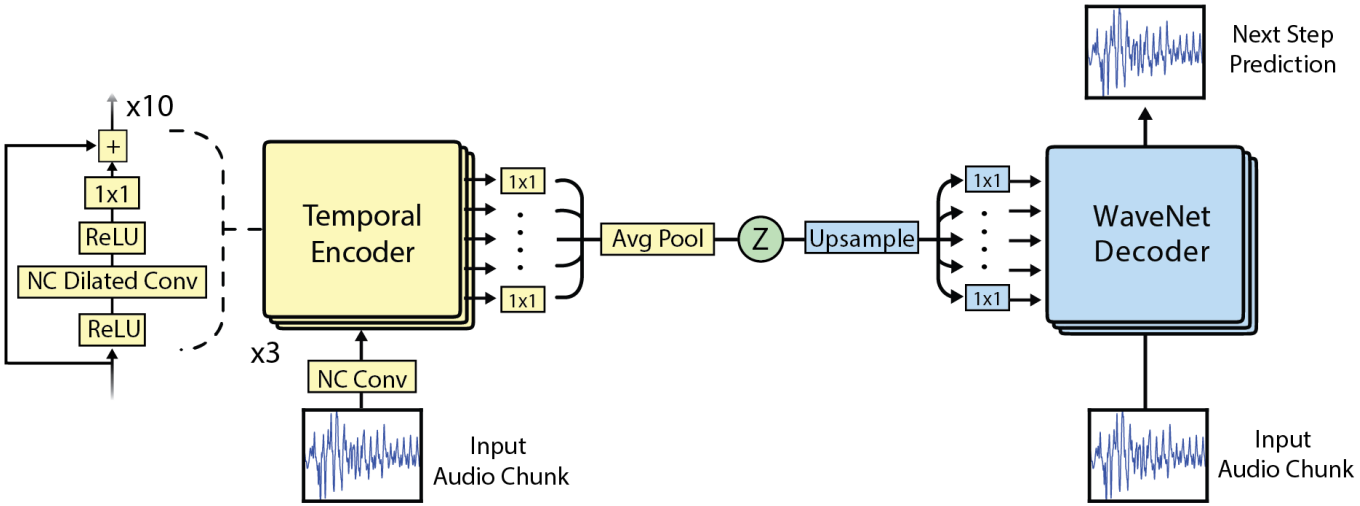
The goal is to model $P(audio \mid note)$ (known as timbre) and it is assumed that $P(note)$ comes from a higher-level "language model" of music, such as the note sequence RNNs we've previously described. While not perfect, this factorization is grounded in how instruments work and is surprisingly effective. Indeed, much modern music production employs such a factorization, using MIDI for note sequences and software synthesizers for timbre. Of course, this works better for some instruments (e.g., piano and electronic synthesizer) than for others (e.g., guitar and saxophone) where note-to-note timbre dependencies are more pronounced.

The NSynth dataset was inspired by image recognition datasets that have been core to recent progress in deep learning. Similar to how many image datasets focus on a single object per example, the NSynth dataset hones in on single notes. We encourage the broader community to use it as a benchmark and entry point into audio machine learning. We hope that this serves as a building block for future datasets and envision a high-quality multi-note dataset for tasks like generation and transcription that involve learning complex language-like dependencies.

Learning Temporal Embeddings

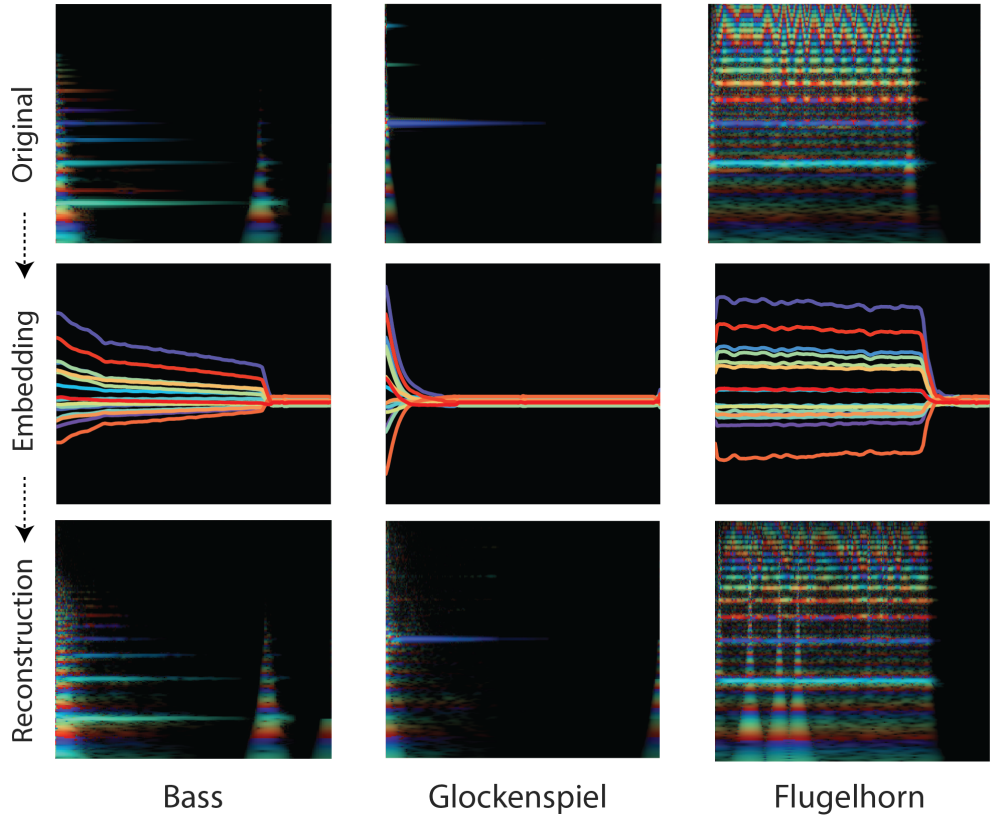
WaveNet is an expressive model for temporal sequences such as speech and music. As a deep autoregressive network of dilated convolutions, it models sound one sample at a time, similar to a nonlinear infinite impulse response filter. Since the context of this filter is currently limited to several thousand samples (about half a second), long-term structure requires a guiding external signal. Prior work demonstrated this in the case of text-to-speech and used previously learned linguistic embeddings to create impressive results.

In this work, we removed the need for conditioning on external features by employing a WaveNet-style autoencoder to learn its own temporal embeddings.



The temporal encoder looks very much like a WaveNet and has the same dilation block structure. However, its convolutions are not causal, so it sees the entire context of the input chunk. After thirty layers of computation, there is then a final average pooling to create a temporal embedding of 16 dimensions for every 512 samples. Consequently, the embedding can be thought of as a 32x compression of the original data.

We condition the vanilla WaveNet decoder with this embedding by upsampling it to the original time resolution, applying a 1x1 convolution, and finally adding this result as a bias to each of the decoder's thirty layers. Note that this conditioning is not external as it's learned by the model. Since the embeddings bias the autoregressive system, we can imagine it acting as a driving function for a nonlinear oscillator. This interpretation is corroborated by the fact that the magnitude contours of the embeddings mimic those of the audio itself.



Bass Original

0:04 / 0:04

Bass WaveNet

0:00 / 0:04

Glockenspiel Original

Glockenspiel WaveNet

Flugelhorn Original

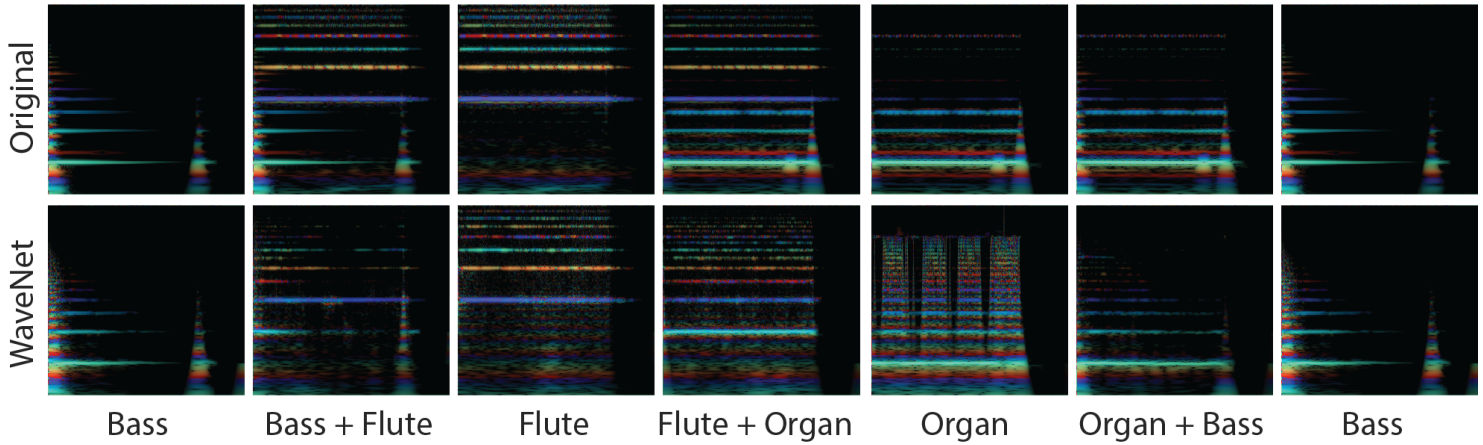
Flugelhorn WaveNet

“Rainbowgrams” of audio and reconstructions for three different instruments. These are CQT spectrograms with magnitude represented by intensity and instantaneous frequency by color. Frequency is on the vertical axis and time is on the horizontal axis. For the embeddings, the different colors represent the 16 different dimensions at 125 timesteps (32ms per step). There is a slight built-in distortion due to the compression of the 8-bit mu-law encoding. It is a minor effect for many samples, but is more pronounced for lower frequencies.

A Latent Space of Timbre and Dynamics

We’ll soon be releasing an instrument as an interactive demonstration, but in the meantime here’s an illuminating example of what you can do with this technology.

Below, there are two columns of audio clips matching the two rows of rainbowgrams. The top row of rainbowgrams corresponds to the left column of audio and is the result of evenly interpolating across instruments by adding the signals. The bottom row of rainbowgrams corresponds to the right column of audio and is the result of using NSynth to linearly interpolate in the embedding space. Try playing the clips starting with the Bass, then the BassFlute, and so on. What you hear in the left column is the linear addition of the signals in the audio output space. As expected, it sounds like the two instruments being played at the same time. In the right column, however, the new notes combine semantic aspects of the two original sounds to create a unique sound that is still musical.



Bass Original
0:00 / 0:04

Bass+Flute Original
0:00 / 0:04

Flute Original
0:00 / 0:04

Flute+Organ Original
0:00 / 0:04

Organ Original
0:00 / 0:04

Organ+Bass Original
0:00 / 0:04

Bass Original
0:00 / 0:04

Bass WaveNet
0:00 / 0:04

Bass+Flute WaveNet
0:00 / 0:04

Flute WaveNet
0:00 / 0:04

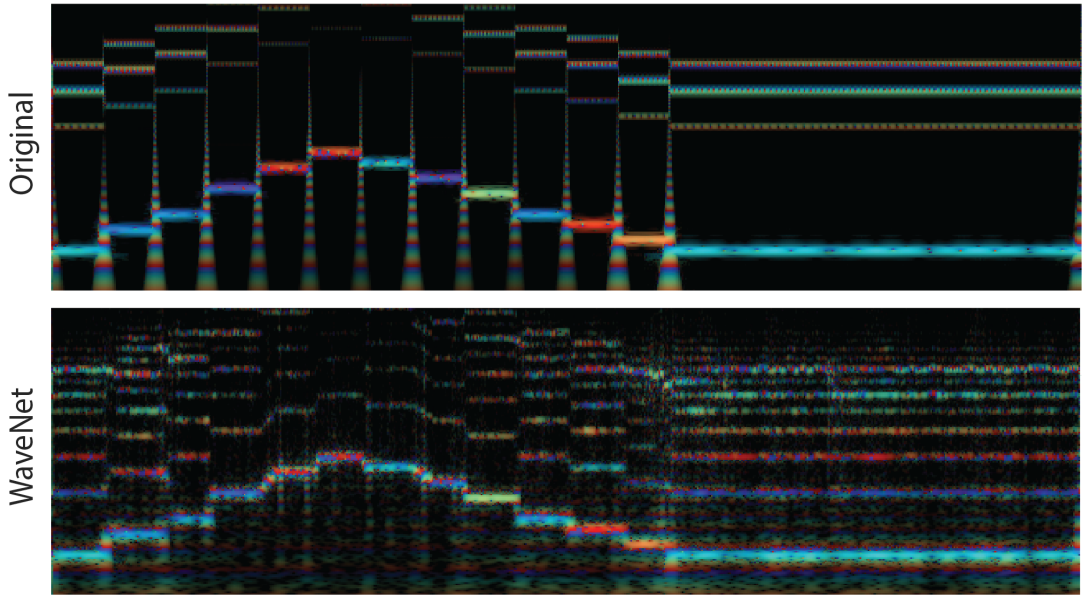
Flute+Organ WaveNet
0:00 / 0:04

Organ WaveNet
0:00 / 0:04

Organ+Bass WaveNet
0:00 / 0:04

Bass WaveNet
0:00 / 0:04

Further, the learned embeddings only capture a local context, much like a spectrogram, which enables them to generalize in time. Despite having only trained on short single notes, the model can successfully reconstruct both a whole series of notes as well as notes played for longer than three seconds.



Scale Original

0:00 / 0:10

Scale WaveNet

0:00 / 0:10

While the WaveNet autoencoder adds more harmonics to the original timbre, it follows the fundamental frequency up and down two octaves. The fact that it has never seen a transition between two notes is clear as its best approximation is to just smoothly glissando between them.

Release++

Besides the music examples and the dataset, we are also releasing the code for both the WaveNet autoencoder powering NSynth as well as our best baseline spectral autoencoder model. In addition, we are releasing the trained weights as a TensorFlow checkpoint and a script to save embeddings from your own WAV files. You can find all the code at our repository and the checkpoint tarball can be downloaded here.

Part of the goal of Magenta is to close the loop between artistic creativity and machine learning, so we have also released playable instruments for you to make your own music with these techonologies.

A Group Effort

This work was a collaboration between the Google Brain team and DeepMind. The main contributors were Jesse Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Karen Simonyan, Mohammad Norouzi, and Doug Eck. We especially want to acknowledge Sander and Karen for their key algorithmic contributions and Adam for handling the brunt of the dataset.

We also want to acknowledge Hans Bernhard, Colin Raffel, Sageev Oore, Yotam Mann, Ron Weiss, and Rif Saurus. Your help was much appreciated.