|  |
| --- |
| **Monophonic Pitch Estimation Using Deep Convolutional Networks** |

**Sean Long Piyush Dahiya Shrey Patel**

College of Arts and Science College of Arts and Science College of Arts and Science

Georgia State University Georgia State University Georgia State University *slong24@student.gsu.edu pdahiya1@student.gsu.edu spatel230@student.gsu.edu*

**Abstract**

In the world of sound recording, our fundamental frequency tool will help aid in speech recognition as well as musical info calculator. Through the process of deep convolutional neural networks and a data-driven algorithm, we were able to develop a model that can testimate monophonic pitches. We evaluated the robustness and accuracy of our model with modifications made along the way due to problems that arose. The upscaling of the training set as well as the addition of dropout layers and batch normalization resulted in major improvements to the models performance.

**1 Introduction**

**1.1 Problem and Goal**

Magenta is a department of Google which has made a staggering amount of progress in the field of data-driven audio synthesis, since its inception in 2016. The “NSynth'' Library - a big data repository of musical notes with varying pitch and timbre dynamics amidst others - coupled with advancements in convolution based autoencoders effectively capturing long-term structure without long-term conditioning have paved way for a plethora of new research in the field. Our project is to estimate pitch via deep networks and its implications in the field of audio processing and neuroscience, specifically by correlating it with auditory pathways and brain activity. The human auditory pathway has many components that control the simplest task like waking up or feeling emotion from noise. “We see regions specialized for deep analysis for isolated sounds (i.e. speaker identification and music perceptions).” [2] By connecting multiple neural networks to simulating the way humans observe sounds, we can identify live sounds using audio databases to train our model. Currently the majority of audio based convolutional networks operate via converting audio into a spectrogram and using preexisting image processing frameworks for analysis. Our approach instead uses audio samples fed into the network as a one dimensional feature vector. This more closely aligns with the human auditory cortex.

**1.2 Proposed Solution**

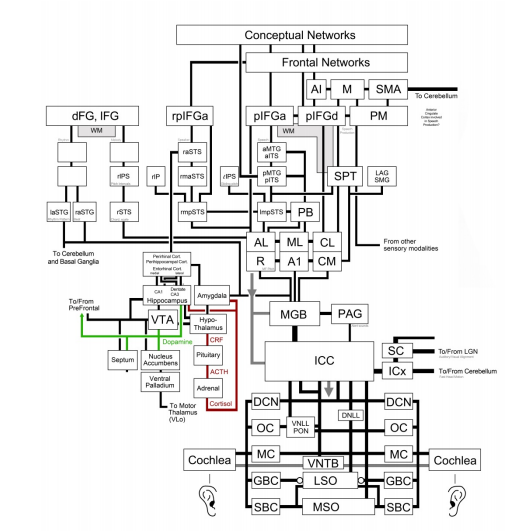
The best way for pitch estimation is to use multiple different deep neural networks and learn which technique would work best. Initially, the goal was to implement a RNN LSTM model, but this caused challenges with the massive datasets such as the one our model was using to learn (70+ GB). Switching to CNN helped speed up training and allowed for quicker and more stable results. Other music retrieval systems like CHORDID and beat detection tend to be better performed by data-driven while a heuristic approach performs better with periodicity. While periodicity is directly related to fundamental frequencies, it did not take into account unknown instruments, noise, and pitch fluctuations[1]. By using past research for speech recognition and instrumental recording software, we were able to derive a Model that is able to identify the pitch of a sound based on quantitative values given by the datasets.

**2 Background**

David Hubel and Torsten Wiesel experimented with the ideas of a mammals brain to hypothesize a model that shows how mammals perceive the world back in the early 1900’s. During the early stages of research they found that time was the variable that caused errors when measuring and comparing, so by plotting the amplitude against frequency allowed for more accurate classification. They also used CNN to train on similarities between variables of digital sound.

In studying humans’ audio corex, a structure was made to replicate all the functions that occur when listening to speech, music, or random sounds that the world is full of. By identifying all the little details that sometimes the brain decides to ignore an input and the importance it puts on other sounds in the forefront. Most components past the ICC contribute to noise cancelation, dialect, and emotional feathers. The human brain’s initial signal processing involves condensing information from the left and right cochlea then combining them. One region processes temporal(phase) differences and the other processes amplitude differences. These two regions then feed into the central interior cochlaeus which does signal normalization before passing the audio data to more specialized regions of the auditory cortex. The interior cochlaeus not only sends data onwards into the audio cortex but it also transmits back to the cochlea and causes it to selectively activate and deactivate the outer cochlea hair cells. These hairs are responsible for the encoding of sound waves signal into electrical signals in the cochlea. This feedback mechanism is analogous to how certain more advanced artificial neural networks feedback processed information to earlier layers to improve performance.

Figure: Diagram of human auditory cortex structure



**2.1 Dataset**

Our data set was obtained from NSynth, which is a part of Google’s Magenta Project. The dataset contains 289,205 training samples and 4096 testing samples in the “.wav” format along with corresponding JSON files with all the file attributes and properties. The two images below are the visual representation of two random sample audio files, and the differences are quite notable.

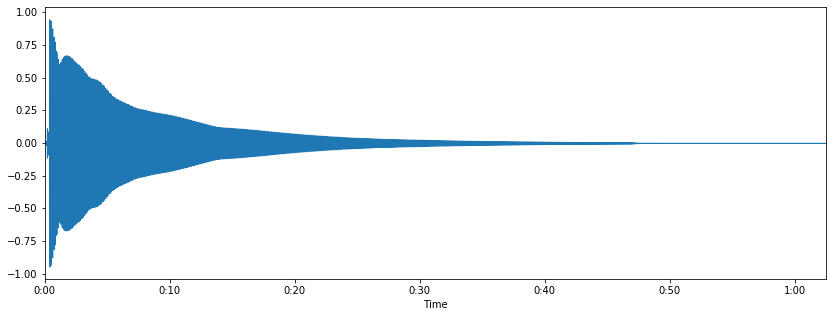
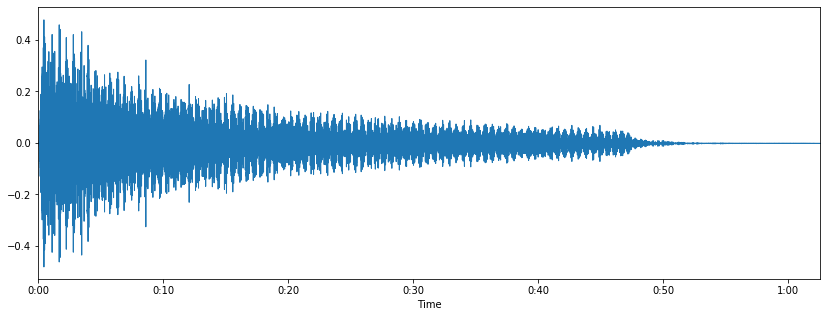
Figure 1: Waveform Representation of sample ‘bass\_synthetic\_068-049-025’

Figure 2: Waveform Representation of sample ‘keyboard\_electronic\_001-021-127’



The audio files were sampled at 16kHz. This means that for each 4 second long file we obtained 64000 values which were converted into an array in Python. The audio files each contained 13 distinct properties from pitch to instrument quality which were stored in a JSON file along with the dataset. For our purposes we were only interested in the “Pitch” attribute, which was isolated as the target label for our samples. Pitch was encoding in the midi tuning standard, which assigns pitch values ranging from 0 to 127. Each pitch value represents one semitone with 21 representing A0 and 108 representing C8. Loading the files yielded an array with dimensions 4096x64000 for testing and 289205x64000 for training, as each audio file has 64000 audio values. Due to computational limitations we spliced and used a 1024 audio value sample from 64000 full samples, yielding the test data as 4096x1024 and training data as 289,205x1024. The pitch values in the midi format range from 0 to 127. Instead of having the pitch values as an one-dimensional array, we used one-hot encoding as often deep learning frameworks require certain standards of data input. The y values yielded an array of 4096x128 and 289,205 respectively. To further ensure the model was reading the data accurately, we increased the dimensions of both the arrays to 3 dimensions, to account for the channels. Since we were limited by computational resources, we reduced the training dataset to 20,000x1024 as we did not have access to a GPU, our memory resources were being exhausted immediately if all the data was loaded onto the memory at once. This was solved by using a training and test data generator which fed the data into the model in batches of size 16. In the future with more resources we intend to use the full dataset with 289,205 values and also test our model once fully trained against the validation dataset in the NSynth dataset which we are not currently using.

**2.2 Related Work**

There has been an increasing amount of research being done in the audio field with deep learning models especially since the past 3 to 4 years. Some of the notable works which we encountered during our work include:

i) “CREPE: A Convolutional Representation for Pitch Estimation” by Kim, Salamon, Li, Bello. [1]

ii) “Deep Pitch: Wide-Range Monophonic Pitch Estimation using Deep Convolutional Neural Networks” by Lloyd Watts [2]

iii) “Convolutional Neural Network for Robust Pitch Determination” by Su, Zhang, Zhang, Gao [3]

**3 Architecture**

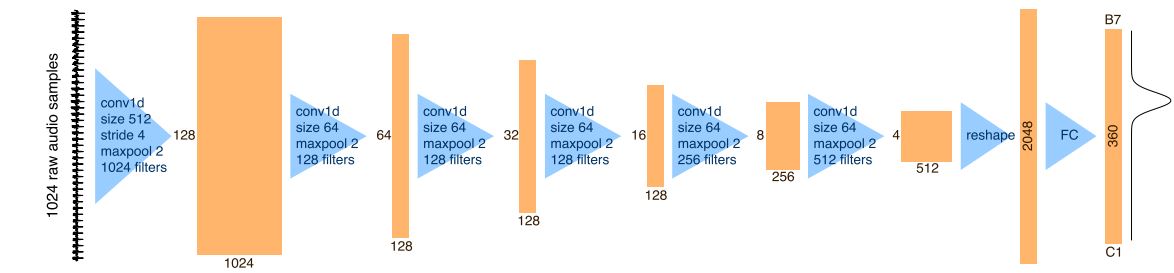
**3.1 CNN**

A Convolutional Neural Network has four main operations (Convolutions, Relu, Pooling/SubSampling, and Fully Connected Layers). Through the Convolutional layers, key features are getting extracted and moved on to the next layer that will refine the detail of the re-pooled sample. Rectified Linear Unit is important because its output is what needs to be learned from the convolutional linear operations by replacing all negative values with zero and keeping the input value if positive. Other functions that could be useful in replacing RELU would be tanh or sigmoid functions although RELU tends to outperform its non-linear counterparts. The pooling process is essentially down-sizing the dimensionality and will still retain the important values from the feature map. The important values can be pooled in a variety of ways including Max, Average, Min, Sum, etc.. In the final fully connected layers, each neuron in the previous layer must be connected to each neuron of the next layer. Essentially the convolutional and pooling layers result in high level features while the fully connected layer classifies these features.

**3.2 Model**

Using the CREPE model as inspiration, we adopted the architecture while making appropriate changes when required to fit our objective of pitch estimation. A critical difference was that in the CREPE model the final layer, the ‘2048-dimensional latent representation is then connected densely to the output layer with sigmoid activation corresponding to a 360-dimensional output vector’[1]. Since the midi pitch ranges from 0 to 127, our final output layer yielded a 128-dimensional output vector, which can be matched with our one-hot encoded y inputs using softmax activation.

Figure: CREPE Model Architecture



The figure above is the original CREPE Model Architecture[1]. We will see the modifications in the implementation part of the report.

We initially created a model, and then created a second one by adding batch normalization and dropout layers, which we will analyze in the implementation and results section.

Our initial model has 6 convolutional layers with varying filters and sizes. The stride parameter is set to 4. Each 1-dimensional convolutional layer is followed by a 1-d maxpool layer which reduces dimensionality of the layer output and feeds into the pipeline to the next layer. After the sixth convolution and pooling, the pipeline output has dimensions 512x4, which are reshaped to 2048 to obtain a latent representation. Then through a fully connected layer using softmax activations we obtain a 128-dimensional final output.

In our second model, we added 8 dropout and batch normalization layers which can be observed in the contrasting images in the next section. Also, in our second model we used a much larger dataset to train the model.

**3.3 Implementation**

The figures below were obtained from our code by running a model summary through the Keras Framework and represent our implementation of the two models we implemented.

Figure: Model 1 Summary Obtained from Keras Model Framework

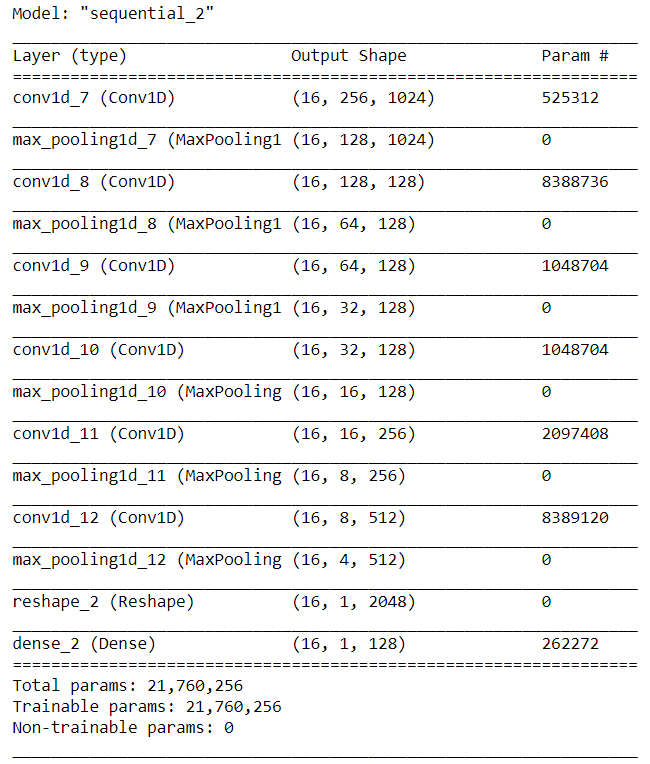
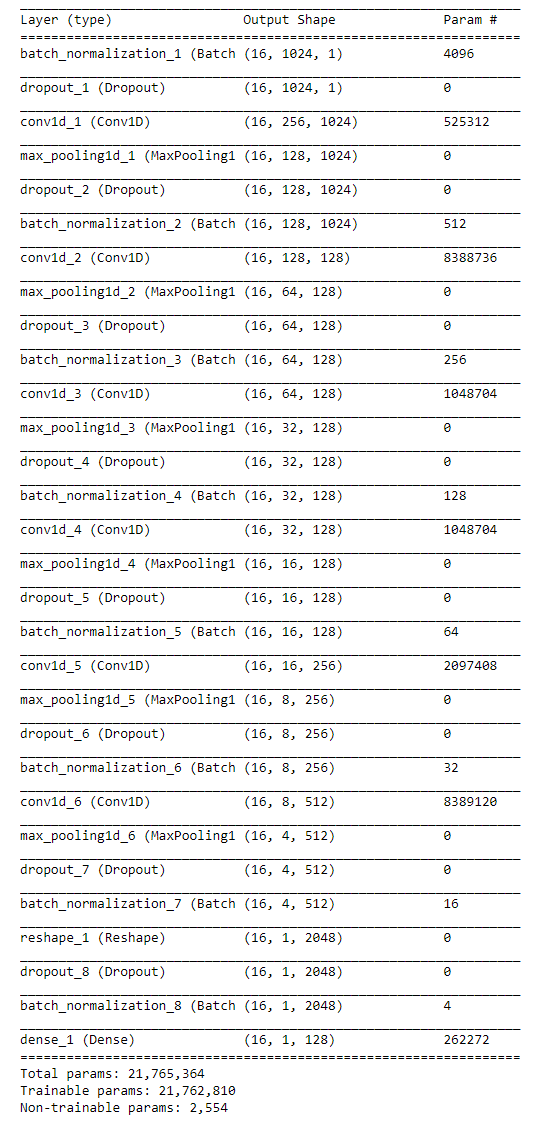


Figure: Model 2 Summary Obtained from Keras Model Framework



Due to a lack of memory resources, a batch generator was used which would feed the data to the model in batches instead of loading the whole dataset into memory which was not possible. We used a batch size of 16 which is represented in the model above.

Some of the key differences are that the second model has additional batch normalization and dropout layers. Also note that the audio files have a single channel unlike images for which the channel is typically 3.

**3.4 Hyper Parameter Tuning**

To obtain optimum results, we tweaked multiple hyperparameters. The batch size was set to 16 after gauging the resources occupied by the system and the time it was taking to compute and execute the model, which resulted in the most efficient use of our resources. We set the number of epochs to 32 after observing the model was not fully converging at 16 epochs. After our first revision we found significant overfitting. To combat this we added label smoothing to our binary cross-entropy function, increased the learning rate, increased the size of our training set and added batch normalization and dropout layers.

Once we implement the model on a larger dataset, we will further tweak the model to work optimally on the newer dataset, using our previously acquired knowledge of the model.

The table below summarizes the different hyperparameters in both the models.

Table: Model 1 vs Model 2 HyperParameters Summary:

|  |  |  |
| --- | --- | --- |
| **Hyper Parameter** | **Model 1** | **Model 2** |
| Atom Learning Rate | 0.001 | 0.002 |
| Train Size | 2867x1024 | 20,000x1024 |
| Test Size | 1228\*1024 | 4096x1024 |
| Label smoothing | ~ | 0.2 |
| Layers Additions | ~ | Batch Normalization |
| Layers Additions | ~ | Dropout Layers (p=0.25) |

**4 Experimental Results and Analysis**

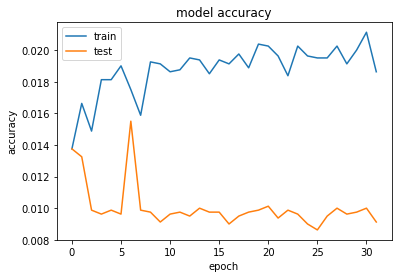
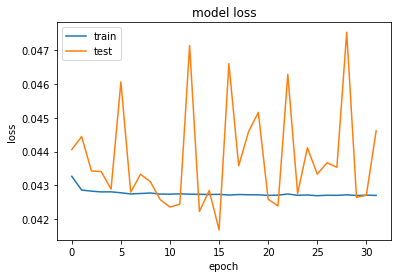
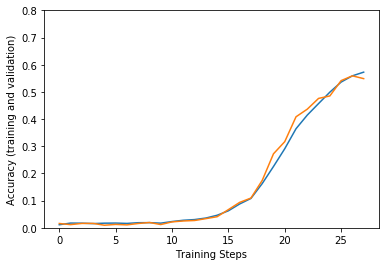
**4.1 Results**

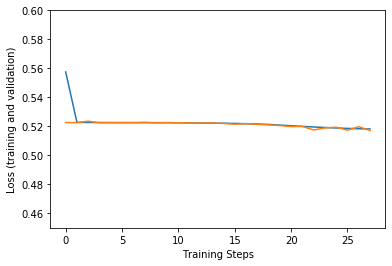
Considerable progress was made with our second model which achieved good metrics.. Our accuracy and loss metrics along with the corresponding graph representations are below:

Table: Accuracy and Loss Metrics for the Model 1

|  |  |
| --- | --- |
| **Accuracy Metrics for Model 1** | 2.06 % |
| **Loss Metrics for Model 1** | 0.04 |
| **Accuracy Metrics for Model 2** | 59.65 |
| **Loss Metrics for Model 2** | 0.52 |

Figures: Accuracy vs Epoch Graphical Representation for Model 1(Left) and Model 2 (Right)

Figures: Loss vs Epoch Graphical Representation for Model 1 (Left) and Model 2 (Right)



**4.2 Evaluation, Analysis and Improvements**

Our initial loss and accuracy metrics have been greatly improved from model 1 to model 2. Some of the reasons for the figures being in the ranges they are in is that we were limited by computational resources to not fully utilize the complete training dataset with 289,205 data samples as we only used about 4096 samples in model 1 and 20,000 randomly sampled ones in model 2.

In the first model our accuracy and loss metrics were 2.06% and 0.04 respectively. In the second model our accuracy and loss metrics were much better at 59.65% and 0.52 respectively.

Further, as we can see in model 2, the model starts to converge after 15 epochs. Since we only ran the model for 28 epochs, we didn’t reach its full potential as it was still making consistent progress upon completion of 28 epochs. By utilizing the full datasets and running the model to a higher number of epochs such as 50 or so will see even better improvements in model 2.

There are a multitude of ways in which we can further improve our model and it’s accuracy metrics towards pitch estimation.

Firstly, by increasing our computational resources we will be able to utilize the full extent of the NSynth Dataset, as currently we are using less than 10% of the training dataset samples. Doing so will train the model with a better capability to learn and estimate pitch, especially when it will be exposed to real life sounds.

Secondly, we are currently using 1024 samples from the full 64000 samples of each audio file spliced from the beginning. Ideally, using the whole 4-second audio clip will greatly enhance the quality of our models pitch estimation as the full audio sample conveys the rise and fall of a sound signal more holistically than a sample splice of the values.

Additionally, once we are able to utilize the full dataset and see improvements in our model, we can further tweak the model’s hyperparameters and layers to more accurately represent the human audio processing architecture through the model. When done so accurately and with the correct hyperparameters metrics and model should be greatly more accurate.

Another possibility which we haven’t yet fully considered is that if we use other metrics such as timbre along with the raw audio samples to further enhance the accuracy of the model by adding another dimension and weighing it accordingly.

**5 Future Applications**

Now that we have established our model and proven results,there are many possibilities for future applications for pitch detection using deep convolutional networks. There are many medical applications, especially in machine hearing which can make use of such a model. Especially, if we use other factors such as timbre when training our model, it can build upon to create a more holistic model which can estimate other features including timbre as well, ideally concurrently with pitch. Also for audio, pitch estimation can aid with removing detuning in songs and voices, along with many other applications such as additive voice reverberation.

**6 Conclusion**

To conclude, we have been effectively able to estimate pitch in the midi range [0,127] using deep convolutional networks from raw audio signals as we have seen. Our second model reached accuracies of nearly 60% while only using less than a tenth of the available dataset and running a limited number of epochs, a considerable improvement from the initial model, and was obtained by modifying the architecture and tuning the hyperparameters. We have established our proof-of-concept for the project. The significance of this project lies in that we don’t convert the audio files to MFCC or other graphs, and rather use the raw audio data sampled from the audio files. We will keep on working on this project with increased computational resources and aim for publication once we complete training the model with the full dataset.

**6 Team Members’ Contributions**

|  |  |
| --- | --- |
| **Task** | **Contributions** |
| Developing Concept/Model for Project | Sean |
| Data Collection, Virtual Machine Setup and Data PreProcessing | Piyush |
| Model 1 Implementation | Sean, Piyush |
| Model 2 Implementation | Sean |
| Model Metrics and Visualizations | Shrey, Piyush |
| Report Formatting | Shrey, Piyush |
| Presentation Research and Formatting | Shrey |
| Evaluation and Analysis | Piyush, Sean |
| Report Submission | Shrey |

**References**

[1] Jong Wook Kim, Justin Salamon, Peter Li, Juan Pablo Bello (2018) CREPE: A Convolutional Representation for Pitch Estimation. Accessed from: https://arxiv.org/abs/1802.06182

[2]Watts, 2018 Deep Pitch: Wide-Range Monophonic Pitch Estimation using Deep Convolutional Neural Networks. Accessed from: http://www.lloydwatts.com/images/2018-06-14\_MonophonicPitchPaper.pdf

[3]Su, Zhang, Zhang, Gao, 2016, Convolutional Neural Network for Robust Pitch Determination, Accessed from: https://ieeexplore.ieee.org/document/7471741

[4] Alexander, J.A. & Mozer, M.C. (1995) Template-based algorithms for connectionist rule extraction. In G. Tesauro, D. S. Touretzky and T.K. Leen (eds.), *Advances in Neural Information Processing Systems 7*, pp. 609-616. Cambridge, MA: MIT Press.

[5] Bower, J.M. & Beeman, D. (1995) *The Book of GENESIS: Exploring Realistic Neural Models with the GEneral NEural SImulation System*. New York: TELOS/Springer-Verlag.

[6] Hasselmo, M.E., Schnell, E. & Barkai, E. (1995) Dynamics of learning and recall at excitatory recurrent synapses and cholinergic modulation in rat hippocampal region CA3. *Journal of Neuroscience* **15**(7):5249-5262.