Automatic Music Transcription with Convolutional Neural Networks

This project was an exploration into the utility of convolutional neural networks in music transcription.

Specifications:

I trained on a 64-bit PC with GPU. My CPU is an Intel i5-7400 @ 3.40gHz, I have 16GB of DDR4 RAM, and my graphics card is a Nvidia GeForce GTX 1050 Ti.

Model:

Here is the two CNN architectures used in this project. I used a window size of 7 and the number of frequencies in the input after preprocessing totalled 174. The loss used was binary_crossentropy, and I used the ADAM optimizer for both models:

	Baseline	**Secondary**
Input	(window_size, num_freqs)	(window_size, num_freqs)
Reshaping	Reshape(window_size, num_freqs, 1)	Reshape(window_size, num_freqs, 1)
	Conv2D(50x3x25)	Conv2D(32x5x4)
	BatchNormalization	BatchNormalization
	ReLU	ReLU
	Dropout(0.2)	Dropout(0.2)
First Stage CNN	MaxPooling(1x3)	-
	Conv2D(50x3x5)	Conv2D(32x3x5)

	BatchNormalization	BatchNormalization
	ReLU	ReLU
	Dropout(0.2)	Dropout(0.2)
Second Stage CNN	MaxPooling(1x3)	MaxPooling(1x2)
Flattening	Flatten	Flatten
	Dense(1000)	Dense(1000)
	BatchNormalization	BatchNormalization
	ReLU	ReLU
Flattening First Stage	Dropout(0.5)	Dropout(0.5)
	Dense(200)	Dense(200)
	BatchNormalization	BatchNormalization
	ReLU	ReLU
Flattening Second Stage	Dropout(0.5)	Dropout(0.5)
	Dense(88)	Dense(88)
Output	Sigmoid	Sigmoid
	----------	----------
Total Params	2, 515, 438	2, 896, 008

Both models took an estimated ~240s per epoch, and I trained both models for 100 epochs. In total, the training time for both models became 8-10 hours. However, as displayed in graphs afterwards, accuracy did not greatly improve in either model after ~epoch 40.

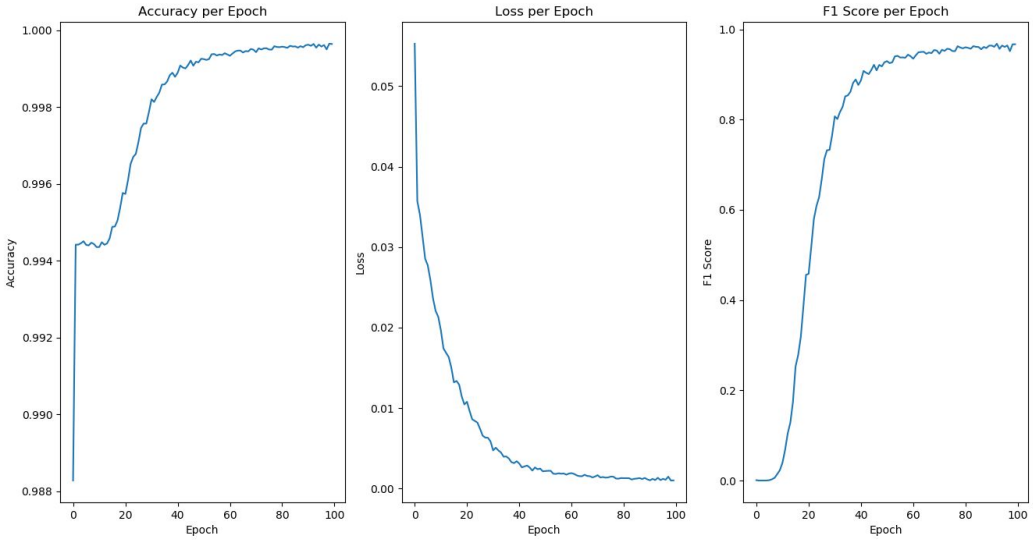
The purpose of the first CNN stage is to determine the onset/offset times of the notes. For this reason the dimensionality is more wider than taller (3x25 in the baseline model). Since the x-axis is my time-axis, a wider filter will result in more features learned on the x-axis.

The purpose of the second CNN stage is to determine the pitch of the notes. For this reason the dimensionality is less wide than the previous CNN (3x5 in the baseline model). Since the y-axis is my frequency-axis, a taller filter will result in more features learned on the y-axis.

The rest of the model is implemented to flatten and output the notes in midi note space (88 keys of the piano).

Performance Graphs:

For Baseline Model:



Pictured: Accuracy on left, Loss in middle, F1 Score on right

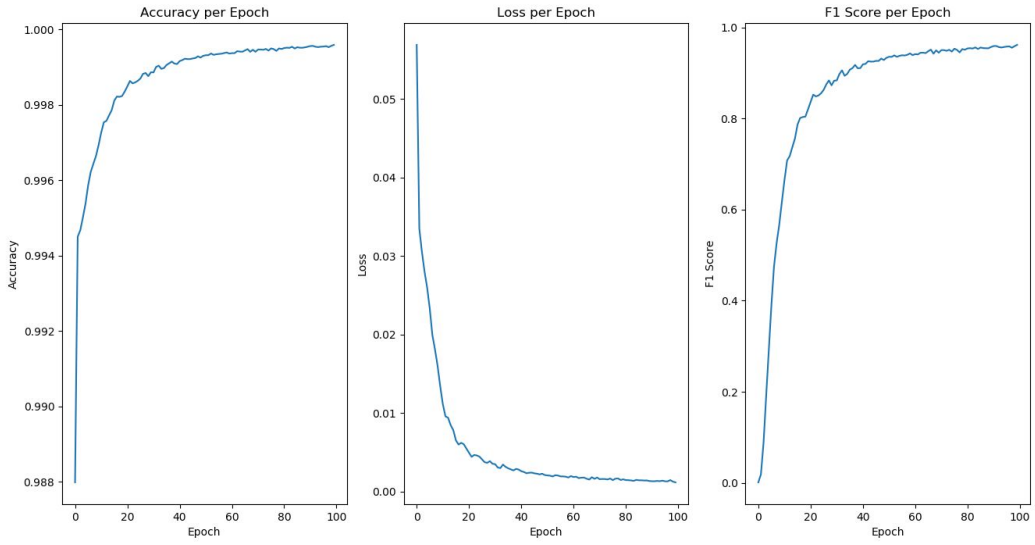
Comments: Accuracy raises very quickly, then strangely pauses before rising again. F1 Score consistently rises. Loss consistently decreases. After ~epoch 40, model training becomes less beneficial.

Final Accuracy: 99.96%

Final Loss: 1.01e-3

Final F1: 9.670e-1

For Secondary Model:



Pictured: Accuracy on left, Loss in middle, F1 Score on right

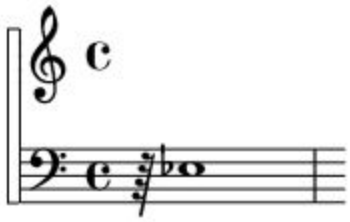
Comments: Accuracy and F1 Score both raise quickly, and loss decreases quickly. After ~epoch 40, model training becomes less beneficial.

Final Accuracy: 99.96%

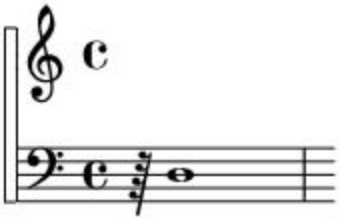
Final Loss: 1.19e-3

Final F1: 9.62e-1

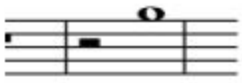
Outputs:



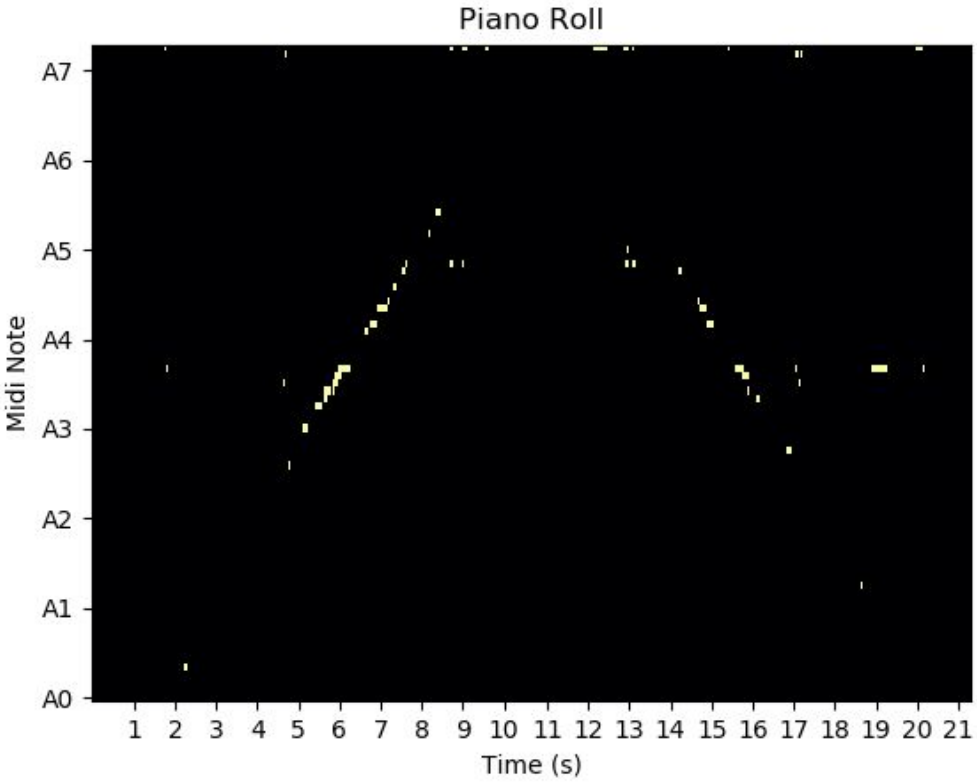
Pictured: The model correctly identifying E-flat 3



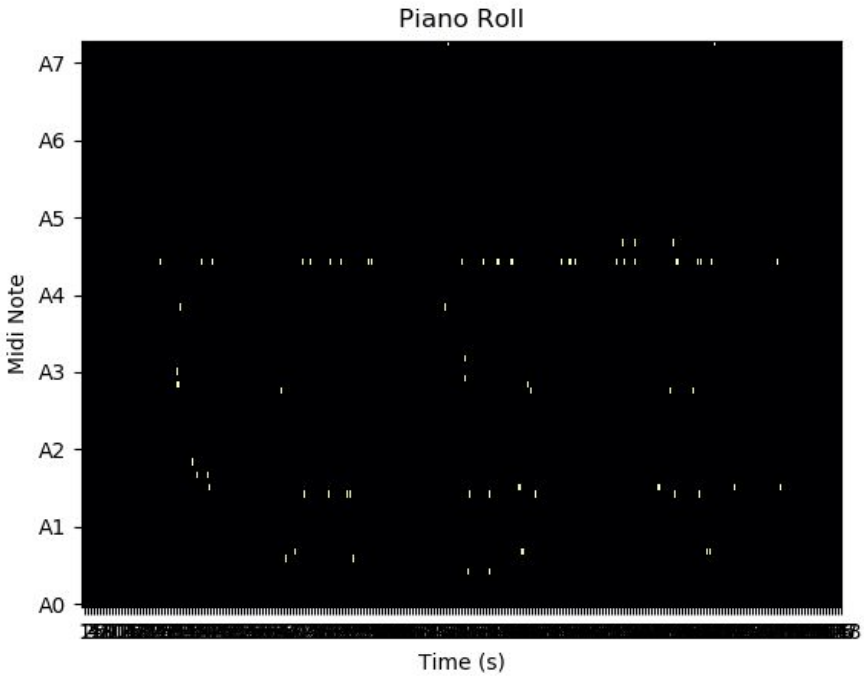
Pictured: The model correctly identifying D3, only one semitone away from E-flat 3



Pictured: The model correctly identifying a G5



Pictured: Semi Successful piano roll transcription of a chromatic scale



Pictured: Poor Transcription of a rather complicated song, Animenz's "Unravel" at https://www.youtube.com/watch?v=sEQf5lcnj_o

Discussion:

The model had some major successes in identifying single notes, and was able to distinguish notes only mere semitones apart. The model also did okay with the chromatic scale and some chords, although it often confused quick jumps in distinct notes as chords.

The model did very poorly with a large song, although it was able to transcribe a semblance of a "left hand" and "right hand" side as seen in the image above.