Pitch Detection with Machine Learning

Young-Shiuan Hsu
ENEE632 Speech and Audio Processing – Spring 2022

***Abstract*—There have been many different algorithms and techniques for pitch extraction. However, there are not a lot of experiments done in machine learning. This project focus on how different machine learning models perform on pitch extraction.**

Keywords—pitch extraction, machine learning, CNN, fully connected model, transfer learning, MFCC, audio processing, PTUB-TUG Pitch tracking database.

INTRODUCTION

Pitch is the highness and lowness of our voice, and it is incredibly important in different languages, especially English. We use pitch in order to express our emotions and attitude through a change in our intonation, or the tone of our voice. Research related to pitch has been very popular in various fields such as health care, speech synthesis, and speech recognition. Since machine learning has been a very popular method for classification nowadays, it is worth experimenting with different machine learning techniques and analyzing its performance.

This project will be focusing on how different machine learning models perform. There will be three steps through the system design. The essential part is the data preprocessing section otherwise the trained model is meaningless. While training the model, we also have to try out different loss functions, activation functions, layers, and the number of nodes in each layer. To see if the result matches our expectations, we have to plot pitch movement and calculate the error to compare which model performs better.

TRAINING DATASET

The training dataset [2] for this project is the PTDB-TUG database produced at Graz University in 2011 for pitch tracking. It consists of 10 female and 10 male speakers. Each individual has a record of approximately 200 different sentences, with a total of around 6000 seconds of recording. The sampling rate of each audio file is 48K Hz.

SYSTEM DESIGN

I. Data Preprocessing

Every audio file in the dataset contains a huge amount of silence. For instance, the first and last second is silence in a three-second audio. To avoid bias in the training dataset, we have to get rid of the silence for every audio file. Then we have to take every 480 samples(10 ms) for each frame and assign a pitch period for the frame according to the labeled dataset. The last step is to normalize the dataset by dividing each value by the maximum value. As a result, we can control the dataset value ranging from -1 to 1 to decrease the influence of outliers.

II. Model Training

By inserting each frame(480 samples) into the model(figure 1), we are able to train the model with approximately 600,000 fames. We are able to try the 1D CNN model, fully connected network, feature model(append the MFCC feature extraction data to the original data), and transfer learning with the original raw data. By calculating the spectrogram of the audio file, we can obtain 2D data which will be the input for training a 2D CNN model.

The table below is the number of layers, activation function, and loss function that is found that has the best performance through trial and error.

Model Name	Layers	Activation function	Loss function
1D CNN	2 feature training, 2 flatten layer	Relu, Sigmoid	sparse_categorical_crossentropy
2D CNN	1 feature training, 2 flatten layer	Relu	sparse_categorical_crossentropy
Fully Connected	5 flatten layer	Relu, Sigmoid	sparse_categorical_crossentropy
Feature Model	5 flatten layer	Relu	sparse_categorical_crossentropy
Transfer Learning	3 flatten layer (appended)	Relu	sparse_categorical_crossentropy

III. Result

The output of each frame is a 400*1 vector and the corresponding index of the maximum value in the array will be the pitch assigned to the particular frame.

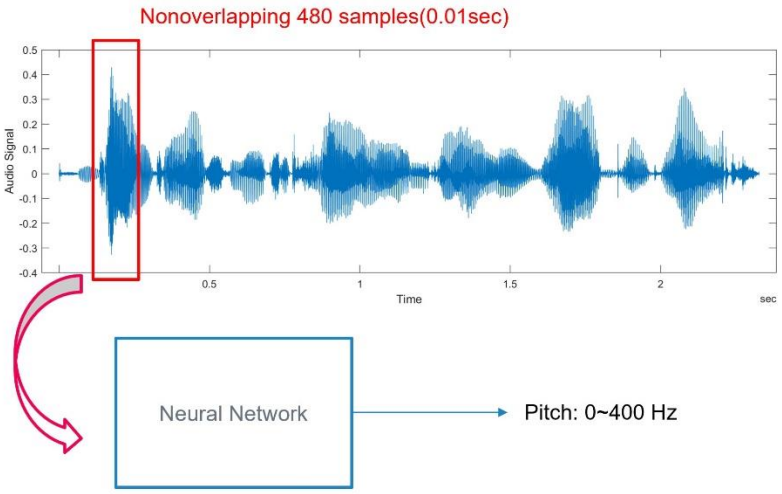

Figure 1

IV. Crepe Package[1]

Python has an inbuilt package for pitch tracking with an accuracy of 99%.

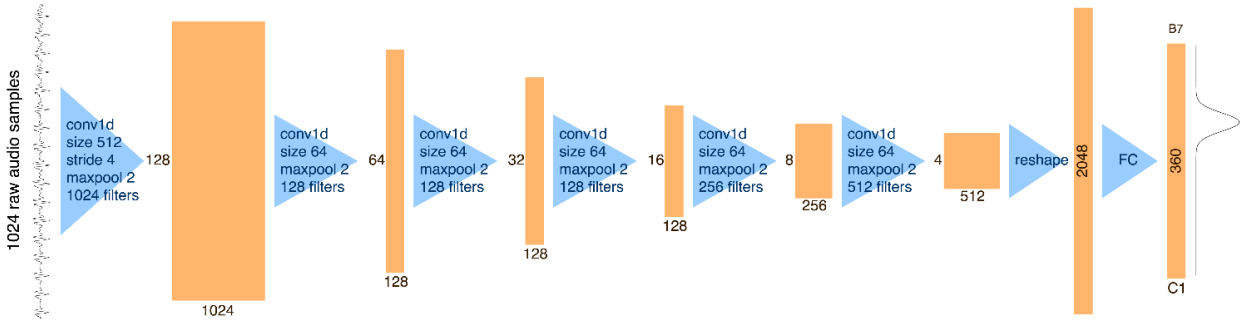

Figure 2

RESULT

The testing data shown in all the results figures are two different speakers. The female speaker is trying to say "Don't ask me to carry an oily rug like that" (Figure 3). In general, female has a pitch from 165Hz~255Hz, and we can see through the pitch movement in the second row which the range is 180Hz~285Hz. The male speaker (Figure 4) is trying to say "The best way to learn is to solve extra problems". According to the pitch movement in the second row, there is no obvious pitch movement in this audio and it ranges from 70Hz~160Hz.

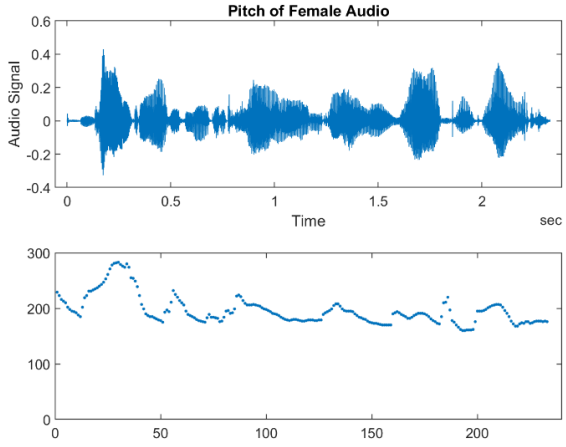

Figure 3

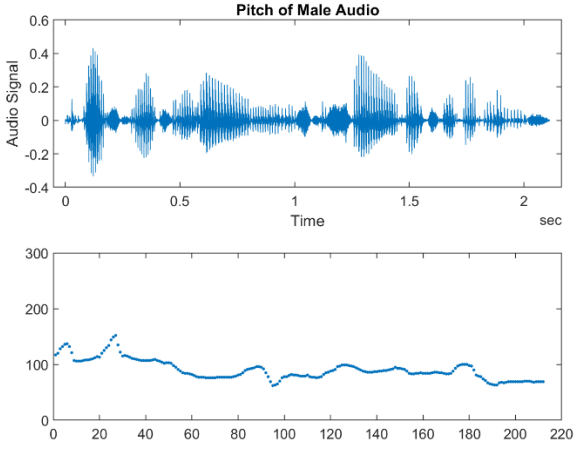

Figure 4

I. 1D CNN model

The first row of the result below is the audio signal of the original audio file (.wav). The second row is the exact pitch given in the dataset. The last row is the prediction of the pitch period of the given frame. The model is trained with 75 epochs with an accuracy of 0.1013. The model is able to successfully pick up the pitch transition for the female speaker, but for the male speaker, the output is not as good as I expected. In this case, the pitch lies around 80Hz but is not doing very well to pick up small transitions.

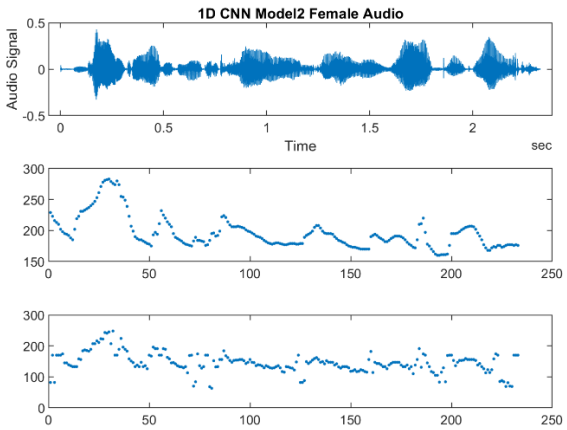

Figure 5

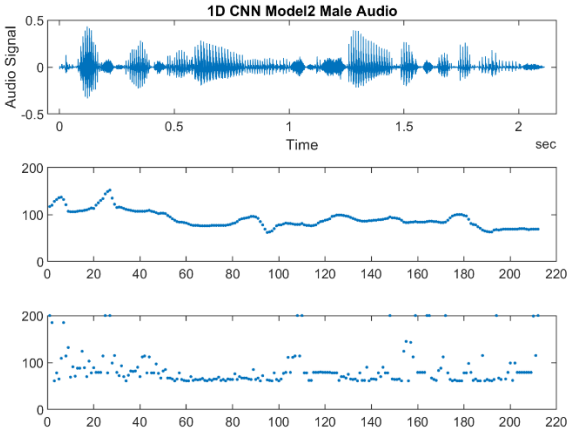

Figure 6

II. 2D CNN model

We can also transfer the data into 2D data by calculating its spectrogram. By shifting 40 samples every time we can turn a 1D 480 samples data into 512*12 data as the input. However, since all the data are stored as float in this case, with limited RAM, we are only able to load 6212 frames to train the model. With a small training dataset, the prediction of the model is doing surprisingly well. By training with 50 epochs, we are able to obtain an accuracy of 0.0918. More importantly, we can see there are fewer uncertainties for the prediction of male speech compared to the prediction of the 1D CNN model.

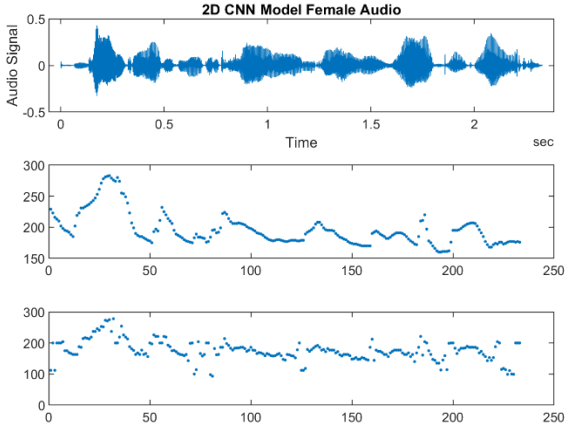

Figure 7

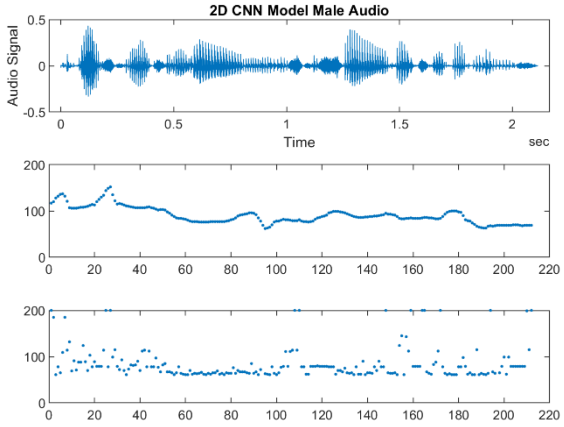

Figure 8

Now, if we compare the error value of the prediction of the 1D CNN model and the 2D CNN model. We can clearly see that the 2D CNN model is doing a lot better for male speakers. Although we cannot obtain an accurate transition, we have greatly reduced the error. For the female speaker, the error for 1D and 2D CNN are performing similarly well.

Model	Female Speaker	Male Speaker
1D CNN	5576	10761
2D CNN	5832	5759

Table 1

III. Fully Connected Model

With a simple model with 3 hidden layers, we can pick up all the pitch transitions for the female speaker. For the male speaker, we can somehow see there might be some pitch movement in the beginning and the pitch lies around 80 Hz throughout the speech.

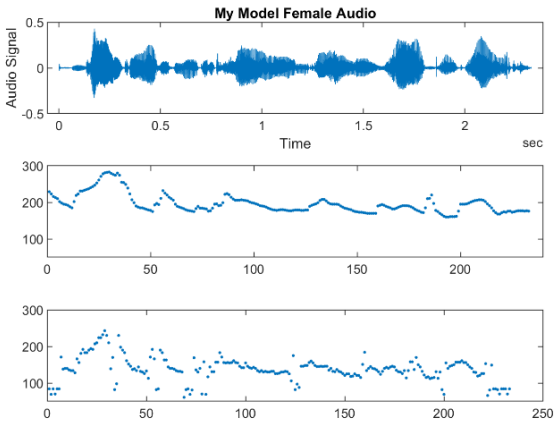

Figure 9

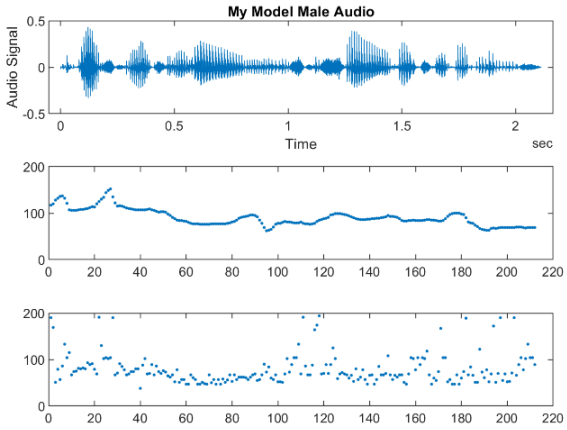

Figure 10

IV. Feature Model (Data + MFCC extracted data)

In this case, I am only taking the 26 dimensions of MFCC feature extraction with the package provided by python. Taking the derivative of the output of the 13 dimension data, we can obtain a total of 26-MFCC data and append it to the existing frame data. By doing so, we can clearly see that the uncertainties in the male speech have been greatly reduced. Therefore, we can see that the additional 26 samples are very powerful.

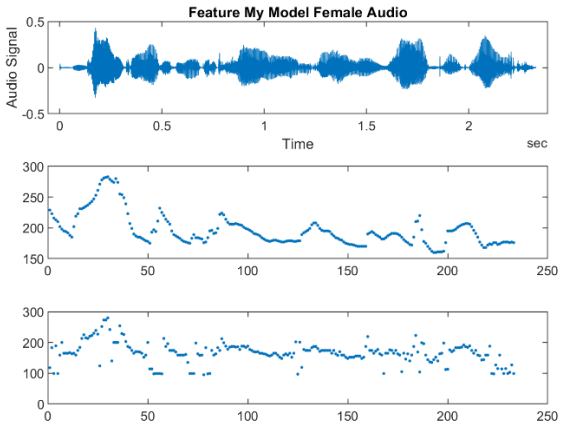

Figure 11

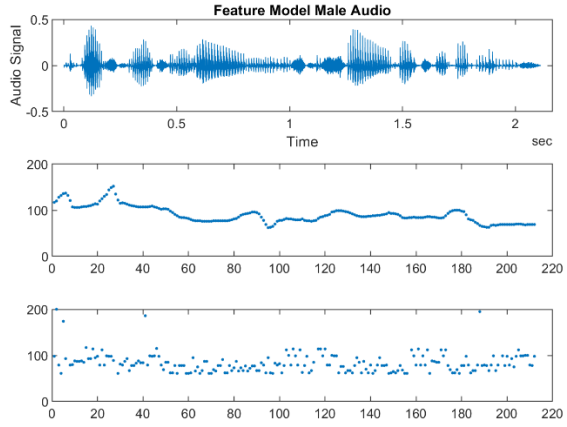

Figure 12

If we compare the error of the two speakers with different data input, I notice that with only 26 samples, we could reduce the error.

Model	Female Speaker	Male Speaker
Fully Connected Model	8042	5167
Feature Model	7489	4053

Table 2

V. Transfer Learning

In this case, I introduced the VGG16 pre-trained model and appended another 6 layers at the end, and froze the first 3 layers. Since transfer learning is taking a significant amount of time, I am only able to train it for around 15 epochs(one epoch takes around 3.5 hours). With only 15 epochs, it is not useable but we can still see some structures in the female speaker. The pitch raises sometimes at the beginning of the speech, but for the male speaker, it is could not pick up anything.

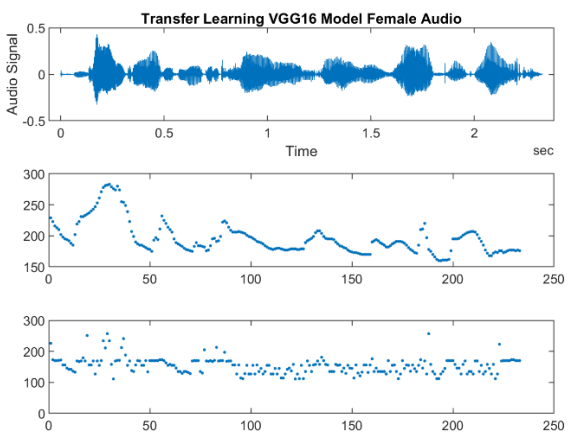

Figure 13

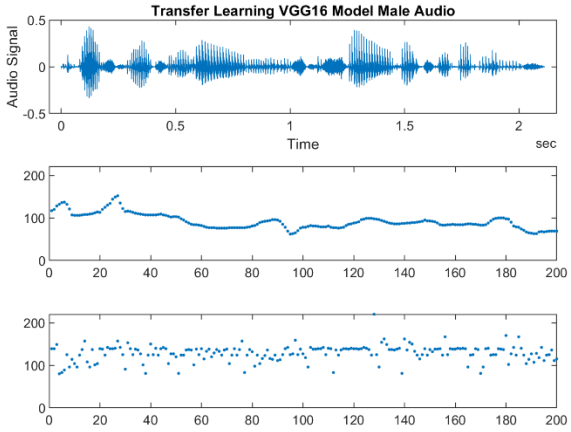

Figure 14

VI. Testing Output

At last, by trying out all the datasets, I also wanted to play around with my own voice. The figure below is the audio when I try to say “The weather is very nice today”. The first row in the figure is the audio waveform, and the second row is the Crepe pitch tracking package inbuilt in python, which we can take as the exact pitch transition of this audio file. The third row and last row is the prediction of the fully connected model and the 1D CNN model I trained previously. We can see that the fully connected model is picking up all the pitch transitions and the structure is aligned perfectly with the prediction of the Crepe package. For the 1D CNN model, there still exist a lot more variations and the prediction around 120 is not matching. Therefore, we can conclude that in this case, the fully connected model is performing better than the 1D CNN model.

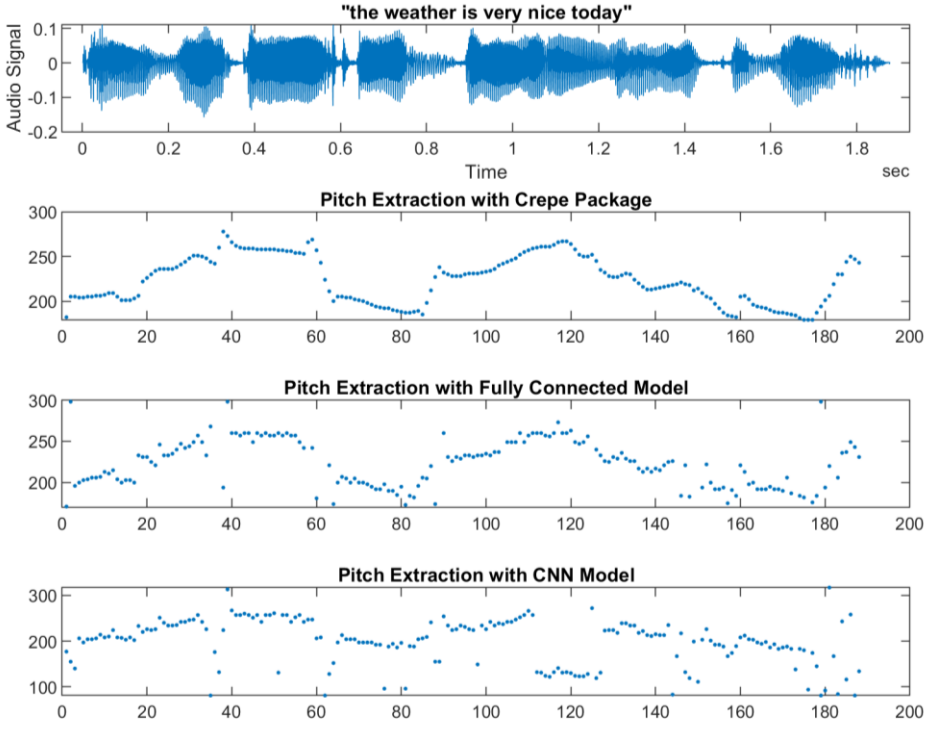

Figure 15

Conclusion

By trying out different models, we are able to see that some models perform surprisingly well. Small models are able to pick up a lot of pitch transitions through time. The fully connected model is able to pick up the pitch structure and the 2D CNN model is also performing well with only a small size of the training dataset. However, all of the models are not able to pick up the pitch transition for the male speaker case. Perhaps it might be the nonoverlapping window I am training the model since pitch could not change gradually in a short amount of time.

References

[1] https://github.com/marl/crepe

[2] https://www.spsc.tugraz.at/databases-and-tools/ptdb-tug-pitch-tracking-database-from-graz-university-of-technology.html

[3] https://en.wikipedia.org/wiki/Pitch_detection_algorithm

[4] https://www.researchgate.net/publication/324922376_An_AnalysisSynthesis_Framework_for_Automatic_F0_Annotation_of_Multitrack_Datasets

[5] https://en.wikipedia.org/wiki/Database_normalization

[6] https://www.scicoding.com/pitchdetection/

[7] https://sigport.org/sites/default/files/docs/crepe.pdf

[8] https://medium.com/swlh/fully-connected-vs-convolutional-neural-networks-813ca7bc6ee5#:~:text=A%20fully%20connected%20neural%20network%20consists%20of%20a%20series%20of,be%20made%20about%20the%20input.

[9] https://medium.com/@james.moody/is-audio-two-dimensional-f073632abc3