Applications of Deep Learning Techniques in Speech Emotion Recognition

Madhu Sree Sankaran (mss1g22@soton.ac.uk, ID: 33891699)
Saran Raj Rajasekaran (srr1g21@soton.ac.uk, ID: 33246807)
Farhana Akter Bina (fab1n22@soton.ac.uk, ID: 33617945)
Ji Hwan Kim (jhk1c21@soton.ac.uk, ID: 33432368)

Abstract

The ability to comprehend and communicate through language is a highly valuable skill possessed by humans. As emotions play a critical role in communication, it is a very challenging task to recognize because of the subjective nature of human moods. Deep learning architectures have a wide range of applications in handling and extracting complicated representations from raw data in various domains which also inlcudes speech and audio processing, recognizing and classifying them. This report proposes the applications of different deep learning techniques in speech emotion classification to categorize the emotional content of audio files, including Convolutional Neural Networks (CNNs), Long Short-Term Memory (LSTM) Networks, Artificial Neural Networks (ANNs), Recurrent Neural Networks (RNNs), and Gated Recurrent Unit (GRU) etc. To ensure that the audio data is as useful as possible for future purposes, models that incorporate Mel-frequency cepstral coefficients (MFCCs) were developed.The performance of these models will be examined based on their accuracy, robustness, and scalability.

Keywords:
Speech Emotions Classification, Deep Learning Models, Melfrequency Cepstral Coefficients

1. Introduction

Speech Emotion Recognition (SER) is a field of research that focuses on developing algorithms and models capable of automatically recognizing the emotional state of speakers from their speech signals. The emotional state of a speaker can be determined based on various features. Detecting and classifying these features accurately is a challenging task that requires advanced machine learning techniques and deep learning architectures.

Data Source: Kaggle: //www.kaggle.com/

Dataset: Toronto Emotional Speech Set (TESS)

Dataset Description: Toronto emotional speech set (TESS) dataset is a collection of 2800 audio recordings that represent seven different states of excitements: happy, sad, angry, surprised, afraid, disgusted, and neural. Two female performers actresses used the transporter statement "Give the signal" speaking 200 target words in a carrier phrase. The dataset is organized in folders containing audio files of each actress's emotions, and each audio file is in WAV format.

Deep Learning Technologies:
i. Convolutional Neural Networks (CNNs);
ii. Long Short-Term Memory (LSTM) Networks;
iii. Recurrent Neural Networks (RNNs);
iv. Artificial Neural Networks (ANNs); and
v. Gated Recurrent Unit (GRU).

2. Implementation of the Architectures

2.1. CNN Architecture

2.1.1 Hyperparameters

The hyperparameters are tuned for model optimization. 'ample rate' is the customary audio processing value. 'duration' is the length of the tracks. 'mfcc' is the common value for extracting the coefficients. 'n_mfcc' captures the number of coefficients to be extracted.

2.1.2 Pre-Processing

An empty array is created to store the path, duration, and the different categories of the type of emotion label. An iterating loop is created to extract the emotion type using the split condition '_' and mapped with the semantic labels. 'librosa.load' loads the entire file path. 'librosa.feature.mfcc' extracts the total number of mfcc coefficients to be extracted. The emotion short forms are replaced with full-length words. (eg ps – surprise). A data frame is created with an audio path, a type of emotion(label), the duration of each audio track, and the dataset. 'df.info' performs exploratory data analysis by providing a piece of brief information about the generated data frame. The extracted mfccs are applied to each audio path, converted to an array, and the dimension is changed to build the CNN architecture.

2.1.3 CNN Model

The 'train-split-test' from the TensorFlow (keras) library splits the data into train, validation, and test sets. The scaling is performed by computing the mean and standard deviation of the training set. The layers are: a) 'CNN', b) 'Dense' with 'ReLU' as the activation function that activates the threshold that sets the '+ve' o/ps unchanged, and '-ve' to zero and 'softmax' in the last layer converts o/ps into PDF to map with the classes, c) The 'Dropout' layer is set to 20% where the inputs are dropped to avoid 'Overfitting'(regularization). The maximum value is selected to produce a summarized output by downsizing the image using the 'Max Pool'. Batch Normalization acts as an extension of 'Gradient Descent' where

the learning rate is in large steps if it moves towards the gradient and is smaller in the opposite direction. The Adam optimizer (Adaptive Learning) complies with the 'sparse categorical cross entropy' loss function and is evaluated with 'Accuracy'.

2.1.4 Evaluation

In validation and test sets, the accuracy is 99% and 99% with the loss dropping to 0.002 and 0.0 for 100 epochs [Figure 1]. In comparison, it performs accurately on the test set. We can see in the figure as well as the validation set 'Accuracy' and 'Loss' metrics, the figure is showing an increasing trend for the accuracy and decreasing trend for the validation loss. For predicting and testing on different samples, the labels are similar indicating the quality performance of the model.

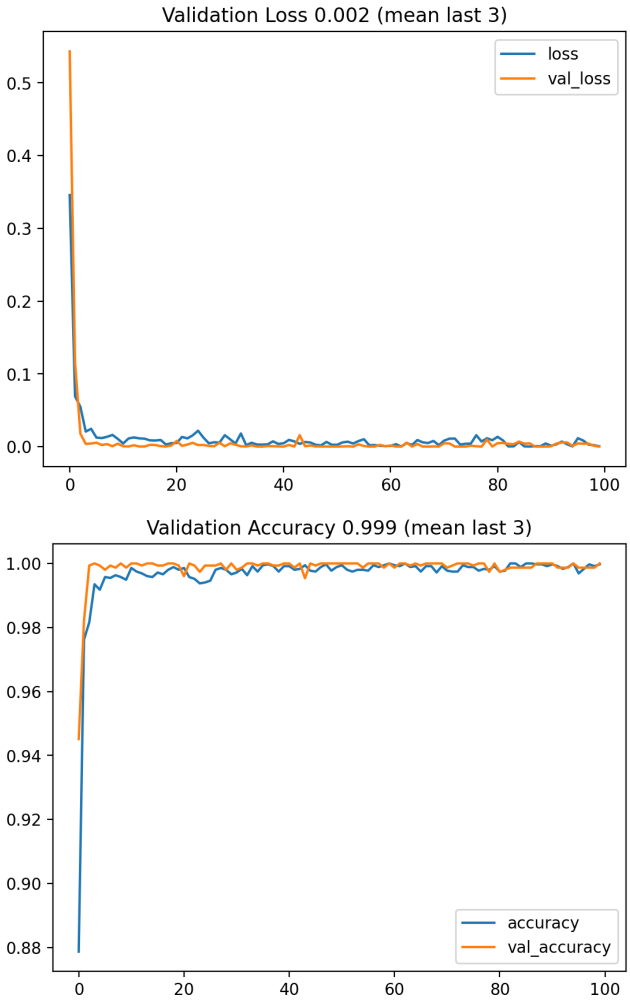

Figure 1. The loss and accuracy of the CNN Architecture

2.2. LSTM Architecture

2.2.1 Pre-Processing

The implementation of LSTM starts by importing essential libraries, such as torch, librosa, os, and numpy. Next, the AudioDataset class is created, which requires the path to a folder containing audio files as its argument. This class retrieves the names of the audio files and their corresponding labels from the folder and preprocesses each audio file by computing the Mel-frequency cepstral coefficients (MFCCs). The getitem method of the AudioDataset class returns a tensor of MFCCs and a tensor of labels for a given index. The len method returns the number of audio files in the dataset.

Following that, the code defines a function named pad sequence, which receives a batch of audio files as input, pads all audio files to an equal length, and transposes the tensor to conform to PyTorch conventions.

The code proceeds to instantiate the AudioDataset class and randomly splits it into training, validation, and testing sets using the random split function. To iterate over the data during training, DataLoader instances are created for each dataset.

2.2.2 LSTM Model

The AudioLSTM class is defined next, which implements an LSTM neural network with a fully connected layer. The forward method takes a tensor of MFCCs as input and returns a tensor of predicted labels. The code then defines the device to be used for training, initializes an instance of the AudioLSTM class, and sets the loss function and optimizer. The model is trained for a fixed number of epochs, with the training progress printed at regular intervals. The training loss and accuracy are recorded for each epoch, and the model is evaluated on the validation set after each epoch. The training loop terminates after the specified number of epochs have been completed.

2.3. Evaluation

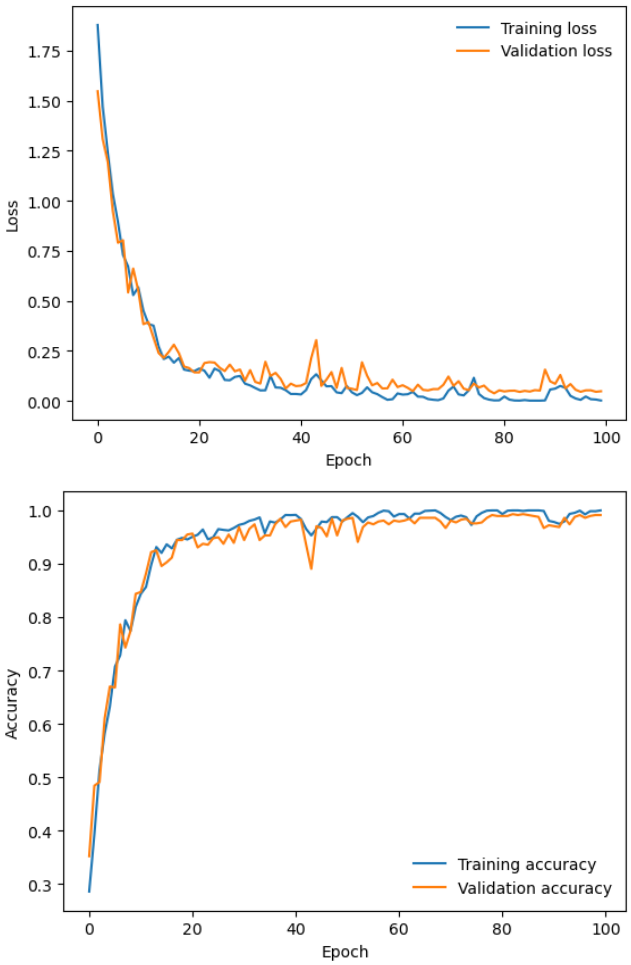

Figure 2. The loss and accuracy of the LSTM Architecture

The above Figure 2 depicts that model was trained for 100 epochs and reached a very high accuracy of 99% in training

and 98% in validation. During training, training and validation loss decreased where as training and validation accuracy has increased over the 100 epochs and testing accuracy achieved 98% and training loss is 0.0012. In first epoch, its started with 26% of training and 41% of validation and we can see increasing trend over the 100 epochs. In loss graph, in initial epoch, it started training loss and validation loss are 1.84 and 1.43 respectively. Over 100 epochs, its gradually decreased to 0.015 and 0.04 for both training and validatio. Finally as we test overall accuracy, we got 98% and found that there are only 3 sample which is predicted wrong label.

2.4. ANN Architecture

2.4.1 Pre-Processing

Implementation of ANN begins by importing the necessary libraries and loading the TESS dataset. The audio files are then coverted and normalized. The MFCC features are extracted using the librosa library. The extracted features and their corresponding labels are stored in a pandas data frame. The features and labels are then converted to numpy arrays, and the labels are one-hot encoded using the to categorical function from keras. The data is split into training and testing sets using train-test-split from sklearn.

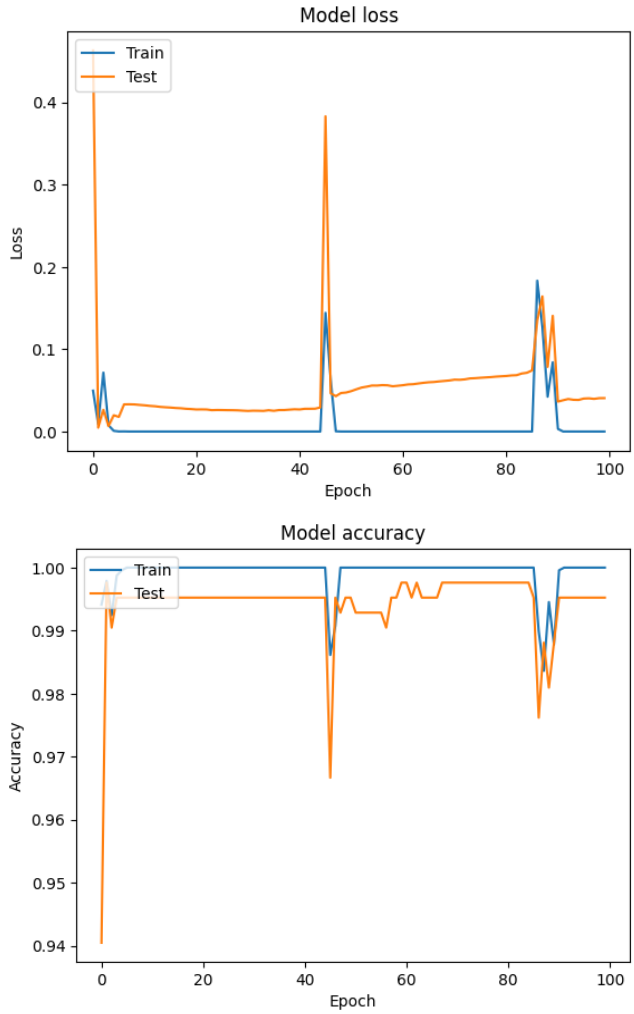

Figure 3. The loss and accuracy of the ANN Architecture

2.4.2 ANN Model

A sequential deep neural network is built using keras, consisting of four dense layers with 256 units each, and an output layer with 7 units for the 7 different emotional states in the dataset. The model is compiled using the Adam optimizer and categorical cross-entropy loss, with accuracy, precision, recall, and AUC as the evaluation metrics. It is trained using the fit function of keras, with 100 epochs and a batch size of 4. The training and testing accuracy, precision, recall, and AUC are evaluated using the evaluate function of keras.

2.4.3 Evaluation

The Figure 3 demostrates that model was trained for 100 epochs and reached a very high accuracy of in training and 99% in validation. During training, the training and validation loss decreased whereas training and validation accuracy has increased over the 100 epochs and testing accuracy achieved 99% and training loss is 0.0019. The initial epoch starts with validation loss: 0.1251, accuracy: 0.9857 and precision: 0.9857. In the final 100 epoch, the validation loss reached to 0.0115, validation accuracy to 0.9976 and validation precision upto 0.9976. In comaprison we can say that, the ANN model validation loss decreases and the accuracy increases in the final epoch.

2.5. GRU Architecture

2.5.1 GRU

As one of the recurrent units in recurrent neural networks, the gated recurrent unit (GRU), is frequently compared to a long short-term memory (LSTM). They are both rooted in the RNN but are devised in order to prevent the gradient vanishing problem of traditional RNN using the tanh unit [Hochreiter, S. ,1997]. GRU is specifically distinguishable from LSTM in terms of the use of its gate. The LSTM uses an additional memory cell and three gates to propagate signals, slowing down the model training. On the other hand, the GRU uses only two gates, which combines the two gates of LSTM. As a result of this difference, the GRU shows a faster learning speed than LSTM without losing its performance [Chung, J., 2014].

2.5.2 GRU Model

The GRU architecture used in this coursework has layers of a GRU layer and two fully connected layers with a rectified linear unit (ReLU) activation function. The GRU layer implemented in this architecture consists of 128 hidden states and two stacked GRU. Subsequently, after passing the ReLU activation function, it is connected to one of the fully connected layers, which has 256 output features. Furthermore, it passes another ReLU activation and encounters the last fully connected layers with seven output features, which is the number of classes of emotions. Then finally, a softmax function at the end of this architecture is applied to get the final result. Moreover, this architecture uses Adam as its optimizer and cross entropy as its loss function, which is widely used for multi-class classification problems.

2.5.3 Evaluation

This GRU model used the dataset pre-processed above. The model was trained for 100 epochs with 32 batch sizes. As a result, the model shows around 98% accuracy for the validation dataset, and its accuracy for the test dataset gives a similar result of 99.3%. It actually reaches its best performance

around the 15th epoch already. Although the high score of its accuracy, since the result in terms of the test dataset is slightly better than the training and validation dataset, the model does not seem to be overfitted.

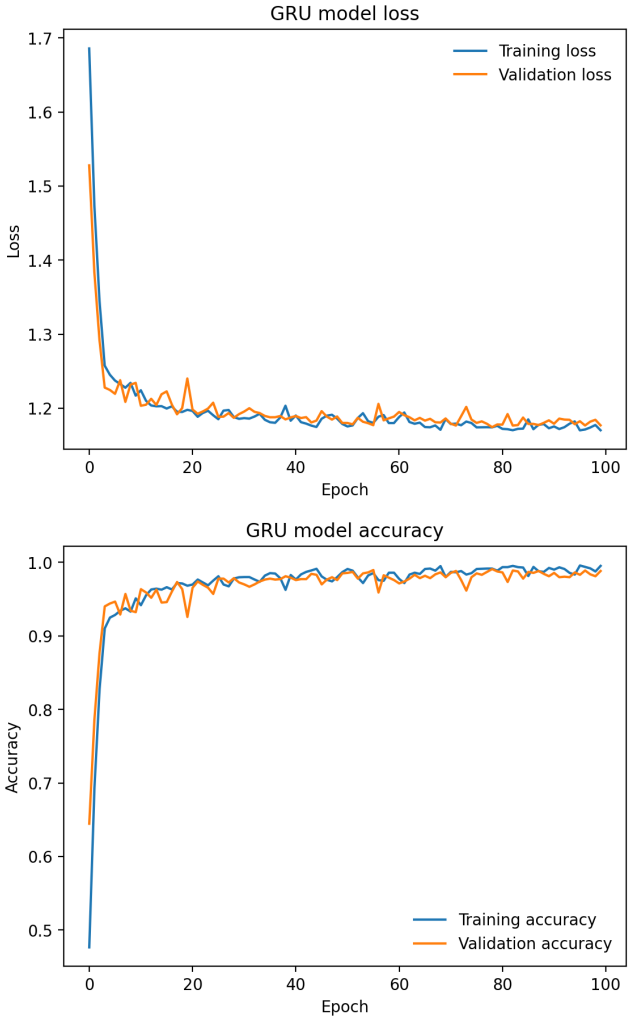

Figure 4. The loss and accuracy of the GRU Architecture

3. Results Discussion

Model	Loss	Accuracy
CNN	0.0020	0.9980
RNN-LSTM	0.0012	0.9893
ANN	0.0019	0.9976
GRU	1.1745	0.9929

Table 1. 'Loss' and 'Accuracy' for each model

Based on the provided model loss and accuracy values, it appears that the RNN-LSTM and ANN models performed better than the CNN and GRU models in the task of speech emotion recognition.

The RNN-LSTM model achieved the lowest model loss of 0.0012, indicating that it was able to minimize the difference between its predicted and actual output. Additionally, the model achieved an accuracy of 0.9893, meaning that it correctly classified 98.93% of the samples. These results suggest that the RNN-LSTM model was able to learn the patterns in the TESS dataset well and make accurate predictions.

The ANN model also performed well, achieving a model loss of 0.0019 and an accuracy of 0.9976. The model loss was slightly higher than that of the RNN-LSTM model, but the accuracy was higher. This suggests that the ANN model was able to generalize better and make fewer incorrect predictions.

The CNN model achieved the lowest model loss of 0.0020, which is still a very good performance, but its accuracy of 0.9980 suggests that it may have overfit the data and made more incorrect predictions on new data. However, without additional evaluation metrics, it is difficult to draw a definitive conclusion about the performance of the CNN model.

The GRU model achieved a relatively high model loss of 1.1745 and an accuracy of 0.9929. These results suggest that the model struggled to learn the patterns in the TESS dataset and made more incorrect predictions compared to the other models.

4. Summary and Conclusion

In summary, based on the provided model loss and accuracy values, the RNN-LSTM and ANN models appear to have performed better than the CNN and GRU models in the task of speech emotion recognition using the TESS dataset. The optimal hyperparameters may vary depending on the specific task and dataset, and hyperparameter tuning can be an iterative process that requires experimentation and evaluation of the model's performance.

In conclusion, we can conlude that, deep learning techniques have shown promising results in speech emotion recognition on the TESS dataset, as demonstrated by the CNN, RNN-LSTM, ANN, and GRU models discussed earlier. However, it is essential to consider additional evaluation metrics, perform cross-validation and testing on new data to draw definitive conclusions about the performance of each model.

5. References

Hochreiter, S. and Schmidhuber, J., (1997). Long short-term memory. Neural computation, 9(8), pp.1735-1780.

Chung, J., Gulcehre, C., Cho, K. and Bengio, Y., (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555.

Kumbar, Bhavana, (2022). A Comparative Study on Adversarial Attacks and Defense Mechanisms. Institute of Electrical and Electronics Engineers.

Pratistha, K.M., Kumari, A. and Devale, P., (2021). Speech Emotion Recognition in Python Using Deep Learning. Volume 9, Issue 8, doi: 10.15680/IJIRCCE.2021.0908019.

Dolka, H., Xavier V. M, A. and Juliet, S. (2021). Speech Emotion Recognition Using ANN on MFCC Features. 3rd International Conference on Signal Processing and Communication (ICPSC), Coimbatore, India, pp. 431-435, doi: 10.1109/ICSPC51351.2021.9451810.

Imambi, S., Prakash, K.B., Kanagachidambaresan, G.R. (2021). PyTorch, Programming with TensorFlow. EAI/Springer Innovations in Communication and Computing. Springer, Cham. pp 87–104