A deep learning approach towards classifying emotion from audio data

Anurag Das

April 19, 2019

Abstract

In this project, a deep learning approach for emotion classification using speech data from different modalities is performed. A convolutional neural network(CNN) that captures discriminative information from audio features is trained to correctly classify the emotion labels. In addition to the CNN model, Long Short Term Memory(LSTM) networks are also experimented with for the classification task. Further experiments also include experimenting with different network architectures of each individual model, using regularization strategies such as dropout, etc. Highest classification accuracy of **100%** is reported on the test set. To ensure that the trained model performs well on an unknown audio sample, different from the samples used for training and testing, **audio samples from a completely separate dataset are collected and tested using the trained model**. To ensure that the results obtained are indeed true, the **Kaggle kernel**[1] **used for training the models is also made public**.

1 Introduction

Human emotions help individuals respond to different situations such as dealing with others in the best possible way and thereby avoiding conflicts or confrontations. Thus, understanding human emotions plays an important role in improved interaction amongst people. Automating this task, may help learners improve their social interaction skills and help them achieve their goals.

This project is an effort in the above direction. Neural Networks which learn to automatically classify emotions from spectral features are made use of. Accuracies reported on the test set for various neural network architectures, show that they perform very well on the test set, thereby correctly predicting emotions from audio samples. The rest of the report is organized as follows, Section 2 talks about the dataset, Section 3 talks about the modifications to the initial proposal, Section 4 discusses the features and models used, Section 5 talks about the hyper parameters for each model, Section 6 talks about various setbacks encountered over the course of the project and their solutions, Section 7 discusses the results, Section 8 discusses if the model actually works, Section 9 provides instructions on testing the trained networks, Section 10 lists some of the conclusions of the project.

[1] https://www.kaggle.com/dasanurag38/audio-emotion-recognition?scriptVersionId=13099888

2 Dataset Description

The RAVDESS Audio-Visual Database of Emotional Speech and Song (RAVDESS) [1] consists of 7356 files, totalling in 24.8 Gb in size. 24 actors record two versions of three different modalities(audio-only, audio-video, video-only) in speech and song formats. Each file was rated 10 times on emotional validity, intensity, and genuineness. Ratings were provided by 247 individuals who were characteristic of untrained adult research participants from North America. A further set of 72 participants provided test-retest data. High levels of emotional validity, interrater reliability, and test-retest intrarater reliability were reported. The dataset has the following labels:- (01 = neutral, 02 = calm, 03 = happy, 04 = sad, 05 = angry, 06 = fearful, 07 = disgust, 08 = surprised).
The distribution of the dataset is reported in the following table:-

Modality	Number of actors	Number of trials per actor	Number of files
Speech(Audio)	24	60	1440
Song(Audio)	23	44	1012
Speech(Audio-Video + Video-only)	24	60	2880
Song(Audio-Video + Video-only)	23	44	2024

In the performed experiments, audio is extracted from the video files, as the facial features from the video files are not used.

3 Modifications from the initial proposal

Several difficulties arose over the course of the project. As a result, changes were made to the initial proposal. These are discussed below:-

- The initial proposal talked about training models posteriorgram(PPG) features. Instead of training on posteriorgram features, MFCC [2] features were used for training the models. The is because the model responsible for training the neural network which outputs the posterior probabilities for a new speech file was trained using Kaldi, which in turn has been built using C/C++. The reasons for not using this Kaldi based approach is two fold. Firstly, installing and running Kaldi is a highly non-trivial task. Although, the Kaldi binaries which perform the PPG extraction are based on C++, the are have a large number of dependencies and operating them requires a fair level of expertise. Moreover, a correct Kaldi installation results in over 12 GB of memory consumption. The second reason is that each output posteriorgram file is of size 4 MB. The large file size is because posteriorgrams are of dimension $N \times 5800$, where N is the number of frames and 5800 is the number of triphone labels over which posterior probabilities are calculated. For 4000 audio files, this results in a feature set size of 16GB. This results in extremely high computational costs. Instead MFCCs are used which are of dimension $N \times 40$, where 40 is the number of MFCC coefficients. As a result, this feature set is much smaller in size in comparison to the PPG features.

- The initial proposal also talked about using the IEMOCAP dataset. The IEMOCAP dataset has multiple speakers included in the same recording and several frames do

not have a particular label associated with them. This increased the complexity of the problem and opting for another dataset much simpler in terms of labels was the preferred solution. As a result, the RAVDESS dataset was used as the replacement.

4 Features and Model Description

Mel frequency Cepstral Coefficients(MFCC) features were used for the experiments. MFCC have been shown to be an accurate representation of the spectra of speech signals and are thus highly discriminative in nature. 40 dimensional MFCCs are extracted for every frame with a 10 ms second shift between consecutive frames.
For the classification task, following four models have been experimented with. The first model is trained using LSTM layers, the remaining three use convolutional neural networks. From now onwards, **the model trained using LSTM will be denoted by Model_A, the remaining three models will be represented by Model_B, Model_C and Model_D**. The shapes of the layers of each model are presented in the following figures:-

Layer (type)	Output Shape	Param #
lstm_2 (LSTM)	(None, 128)	66560
dense_4 (Dense)	(None, 64)	8256
dropout_3 (Dropout)	(None, 64)	0
activation_4 (Activation)	(None, 64)	0
dense_5 (Dense)	(None, 32)	2080
dropout_4 (Dropout)	(None, 32)	0
activation_5 (Activation)	(None, 32)	0
dense_6 (Dense)	(None, 8)	264
activation_6 (Activation)	(None, 8)	0

Total params: 77,160
Trainable params: 77,160
Non-trainable params: 0

Figure 1: Model summary for Model_A

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_27 (Conv1D)           (None, 38, 8)             32
_________________________________________________________________
activation_47 (Activation)   (None, 38, 8)             0
_________________________________________________________________
conv1d_28 (Conv1D)           (None, 36, 16)            400
_________________________________________________________________
activation_48 (Activation)   (None, 36, 16)            0
_________________________________________________________________
max_pooling1d_7 (MaxPooling1 (None, 18, 16)            0
_________________________________________________________________
conv1d_29 (Conv1D)           (None, 16, 32)            1568
_________________________________________________________________
activation_49 (Activation)   (None, 16, 32)            0
_________________________________________________________________
conv1d_30 (Conv1D)           (None, 14, 16)            1552
_________________________________________________________________
activation_50 (Activation)   (None, 14, 16)            0
_________________________________________________________________
max_pooling1d_8 (MaxPooling1 (None, 7, 16)             0
_________________________________________________________________
flatten_8 (Flatten)          (None, 112)               0
_________________________________________________________________
dense_17 (Dense)             (None, 8)                 904
_________________________________________________________________
activation_51 (Activation)   (None, 8)                 0
=================================================================
Total params: 4,456
Trainable params: 4,456
Non-trainable params: 0
_________________________________________________________________
```

Figure 2: Model summary for Model_B

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_5 (Conv1D)            (None, 40, 8)             48
_________________________________________________________________
activation_17 (Activation)   (None, 40, 8)             0
_________________________________________________________________
conv1d_6 (Conv1D)            (None, 40, 16)            656
_________________________________________________________________
activation_18 (Activation)   (None, 40, 16)            0
_________________________________________________________________
dropout_7 (Dropout)          (None, 40, 16)            0
_________________________________________________________________
max_pooling1d_2 (MaxPooling1 (None, 5, 16)             0
_________________________________________________________________
conv1d_7 (Conv1D)            (None, 5, 32)             2592
_________________________________________________________________
activation_19 (Activation)   (None, 5, 32)             0
_________________________________________________________________
dropout_8 (Dropout)          (None, 5, 32)             0
_________________________________________________________________
conv1d_8 (Conv1D)            (None, 5, 16)             2576
_________________________________________________________________
activation_20 (Activation)   (None, 5, 16)             0
_________________________________________________________________
flatten_3 (Flatten)          (None, 80)                0
_________________________________________________________________
dense_9 (Dense)              (None, 8)                 648
_________________________________________________________________
activation_21 (Activation)   (None, 8)                 0
=================================================================
Total params: 6,520
Trainable params: 6,520
Non-trainable params: 0
_________________________________________________________________
```

Figure 3: Model summary for Model_C

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_9 (Conv1D)            (None, 40, 128)           768
_________________________________________________________________
activation_22 (Activation)   (None, 40, 128)           0
_________________________________________________________________
dropout_9 (Dropout)          (None, 40, 128)           0
_________________________________________________________________
max_pooling1d_3 (MaxPooling1 (None, 5, 128)            0
_________________________________________________________________
conv1d_10 (Conv1D)           (None, 5, 128)            82048
_________________________________________________________________
activation_23 (Activation)   (None, 5, 128)            0
_________________________________________________________________
dropout_10 (Dropout)         (None, 5, 128)            0
_________________________________________________________________
flatten_4 (Flatten)          (None, 640)               0
_________________________________________________________________
dense_10 (Dense)             (None, 8)                 5128
_________________________________________________________________
activation_24 (Activation)   (None, 8)                 0
=================================================================
Total params: 87,944
Trainable params: 87,944
Non-trainable params: 0
_________________________________________________________________
```

Figure 4: Model summary for Model_D

5 Hyper-parameters

Table 1: Hyper parameters for Model_A

Batch Size	1
Epochs	100
Dropout	0.4

Table 2: Hyper parameters for Model_B

Batch Size	1
Epochs	100

Table 3: Hyper parameters for Model_C

Batch Size	1
Epochs	100
Dropout	0.1-0.2

Table 4: Hyper parameters for Model_D

Batch Size	1
Epochs	100
Dropout	0.1

From the above figures, it is observed that every model is trained using a batch_size of 1. **Using a higher batch_size did not lead to further improvements. That's why, only a batch_size of 1 was used for training all the models**. Model_B is similar to Model_C except the former does not have any Dropout layer. Model_D only has three hidden layers while Model_B and Model_C have five hidden layers.

6 Setbacks and Solutions

The RAVDESS dataset consists of data from both speech and song modalities recorded in only audio, only video and audio-video(AV) formats. To train the model, initially only speech data was considered. This is because song data is a different modality and using song data while training may bias the model such that the model may performs poorly on a test case, different from the dataset under consideration. However, this also means that there is very limited speech data(1400 files) available for training, which may lead to overfitting. Unfortunately, this is indeed the case as shown in the following figures:-

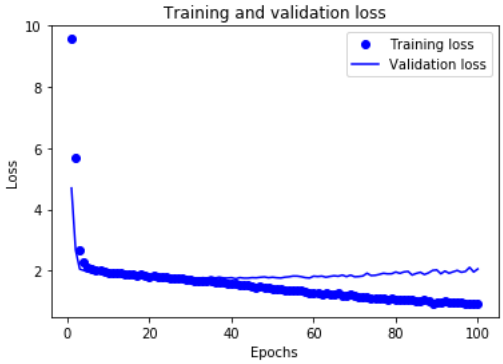


Figure 5: train and validation loss vs number of epochs on baseline model

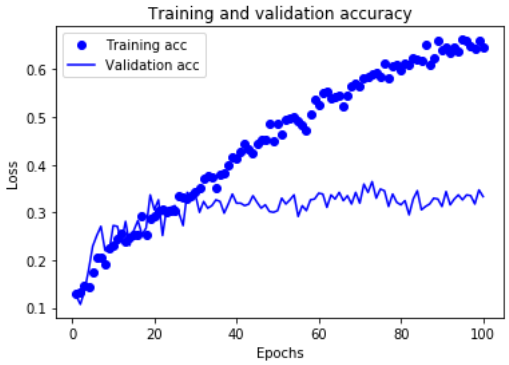


Figure 6: train and validation accuracy vs number of epochs on baseline model

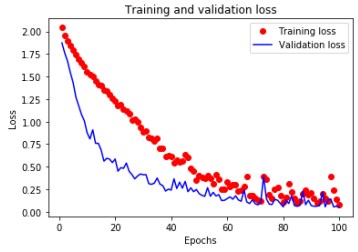
Clearly, the model easily overfits on the training data and does poorly on the test data as shown by the respective loss functions. **The above model was trained using the architecture of Model_C for 100 epochs. From now onwards, this model will referred to as the baseline model**.

To overcome this, the data from the song modality is used. This increases the total number of samples available for training(4000). This drastically improves the performance on the test set. The performance on the test set and the data augmentation procedure is highlighted further in the results section.

7 Results

Out of the available 4904 files, we use 3923 files for training, 490 files for validation and 491 files for testing. To ensure fair comparison between all models, each model is trained for 100 epochs.

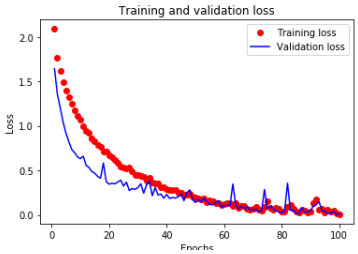
Surprisingly, augmenting the data by using data from the song modality results in drastic improvement in performance as shown below:-



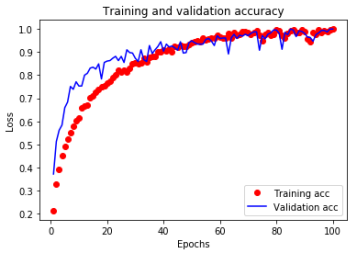
(a) validation and training loss for model_A

(b) validation and training accuracy for model_A

Figure 7: Model_A

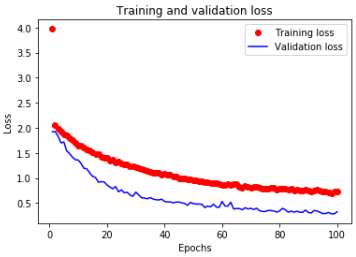


(a) validation and training loss for Model_B

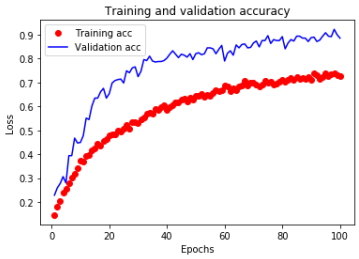


(b) validation and training accuracy for Model_B

Figure 8: Model_B

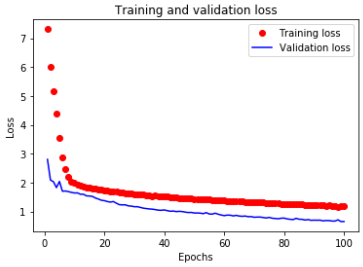


(a) validation and training loss for Model_C

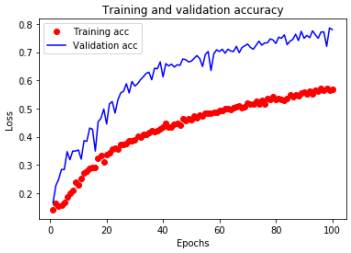


(b) validation and training accuracy for Model_C

Figure 9: Model_C



(a) validation and training loss for Model_D



(b) validation and training accuracy for Model_D

Figure 10: Model_D

The loss and accuracies for each individual model is shown below:-

Model	Training loss	Training accuracy	Validation loss	Validation accuracy	Test accuracy
Model A	0.0824	0.9768	0.0491	0.9761	0.9857
Model B	0.00103	1.00	0.0064	1.00	1.0
Model C	0.6993	0.7466	0.2287	0.9265	0.9389
Model D	1.1739	0.5695	0.06565	0.7796	0.794

100% accuracy is obtained by one out of the four models. Again, **to make sure that these results are indeed true, users are welcome to check out the link to the Kaggle Kernel provided previously**.

8 Does the model actually work?

The model performs well on the test set chosen from the RAVDESS dataset. However, an interesting question is whether the model works well on an unknown dataset. For this problem, audio data from SAVEE dataset [3] are tested. The SAVEE dataset consists of Audio-Visual data from British English speakers. It has the following emotions:- calm, anger, disgust, fear, happiness, sadness, surprise, neutral. The RAVDESS dataset does not have the common emotion. As a result, test samples labelled "common" are ignored. To test this, some samples fron the SAVEE dataset are provided in the github repo.

9 Instructions on testing trained DNN and using the GUI

9.1 Installation Dependencies

- Tkinter
- Playsound
- Numpy
- Tensorflow
- keras
- librosa
- seaborn
- matplotlib

9.2 Execution

To run the demo, run the following commands:-

- pip install requirements.txt
- python gui.py

9.3 Annotated code

Annotated code is provided in the Kaggle Kernel[2]. It is also availbale as an IPython Notebook titled emotion_recog.ipynb, provided in the Github repo.

9.4 Github Repo

Link to the Github Repo is here.

9.5 Video Link

Video link to the demo is here.

10 Conclusion

Several deep neural architectures are investigated for the audio classification task. Although all models perform competitively on the training set, they perform poorly on the test set. Data augmentation, however results in improved performance on the test set comparable to the train set.

References

[1] Livingstone, Steven R and Russo, Frank A, *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, 2018)

[2] Molau, Sirko and Pitz, Michael and Schluter, Ralf and Ney, Hermann, *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings*, 2001

[3] Jackson, P and Haq, S, *Surrey audio-visual expressed emotion (savee) database*, 2014.

[2]https://www.kaggle.com/dasanurag38/audio-emotion-recognition?scriptVersionId=13099888