Emotion Classification of Audios Using Convolutional Neural Networks

Zeduo Zhang
Master of Computer Science AI
Western University
London, Canada
zzhan762@uwo.ca

Kun Wang
Master of Computer Science
Western University
London, Canada
kwang645@uwo.ca

***Abstract*—This project focuses on how to use TensorFlow to build Convolutional Neural Networks (CNNs) based model to do audio emotion classification in terms of features selection, data augmentation, learning rate and layers. We would analyze how these factors affect the performance of our model and address the overfitting problem. The datas are from the Kaggle website, called "RAVDESS Emotional Speech Audio". For data augmentation, we analyze the effect of applying noise, time-shifting, time stretching and pitch shifting to the training dataset, and choose the combination of them that improves the accuracy and reduces the overfitting most. In addition, we study the accuracy and loss using different audio features such as Mel-frequency cepstral coefficients and Mel-spectrogram, etc. and figure out which features are related to the emotion of audio and can be used as the inputs to our CNN's based model. After that, we would adjust the learning rate, and find the most suitable one for our dataset. After discussing on these techniques, we achieve our final model with simple model architecture. The final model can only reach 55.55% accuracy and still have a overfitting problem.**

***Index Terms*—speech emotion recognition, Tensorflow, Convolutional Neural Networks, RAVDESS Emotional Speech Audio Dataset**

I. Introduction

Speech is the most powerful and efficient way of human communication, and we believe speech signals can be used to be the bridge of the communication between machine and human. Machines can analyze the tone, mood and content of the speech to communicate with humans. Recently, lots of researchers focus their studies on speech content recognition. However, we know that it is not sufficient to only know the content of the speech, and different emotions can be expressed with the same speech utterance. Therefore, speech emotion recognition plays an important role in speech recognition and human-machine communication [1].

Speech emotion recognition is tough to solve because so far we do not know which features of audio signals are related to emotion. In addition, a voice may contain multiple emotions which are represented by different parts of the voice. Moreover, a speech signal can express different emotions according to different speakers, cultures and environments.

With regards to speech emotion recognition, the most important thing is to find out the most important and related features. That is a good feature or a combination of features can improve the performance of the model to a large extent. Instead of splitting the signal into blocks and studying them, we prefer to focus on the global statics of the whole audio.

That is no matter how long the audio was, the size of the features vector is always the same, and this matches the input requirement of almost all the classification models such as deep neural network, decision tree classification and K-Mean. So far, researchers have not figured out which features decide emotion. Ayadi, Kamel and Karray mentioned in [1], speech features can be split into four groups: continuous features, qualitative features, spectral features and TEO (Teager energy operator)-based features. In this project, we will talk about several common speech features which can easily be extracted by using Tensorflow. While training a model, the overfitting problem and performance are also the issues we are worried about. It is usual to use an overparameterized model to train a model if resources are sufficient. However, this would cause an overfitting problem, and thus we need to reduce overfitting using several methods such as data augmentation and dropout. In addition, the learning rate is also an important problem we need to discuss since it can help to reduce the training time and approach the optimal solution. So, we would discuss learning rate decay and learning rate reduction in this project. As for the detailed information of our project, we use Python as the programming language, Tensorflow as the platform, and Convolutional neural networks (CNNs) based model as our candidate model. The reason we choose Tensorflow is it provides sufficient and useful functions that allow us to easily do the speech data augmentation, features selection and building CNN based model. Since the features we use are like Mel-frequency cepstral coefficients and Mel-spectrogram which can be viewed as a graph, and thus we use CNN based model as our candidate model.

Audio processing and speech emotion recognition are common problems we would face as a learner of machine learning. Therefore, it is good to know how to use Tensorflow and Python to process and visualize audio signals and recognize the speech emotion using CNNs.

II. DATASET

The dataset that is used in this project is an online open-source named Ryerson Audio-Visual Database of Emotional

Speech and Song (RAVDESS) [2] which originally contains 7356 files. Only 1440 files of this dataset are used in our project. These 1440 emotional speech audios are vocalized by 24 professional actors which are 12 female, 12 male and 60 trials per actor. The speech contents are two lexically coordinated statements which are "Kids are talking by the door" and "Dogs are sitting by the door" in a neutral North American accent. There are totally eight speech emotions: neutral, calm, happy, sad, angry, fearful, disgust and surprised. Each of the audio files has a unique filename that explains all the characteristics of the audio sample. For instance, for a file name like" 03-01-05-01-01-01-08.wav", it means audio-only (03), speech (01), angry (05) normal intensity (01), statement "kids" (01), 1st repetition (01) and 8th actor (08). For more information, one can check the Kaggle website [3]. Each speech emotion contains 192 audio samples, except for the "neural" emotion, which only has 96 audio samples. 60% of the samples in the dataset are used as training sets. 20% and 20% of the samples are respectively used as the validation set and testing set.

III. METHODS

A. Baseline Model

Before building and training our candidate models and analyzing the effect of each technique, we want to build a simple CNNs based model to test our dataset and make sure that the basics of the ideal process of our project work. In addition, the performance of this baseline model, including accuracy and loss, can be considered as the lower bound or comparison of our future modified models.

Our baseline model contains two regular loops (one convolutional layer and one max pooling layer), one flatten layer and one dense layer. We use Adam as our optimizer with a learning rate of 0.001, categorical cross-entropy as our loss function, accuracy as our metrics and softmax as the activation layer of the output. This setting would remain the same for our future candidate model except for the learning rate and hidden layers. The structure of the baseline model can be visualized in Fig. 1.

The training accuracy and validation accuracy are shown in Fig. 2, and the training loss and validation loss are shown in Fig. 3. From the accuracy plot, we can see that the process is relatively stable and the validation accuracy achieves approximately 50%, and finally converges after 150 epochs. From the loss plot, the baseline model overfits after approximately 50 epochs. The model finally achieves 61.8% accuracy and a 1.33 loss value. The results of the baseline model shown above would be used for comparison in our project.

B. Data Augmentation

We can create more different but related examples to the training datasets, to improve the diversity and complexity of our training dataset and consequently improve the performance and address the overfitting problem [4]. In this project, we simply analyze four data augmentation techniques that can simply be coded using Numpy and Librosa library in Python and choose the best one to use [5].

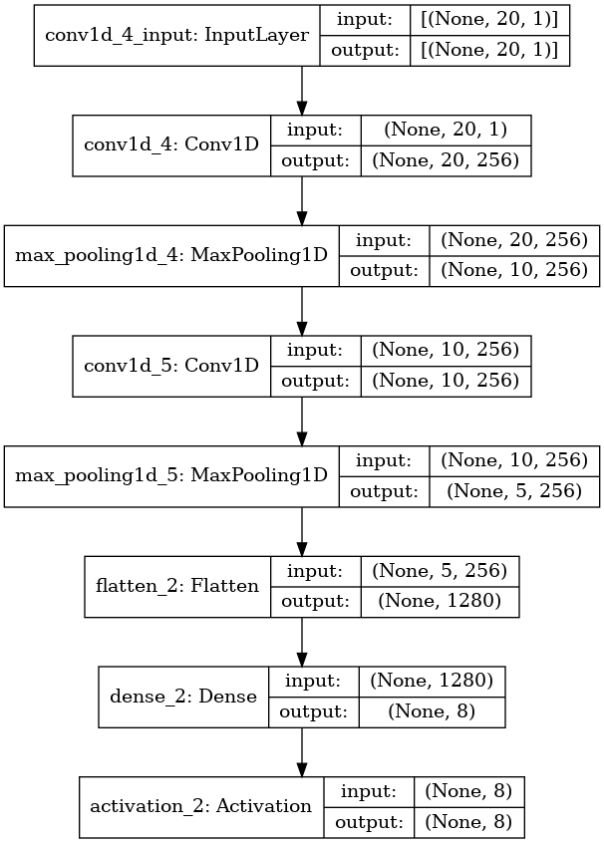

Fig. 1. This is the visual structure of our baseline model

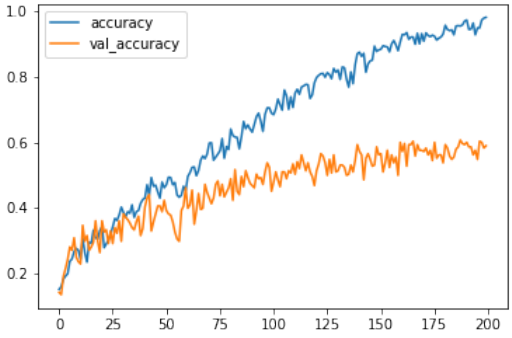

Fig. 2. This plot shows the training accuracy and validation accuracy of the baseline model

- Noise
  We use a function in Numpy to add a random normal noise which is multiplied by a small rate to the audio signal.
- Time shifting
  Time shifting is just simply shifting the audio to the left or right. For example, if we shift the audio to the right for

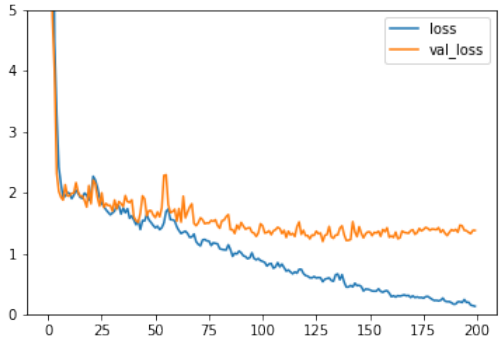

Fig. 3. This plot shows the training loss and validation loss of the baseline model

n seconds, the wave plot of the audio would be shifted to the right for n seconds, that is the audio would delay n seconds to play the sound. It is easy to use a function in Numpy to achieve this.

- Time stretching
  For this part, we change the speed of the audio by stretching the time series with a specified rate, that is the wave plot is stretched along the x-axis, and the speed of the sound is also changed.
- Pitch shifting
  Pitch shifting just simply shifts the audio signal upward or downwards with a specified rate. As mentioned in [6], this pitch shifting is time-independent, that is adjusting the pitch does not affect the speed and vice versa.

The effect on the audio signal of the data augmentation techniques mentioned above is shown in Fig. 4. From the perspective of humans, we still can figure out that these four audio signals are from the same example in the dataset.

We use the baseline model to test the effects of combination of different data augmentation techniques, and the results are shown in Fig. 5 and Fig. 6 which are corresponding to validation accuracy and validation loss.

From the validation accuracy graph, we can see that adding data augmentation techniques improve the accuracy of the model before around 125 epochs. After 125 epochs, the baseline model outperforms other models with data augmentation except for the yellow line (model with noise data). As a whole, the technique of adding noise to the training data has the highest accuracy. From the validation loss graph, we can see that, with a large number of runs, adding data augmentation doesn't reduce the overfitting, on the opposite, the overfitting problem gets worse. However, before around 50 epochs, the models with data augmentations have lower loss values than the baseline model. Especially, the yellow line has a lower loss value than the baseline model before around 85 epochs at which the yellow line has higher validation accuracy than the baseline model. To overcome this tradeoff, we can use early stopping to stop the model earlier and save the best model before it overfits. Based on the current results, we can see

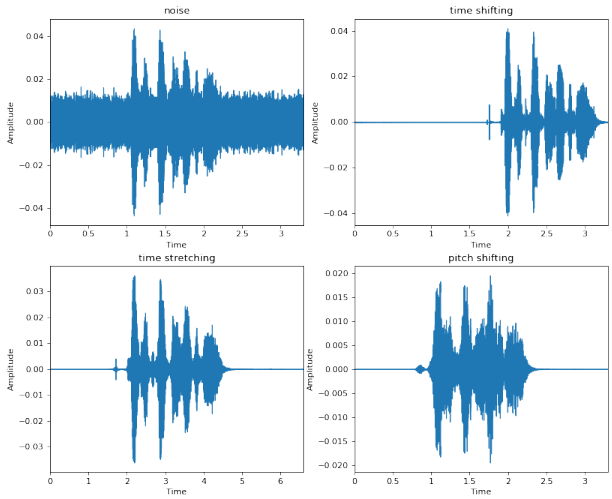

Fig. 4. These are the audio signals modified by different data augmentation techniques, and the corresponding techniques are shown in the title of each plot. The top left is adding the random normal noise with factor 0.004. The top right is time shifting with factor rate*100. The bottom left is time stretching with factor 0.5. The bottom right is pitch shifting with factor -5.

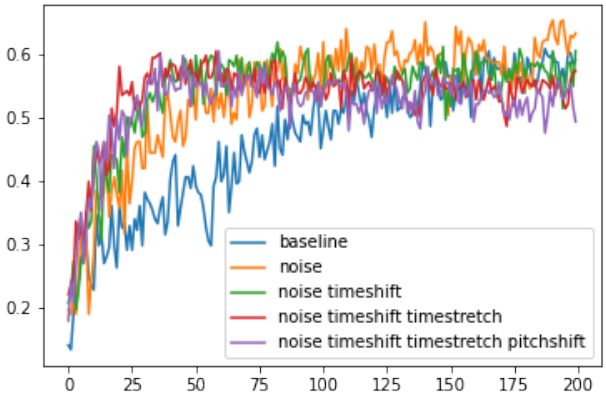

Fig. 5. This plot shows he validation accuracy of different combination of data augmentation techniques. The involved data augmentation techniques are indicated on legend.

that only using noise technique to do the data augmentation makes the model performs the best.

C. Features Selection

As for features selection, we cannot make sure which feature or combination of features are significantly related to the emotion of audio. Therefore, in this section, we would talk about several interesting and famous audio features and analyze their effects on audio emotion recognition with CNN based model.

- Zero-crossing Rate

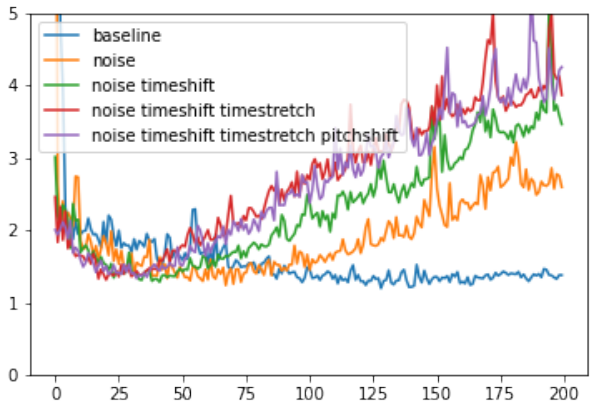

Fig. 6. This plot shows he validation loss of different combination of data augmentation techniques. The involved data augmentation techniques are indicated on legend.

zero-crossing rate (ZCR) is a rate at which a signal changes from positive to negative or from negative to positive, which means it is the rate of sign-changes of the signal during the waveplot [9]. Given by [7], the ZCR is defined by the following equations.

$$Z(i) = \frac{1}{2W_L} \sum_{n=1}^{W_L} |sgn[x_i(n)] - sgn[x_i(n-1)]| \quad (1)$$

where sgn() is the sign function, i.e.

$$sgn[x_i(n)] = \begin{cases} 1 & x_i(n) \geq 0, \\ -1 & x_i(n) > 0 \end{cases} \quad (2)$$

Since ZCR is a rate used to measure the noisiness of audio, and thus it kind of has no relationship with the emotion on which we focus in this project.

- Mel-frequency Cepstral Coefficients (MFCCs)
  The advantages of MFCCs mentioned in [8] are they can describe the overall shape of the whole structures of the spectrum and it is a feature that is widely used in speech recognition systems. Lalitha et al. demonstrate in their article [9] that MFCCs are commonly used in Speech Emotion Recognition (SER), and there are lots of algorithms where MFCCs are included in their feature vector. Therefore, using MFCCs as one of the features is significant and useful.
- Melspectogram
  Roberts has already demonstrated the Mel Spectrogram clearly in [10], and we will follow his outline to explain what Mel Spectrogram is, and why it is a significant feature in our study.
  - Fourier Transform
    Normally, an audio signal is a combination of lots of single-frequency sine or cosine waves, that is an audio signal can be separated into a single wave that can be represented by amplitude and frequency. And for each wave, we only capture its amplitude and frequency and then put it into the plot that has frequency as its x-axis. In addition, Fourier transform is a technique for us to transfer a wave into its corresponding frequency and frequency amplitude. Especially, fast Fourier transform (FFT) is widely used in signal processing which can compute the Fourier transform efficiently.
  - Short Time Fourier Transform
    For a speech, the content of audio always and frequently changes, that is the signals are not periodic. To do this, we have to separate the signal into several overlapping windowed segments, and we compute the FFT of each segment,
  - Spectrogram
    The spectrogram is a detailed and visual representation of the spectrum, which can be used to represent the time, frequency, and amplitude [12]. It displays the changes in the frequencies in a signal over time and the frequencies' amplitude is represented in another dimension which is usually represented by variable brightness or colour.
  - The Mel Spectrogram
    Before talking about the Mel-spectrogram, we have to mention the mel scale, which transfers the frequency to the unit of pitch, and it is easier for humans to perceive. Obviously, "a Mel-spectrogram is a spectrogram where the frequencies are converted to the mel scale", defined in [12].
- chroma_stft
  Stft stands for short-time Fourier transform which is mentioned above. Chroma_stft is to compute a chromagram from a waveform or power spectrogram [13].
- chroma_cqt
  Chroma_cqt uses CQT to generate a spectrogram and then projects that spectrum to represent a chroma. CQT stands for Constant-Q chromagram. Compared with STFT, the frequency scale of CQT is logarithmic, not linear. It is, in general, more appropriate to musical data.
- chroma_cens Chroma_cens is to compute the chroma variant chroma energy normalized statistics(CENS) [14]. The basic idea of CENS is using statistics over large windows smooths local deviations. It connects to the short-time harmonic content of the audio signal [15]. It also has variations of properties such as dynamics, timbre, articulation, and temporal micro-deviations.

Based on the description above, we choose MFCCs and Melspectorgram as our candidate features and analyze both of them and the combination with other insignificant features. The results are shown in Fig. 7 and Fig. 8. Then chorma stands for the combination of chroma_stft, chroma_cqt and chroma_cens.

From Fig. 7, we can see that using only MFCCs performs the best. In addition, adding additional features to the model with MFCCs still lowers down the accuracy. From Fig. 8,

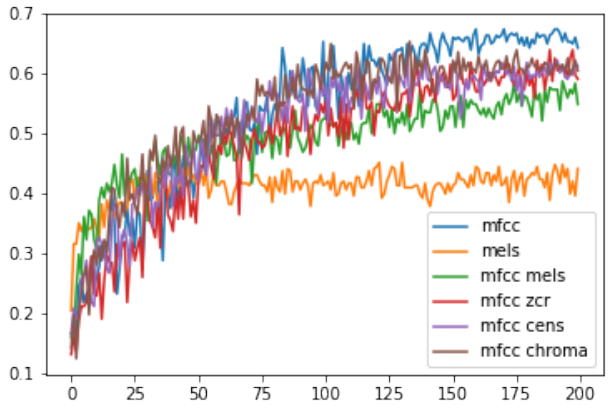

Fig. 7. This plot shows the validation accuracy of model with baseline structure and different features. The involved features are indicated on legend.

except for Melspectorgram and the combination of Melspectorgram and MFCCs, the others have low and stable loss values. Therefore, we can conclude that the MFCCs are the most suitable features within the features discussed above in this project.

D. Learning rate

It is really important to choose a suitable learning rate for a specific dataset and model [11]. It is a hyperparameter that determines how much the model weights are changed each step in response to the loss function. If the learning rate is too small, the training process would be pretty long and it takes a long time for the model to converge. If the learning rate is too large, we probably won't be able to reach the optimal weight since we are just bouncing around, and the training process would be unstable. Instead of keeping the learning rate unchanged throughout the training process, we can use a dynamic learning rate. We will discuss and compare the effect of exponential decay and reduction in this project.

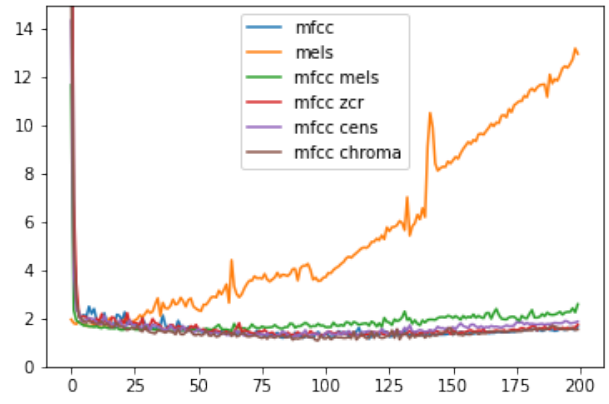

Fig. 8. This plot shows the validation loss of model with baseline structure and different features. The involved features are indicated on legend.

- Learning Rate Decay
  For decay, we use tensorflow.keras.optimizers.schedules.ExponentialDecay() function provided by Tensorflow. The formula of ExponentialDecay is defined in Keras API reference site [4].

```
def decayed_learning_rate(step):
        return initial_learning_rate *
        decay_rate ^ (step/decay_steps)
```

  Using the exponential decay schedule, the learning rate is lowered down each step. According to the exponential decay function, the amount of learning rate to be reduced depends on the initial learning rate, decay rate and decay step you set. Thus, we use different initial learning rates, decay_rate and decay_steps to train the model, compare their effects and choose the best one.
- Learning Rate Reduction
  For reduction, we use tf.keras.callbacks.ReduceLROnPlateau() function provided by Tensorflow. The rate reduction is different from rate decay. The rate does not decrease every step, it would only be reduced when a metric has stopped improving. We can set patience which is the number of epochs with no improvement after which the learning rate will be reduced, and set the minimum rate which is the lower bound of the learning rate [5].
  - factor: the rate to reduce the learning rate by lr = lr * factor.
  - min_lr: the lower bound of the learning rater.
  - patience: if the metrics do not change after this number of epochs, the learning rate would decrease.

We want to analyze the effects of the different learning rates, and the results are shown in Fig. 9 and Fig. 10. From the graphs, we can see that when the learning rate is too large like 0.01, the model cannot learn anything from the training data set. When the learning rate is too small like 0.0001, it takes a long time for the model to learn and reach the optimal solution. Therefore, we want to choose the appropriate initial learning rate which is 0.001 based on the information from figures. And choose the suitable learning rate schedule to decrease it.

We know the batch size is 64, and we have 1440 * 0.6 * 2 =1728 training audio signals since we use 60% of the datasets as training datas and use noise as the data augmentation techniques. Then there are 1728/64 =27 steps in each epoch. For example, if we want the learning rate to exponentially decay with decay_rate every epoch, we have to set the decay_steps to 27. Therefore, in order to choose the best decay_steps, we fix the initial rate as 0.001 and decay_rate as 0.96 and analyze different decay_steps (27, 50,70,100) to choose the best one. The results are shown in Fig. 11 and Fig. 12. We can see that when the decay_steps is 70, the model has the best performance.

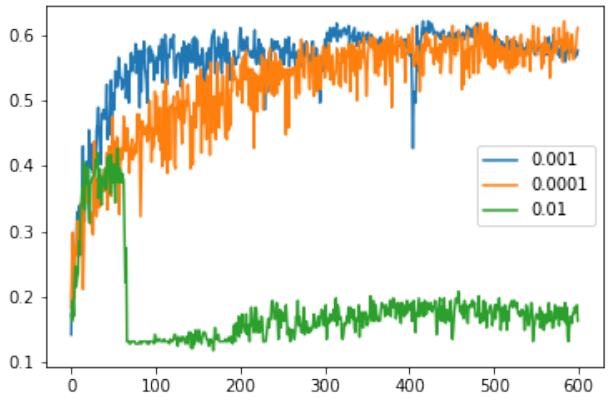

Fig. 9. This plot shows the validation accuracy of model with baseline structure and different learning rate. The involved rates are indicated on legend.

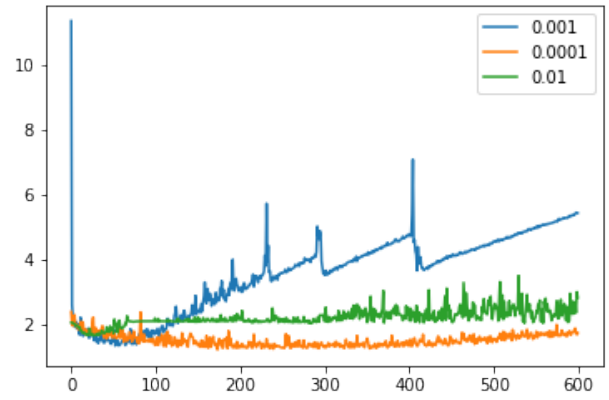

Fig. 10. This plot shows the validation loss of model with baseline structure and different learning rate. The involved rates are indicated on legend.

To choose the best reduction function, we set the minimum rate to 0.0001 and analyze the performance with different factors and patience. The results are shown in Fig. 13 and Fig. 14. From the validation accuracy plot Fig. 13, we can see that when the factor is 0.8, the model has higher accuracy. From the validation loss plot Fig. 14, it is easier to figure out that when the patience is 5, the model has the lower loss value. Therefore, for reduction, we choose 0.8 as the factor and 5 as the patience.

Let's put these methods together, and the results are shown in Fig. 15 and Fig. 16. It is easy to figure out that no matter we choose reduction or decay function, they have a similar performance. We choose the decay function to decrease our learning rate in our final model since it has relatively high accuracy and low loss value.

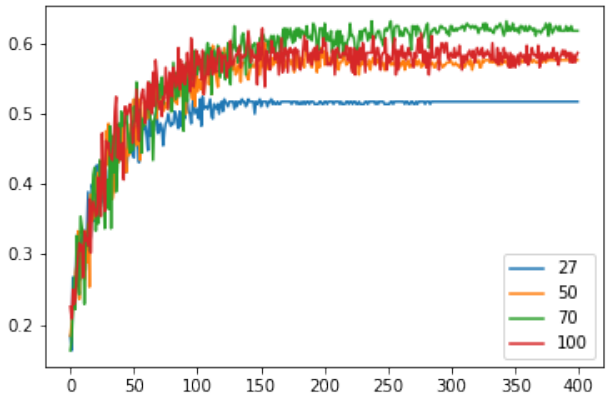

Fig. 11. This plot shows the validation accuracy of model with baseline structure and different decay functions. The involved decay parameters are indicated on legend.

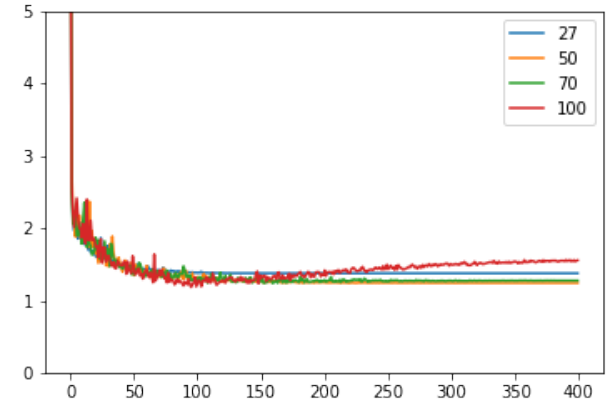

Fig. 12. This plot shows the validation loss of model with baseline structure and decay functions. The involved decay parameters are indicated on legend.

E. Final Model

We didn't talk about model architecture and layers in this project. Our final model simply uses the baseline architecture, Mel-frequency cepstral coefficients as the feature, noise as data augmentation technique and decay as learning rate schedule. The performance of the final model is shown in Fig. 17 and Fig. 18. We can see that the accuracy is not good enough but we believe it is expected using this kind of simple model, and the overfitting problem still exists. This model achieve 55.55% on our test set.

IV. Conclusion

To conclude, we use TensorFlow and Python to build a CNN-based model to fit the dataset found from Kaggle. We create the baseline model for comparison and analyze different parts of the model. Finally, we achieve a final model with 55.55% accuracy on our test set. This accuracy is terribly bad

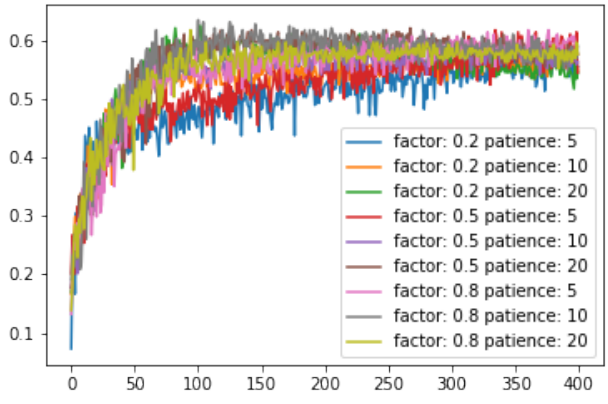

Fig. 13. This plot shows the validation accuracy of model with baseline structure and different reduction rate. The involved reduction parameters are indicated on legend.

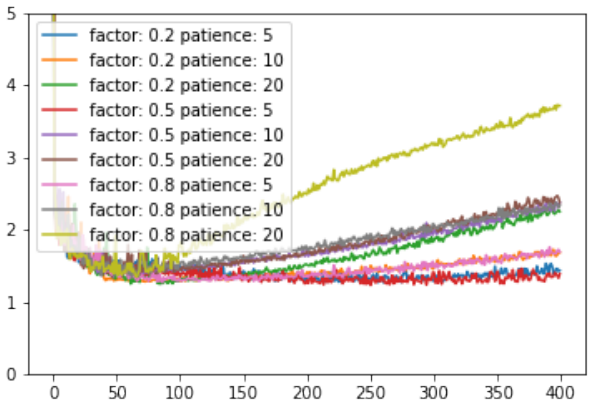

Fig. 14. This plot shows the validation loss of model with baseline structure and reduction rate. The involved reduction parameters are indicated on legend.

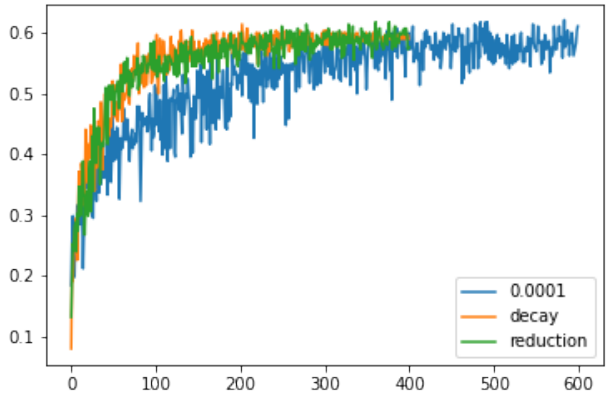

Fig. 15. This plot shows the precise comparison for decay and reduction's accuracy.

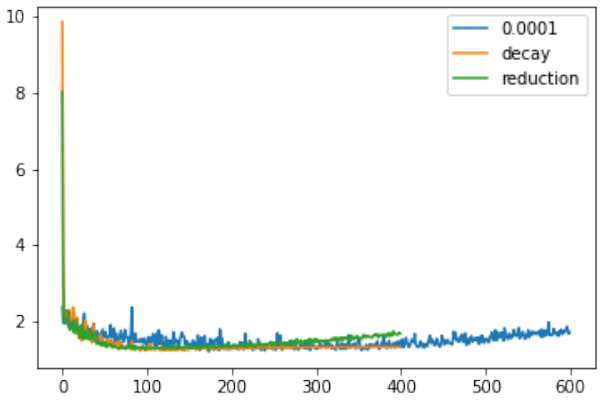

Fig. 16. This plot shows the precise comparison for decay and reduction's loss.

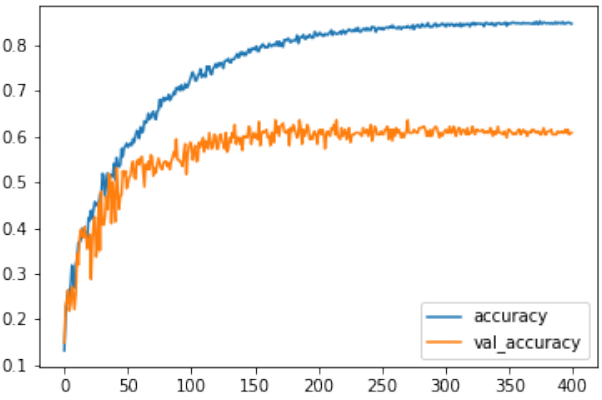

Fig. 17. This training accuracy and validation accuracy of our final model

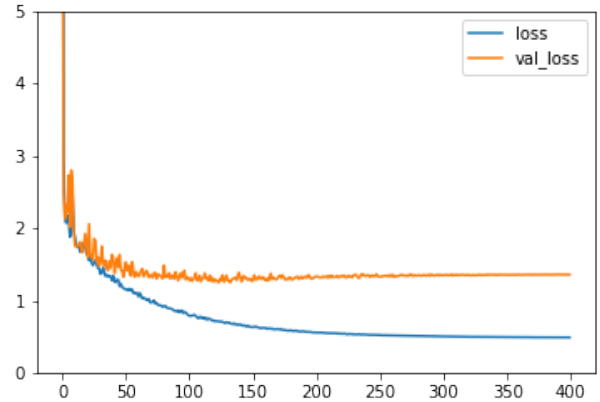

Fig. 18. This training loss and validation loss of our final model

because the architecture of our model is really simple. Therefore, there are some more works the need to be done in the future. For example, one could find some other features that are more suitable to the dataset to obtain a better performance of the result. A more sophisticated model architecture is also a way to improve this project's performance such as ResNet and MobileNet. One could even try to implement networks other than CNNs to see if they can have better accuracy.

ACKNOWLEDGMENT

We would like to express our gratitude and thanks to professor Boyu Wang for inspiring our interest in machine learning and providing such a well-structured and systematic course content of theoretical machine learning throughout the whole semester.

REFERENCES

[1] M. E. Ayadi, M. S. Kamel, F. Karray, "Survey on speech emotion recognition: Features, classification schemes, and databases," Pattern Recognition, vol. 44, no. 3, March., pp. 572-587, 2011.

[2] Livingstone SR, Russo FA (2018) The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS): A dynamic, multimodal set of facial and vocal expressions in North American English. PLoS ONE 13(5): e0196391. https://doi.org/10.1371/journal.pone.0196391.

[3] Kaggle.com. 2021. RAVDESS Emotional speech audio. [online] Available: https://www.kaggle.com/uwrfkaggler/ravdess-emotional-speech-audio [Accessed 22 December 2021].

[4] Ma, E., 2021. Data Augmentation for Audio. [online] Medium. Available: https://medium.com/@makcedward/data-augmentation-for-audio-76912b01fdf6 [Accessed 21 December 2021].

[5] L. Nanni, G. Maguolo, M. Paci, "Data augmentation approaches for improving animal audio classification", Ecological Informatics, vol. 57, May. 2020.

[6] MusicRadar. 2021. How to understand the basics of pitchshifting. [online] Available: https://www.musicradar.com/tuition/tech/how-to-understand-the-basics-of-pitchshifting-625517 [Accessed 21 December 2021].

[7] T. Giannakopoulos and A. Pikrakis, "Introduction to Audio Analysis," ScienceDirect, 2014. [Online]. Available: https://www.sciencedirect.com/book/9780080993881/introduction-to-audio-analysis. [Accessed: 22-Dec-2021].

[8] V. Velardo, "Mel-Frequency Cepstral Coefficients Explained Easily," YouTube, 05-Oct-2020. [Online]. Available: https://www.youtube.com/watch?v=4_SH2nfbQZ8. [Accessed: 22-Dec-2021].

[9] S. Lalitha, D. Geyasruti, R. Narayanan, Shravani M, "Emotion Detection Using MFCC and Cepstrum Features", Procedia Computer Science, vol. 70, pp. 20-35, 2015.

[10] L. Roberts, "Understanding the Mel Spectrogram," Medium, 14-Mar-2020. [Online]. Available: https://medium.com/analytics-vidhya/understanding-the-mel-spectrogram-fca2afa2ce53. [Accessed: 22-Dec-2021].

[11] J. Brownlee, 2021. Understand the Impact of Learning Rate on Neural Network Performance. [online] Machine Learning Mastery. Available: https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/. [Accessed 15 December 2021].

[12] "Understanding Spectrograms," iZotope, 11-Apr-2019. [Online]. Available: https://www.izotope.com/en/learn/understanding-spectrograms.html. [Accessed: 22-Dec-2021].

[13] "librosa.feature.chroma_stft," librosa.feature.chroma_stft - librosa 0.8.1 documentation. [Online]. Available: https://librosa.org/doc/main/generated/librosa.feature.chroma_stft.html. [Accessed: 22-Dec-2021].

[14] "librosa.feature.chroma_cens," librosa.feature.chroma_cens - librosa 0.8.1 documentation. [Online]. Available: https://librosa.org/doc/main/generated/librosa.feature.chroma_cens.html. [Accessed: 22-Dec-2021].

[15] Chroma Toolbox: Pitch, Chroma, CENS, CRP. [Online]. Available: http://resources.mpi-inf.mpg.de/MIR/chromatoolbox/. [Accessed: 22-Dec-2021].