Using Machine Learning to Program a Synthesizer

Aaron Ferrer, Dhireesha Kudithipudi
The University of Texas at San Antonio, San Antonio TX, 78249

Name: *Aaron Ferrer*
Status: *Junior*
Department: *College of Engineering*
Area of Study: *Computer Engineering*
USDA/UTSA Mentor(s): *Dhireesha Kudithipudi*

WeARE Research Area

This research area concerns the applications of artificial intelligence algorithms. In this study, artificial intelligence is used to potentially create tools for musicians.

Motivation or Background

Digital synthesizers are used to generate a wide variety of instrumental sounds. However, the reverse – to use sounds to program synthesizers – is a tedious task for musicians that takes a skilled ear and some luck. Modern synthesizers typically have thousands of presets that can be used as data to train a neural network. Thus, a neural network can, given music/instrument samples, predict synthesizer presets. A previous study on this subject had used a very, very simple synthesizer, so a more complex synthesizer is studied.

Because so much music data is collected, it is also interesting to use machine learning for dimensionality reduction in order to plot these sound samples into a graph, effectively showing how capable the synthesizer is in designing sounds.

Objectives

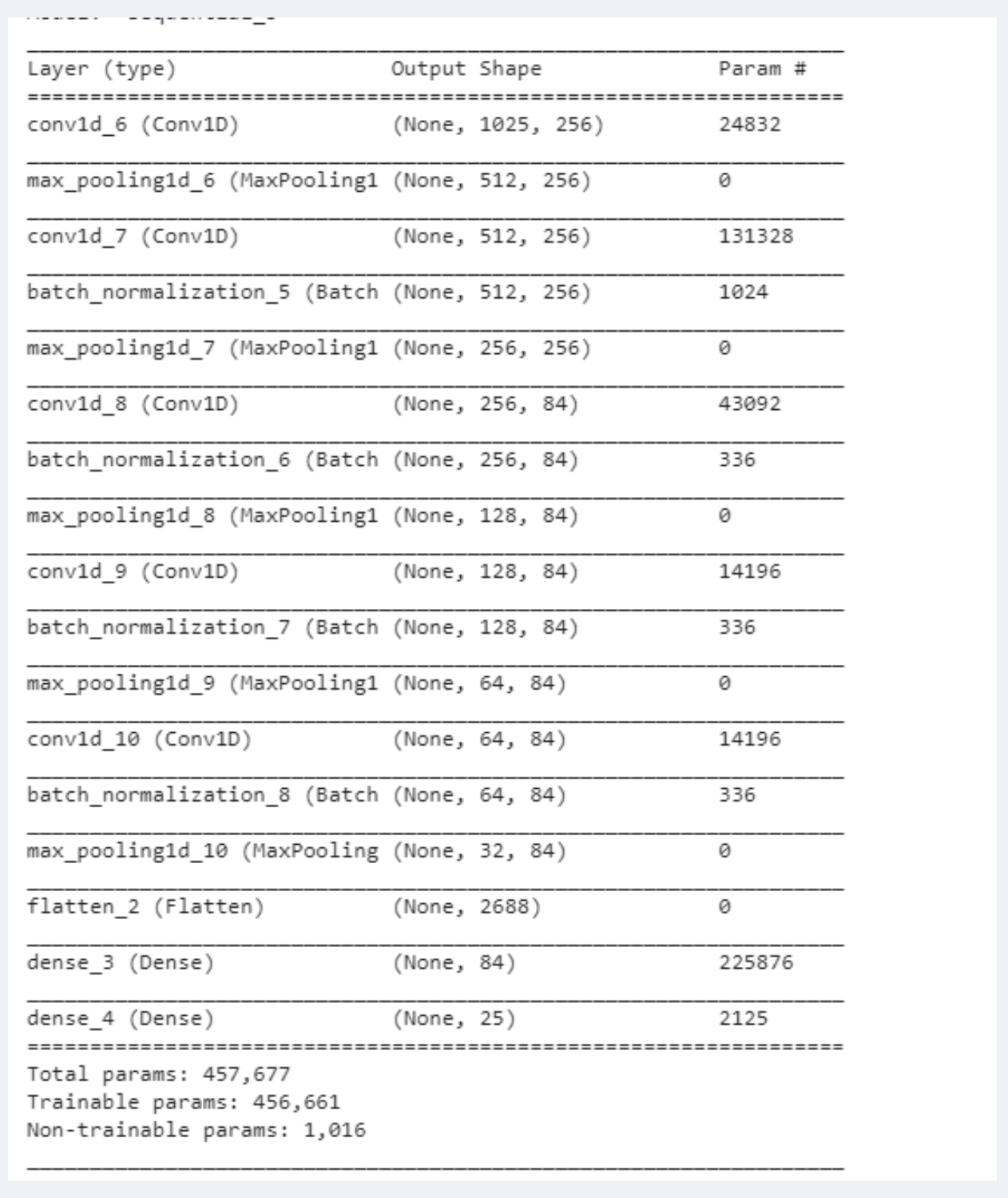
1. Find a suitable synthesizer to collect several sound samples and their synthesizer preset counterparts.
2. Analyze the breadth of these samples using dimensionality reduction techniques.
3. Train a neural network, and assess performance not only statistically, but also by programming the synthesizer with predicted presets.

Methodology

The Sylenth1 synthesizer was chosen as it had several banks sometimes holding 512 presets each. Using *Preset-2-Excel* by Christian Budde, 3 factory banks were extracted, giving 1536 samples to train and test on. With the *FL Studio* digital audio workstation, the *Edison* plugin was used to record 1.048 second samples at 128 kbps, recording a C5 note. Data is then compressed using short term Fourier transform.

2 factory banks were chosen as training samples (1024 samples) and 1 factory bank (512 samples) was chosen for test samples. Furthermore, audio clips from 3 songs (without pre-programmed presets in the synthesizer) were experimented with.

The convolutional neural network architecture is based on a previous study by Harrison Taylor, but a tanh activation is used at the end, shown in Figure 1 [1]. The data is also plotted using Principal Component Analysis (PCA) in order to visualize the extent of sounds generated by the synthesizer.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv1d_6 (Conv1D)            (None, 1025, 256)         24832     
_________________________________________________________________
max_pooling1d_6 (MaxPooling1 (None, 512, 256)          0         
_________________________________________________________________
conv1d_7 (Conv1D)            (None, 512, 256)          131328    
_________________________________________________________________
batch_normalization_5 (Batch (None, 512, 256)          1024      
_________________________________________________________________
max_pooling1d_7 (MaxPooling1 (None, 256, 256)          0         
_________________________________________________________________
conv1d_8 (Conv1D)            (None, 256, 84)           43092     
_________________________________________________________________
batch_normalization_6 (Batch (None, 256, 84)           336       
_________________________________________________________________
max_pooling1d_8 (MaxPooling1 (None, 128, 84)           0         
_________________________________________________________________
conv1d_9 (Conv1D)            (None, 128, 84)           14196     
_________________________________________________________________
batch_normalization_7 (Batch (None, 128, 84)           336       
_________________________________________________________________
max_pooling1d_9 (MaxPooling1 (None, 64, 84)            0         
_________________________________________________________________
conv1d_10 (Conv1D)           (None, 64, 84)            14196     
_________________________________________________________________
batch_normalization_8 (Batch (None, 64, 84)            336       
_________________________________________________________________
max_pooling1d_10 (MaxPooling (None, 32, 84)            0         
_________________________________________________________________
flatten_2 (Flatten)          (None, 2688)              0         
_________________________________________________________________
dense_3 (Dense)              (None, 84)                225876    
_________________________________________________________________
dense_4 (Dense)              (None, 25)                2125      
=================================================================
Total params: 457,677
Trainable params: 456,661
Non-trainable params: 1,016
_________________________________________________________________
```

Fig. 1
Convolutional Neural Network architecture.

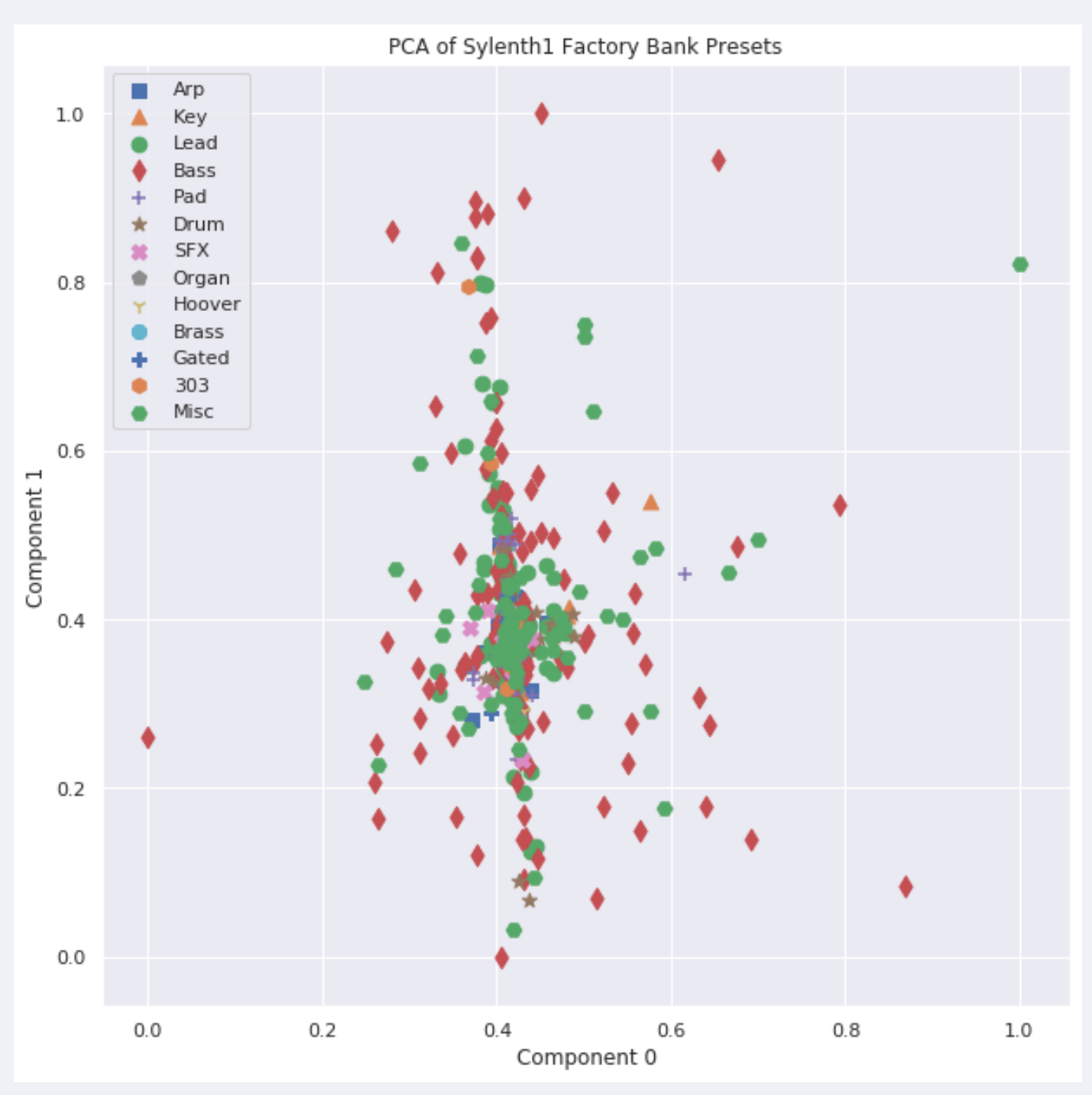


Fig. 2
PCA of test samples. Components are arbitrary.

Results

Training accuracy reached an average of 40.5% (±2.2%) and a loss of 0.0321, showing that accuracy can't improve more as the loss has closely converged to 0.

Average testing accuracy was 33.98% with a loss 0.104.

In analyzing the PCA of this data, the features that capture the most variance are timb re (waveform) and dynamics. Based on this PCA analysis, in order to create a model that ca ptures different kinds of sounds, a different synthesizer and more parameters will have to be used. Even without 100% accuracy, the presets generated by this project are very usable by musicians.

To listen to the results of PCA and to predicted test samples, please scan the QR code listed in Figure 5.

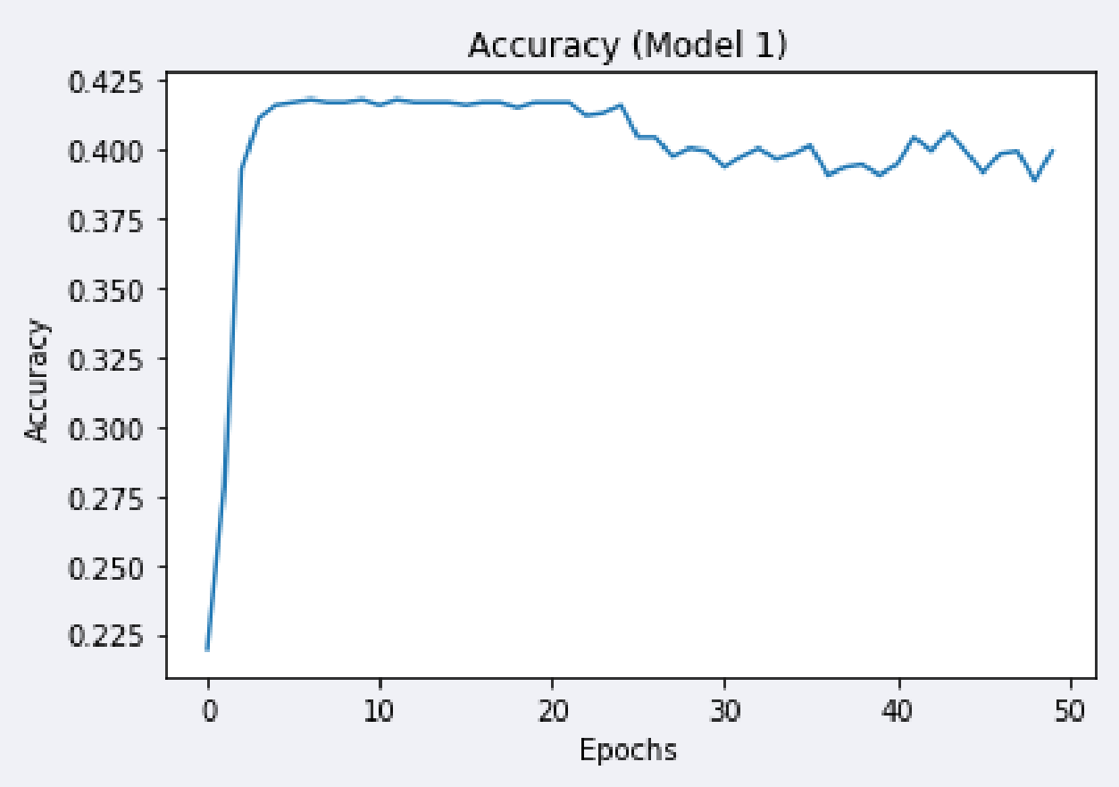


Fig. 3
Training accuracy

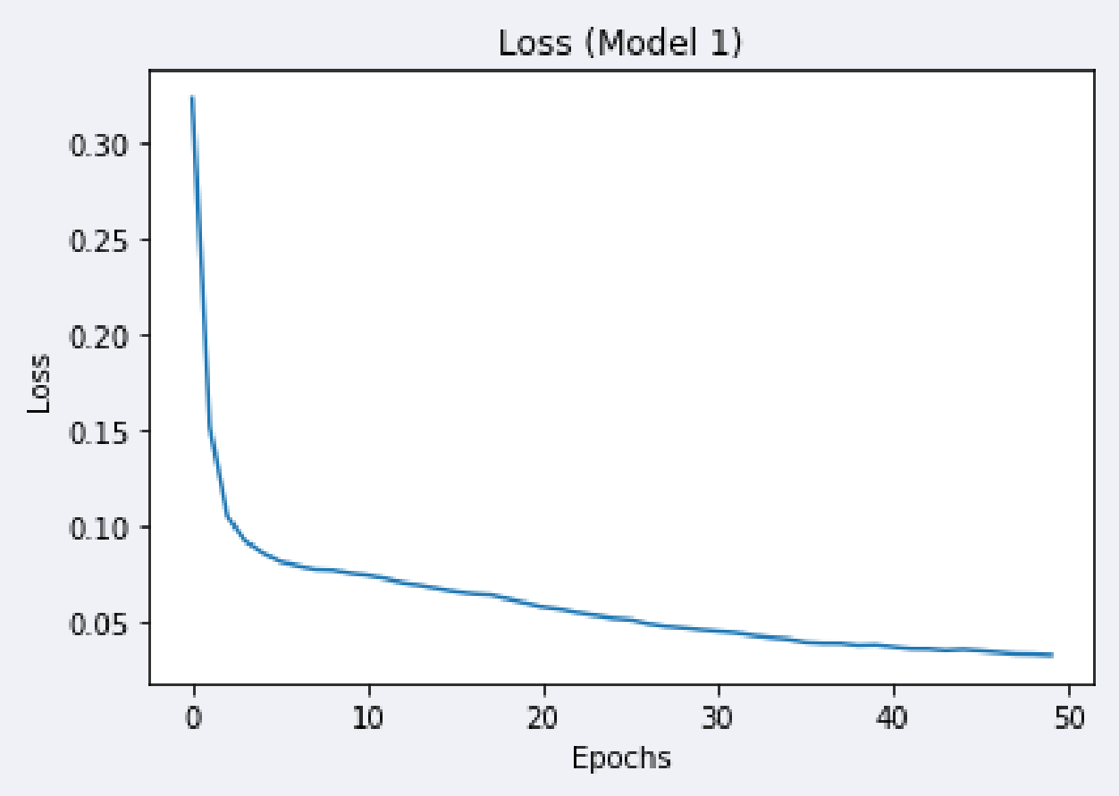


Fig. 4
Training loss.

Fig. 5
PCA and prediction results.

Skills and Experience

Skills included data science (analysis, wrangling, and artificial intelligence) and Python programming. Through this project, I experienced how to develop artificial intelligence to solve a statistical regression problem, building a convolutional neural network from scratch, using signal processing to allow music data to be used as training data, using dimensionality reduction techniques to make graphing high dimensionality data possible, and deploying an artificial intelligence algorithm.

What I Learned

Projects using artificial intelligence won't reach perfect accuracy without big data, but the results, especially in applications where precision isn't required, still provide interesting results. I also learned helpful methods to analyze data, such as dimensionality reduction techniques and signal processing techniques.

Future Plans

This project was proposed to create a completely new synthesizer that uses this prediction mo del to create presets from sounds. This is planned to be implemented in the JUCE framework, which is used to create synthesizers.

Acknowledgments

This work is supported by the USDA National Institute of Food and Agriculture, Interdisciplinary Hands-on Research Traineeship and Extension Experiential Learning in Bioenergy/Natural Resources/Economics/Rural project, U-GREAT (UnderGraduate Research, Education And Training) program (2016-67032-24984).

References

1. Reverse Engineering Audio Synthesiser Sounds. (2018, May). Retrieved March 9, 2020, from https://pats.cs.cf.ac.uk/@archive_file?p=921&n=final&f=1-main.pdf&SIG=f34d8ce813ffce3ec01f9ead87cc09ffeb6e1e6115657ae2aba84ef17105a31d.
2. Fedden, L. (2017, November 21). Comparative Audio Analysis With Wavenet, MFCCs, UMAP, t-SNE and PCA. Retrieved March 13, 2019, from https://medium.com/@LeonFedden/comparative-audio-analysis-with-wavenet-mfccs-umap-t-sne-and-pca-cb8237bfce2f.