DeepSynth: Synthesizing A Musical Instrument With Video

Tal Stramer
Stanford University, Department of Computer Science

Background

- Creating an electronic instrument requires 1) manually creating tables to map notes and desired characteristics to audio 2) physically building a representation of the instrument
- In virtual reality, can we automate the process? Create a playable virtual instrument based on examples of a real instrument being played.
- Recent digital synthesizers based on neural network architectures map individual notes and desired characteristics to audio but ignore the physical interaction with the original instrument

Problem Statement

- **Problem**: Given a silent video of a musical instrument being played, predict the audio outputted by the musical instrument
- **Approach**: CNN to encode video frames and CNN based autoregressive model to generate audio conditioned on video
- **Evaluation metrics**
  - Train: Softmax loss on quantized version of audio waveform with 256 buckets. Real audio frames used to generate next
  - Test: Same loss. Generated audio frames used to generate next

Dataset

- Dataset generated based on 20 classical songs in MIDI format fed into a piano simulator
- Audio waveform generated based on a digital synthesizer of a grand piano sampled at 4 kHz
- Audio split up into 5 second segments with 2 second overlapping windows
- Sparse representation of video based on video frames at every note press
- Data split into 60% training, 20% validation, and 20% testing.

Figure 1. Sample video frame

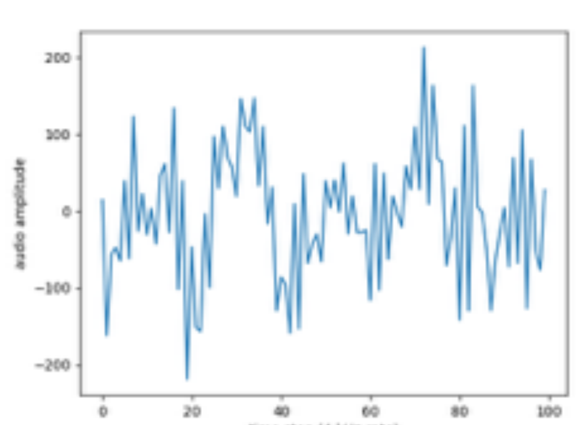

Figure 2. Sample audio waveform.

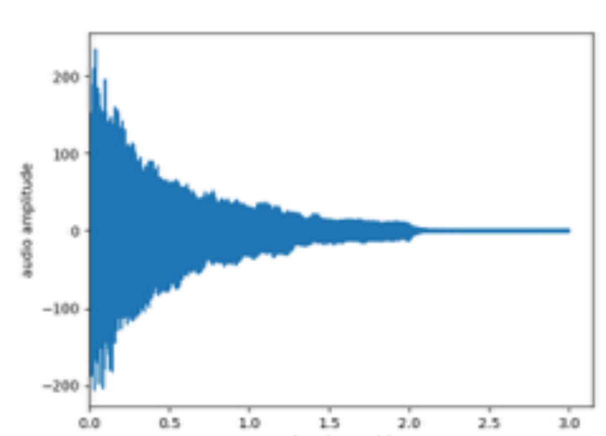

Figure 3. Single note attack decay

Methods and Materials

- **Frame encoder**: Convert video frames to feature vectors using VGG-like CNN
- **Audio generator**:
  - Model audio generation by a stack of causal convolution layers
  - Predict audio waveform at each time step based on current video frame features and previous video features and audio frames.
  - Use exponentially increasing dilation factor at each layer to increase receptive field without decreasing resolution
  - Use residual and skip connections to speed up training and allow for more layers
  - Apply dropout to audio inputs to increase reliance on video frames during training
  - Use learning rate annealing to speed up training
- **Cost function**: Cross-entropy loss over training examples

Experiments

- Experiment with various parameters
  - Dropout percentage: 50% or 75% of the audio
  - Number of layers in audio generation network: 7 or 10 (affects receptive field and expressive power)
  - Number of features per layer of audio network: 128, 256, or 512
- Run 100 epochs per experiment
- Evaluate validation loss after each epoch on 100 examples from validation set. Choose model from epoch with lowest loss.

Dropout percent	Number of layers	Features per layer	Train Loss	Validation Loss
0.5	7	128	1.54	2.91
0.5	7	256	1.42	2.81
0.5	10	512	1.2	2.62
0.75	7	128	1.65	2.73
0.75	7	256	1.60	2.53
0.75	**10**	**512**	**1.45**	**2.41**

Table 1. Experiment results.

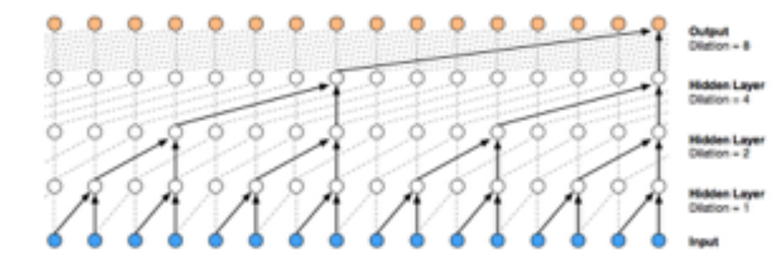

Figure 4. Illustration of audio generation model from Oord et al [1].

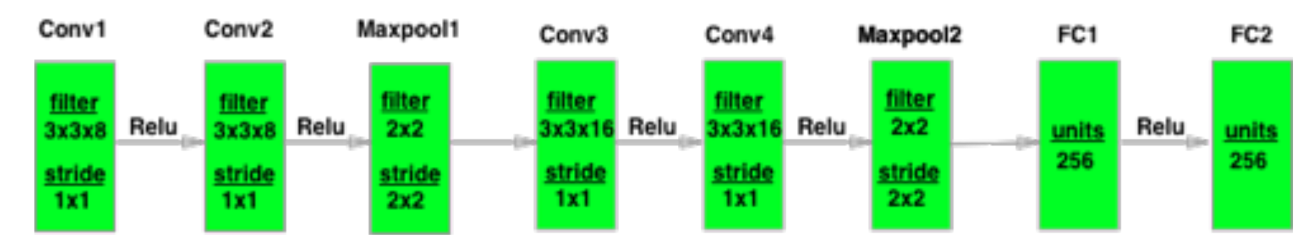

Figure 5. Illustration of frame encoder CNN layers

Discussion

- Ability to generate correct audio at test time highly dependent on using dropout on audio input during training - likely because otherwise it will rely mostly on previous audio and ignore video
- Generation very slow at test time due to having to generate audio sequentially - a non-generative network that predicts output only based on video frames would be much faster.
- Would be interesting to try a regression-based cost function, as distance between predicted and actual audio relevant to generation quality and is not captured directly by softmax loss.

Conclusions

- Relatively simple neural architecture can achieve high performance on generating raw audio from video
- No manual feature engineering or data labeling is a huge benefit.
- Future work:
  - Use real videos instead of simulated ones, like piano videos from the Youtube 8M dataset
  - Build a virtual representation of the instrument from video examples and combine with audio model
  - Build a more efficient generator by using low-level optimizations
  - Analyze how model behaves on different types of music

References:
[1] Van den Oord, Aron, et al. "Wavenet: A generative model for raw audio." CoRR abs/1609.03499 (2016).