SAPIENZA
UNIVERSITÀ DI ROMA

Artificial Intelligence and Robotics

Neural Networks

2020 April

Nijat Mursali | 1919669

Yunus Emre Darici | 1900161

Speech Emotion Classification Using Attention-Based LSTM

Rome, Italy

Table of Contents

Abstract

Automatic speech emotion recognition has been a research hotspot in the field of human-computer interaction over the past decade. However, due to the lack of research on the inherent temporal relationship of the speech waveform, the current recognition accuracy needs improvement. To make full use of the difference of emotional saturation between time frames, a novel method is proposed for speech recognition using frame-level speech features combined with attention-based Long Short-term Memory (LSTM) recurrent neural networks. Framelevel speech features were extracted from waveforms to replace traditional statistical features, which could preserve the timing relations in the original speech through the sequence of frames.

This project put forward a method for speech recognition and speech emotion recognition based on LSTM. The fundamental reason of using this model is its ability to get the better results. First of all, we have extracted the model from the dataset that professors suggested and then we have applied LSTM as a statistical classifier. The results of the experiment show that the proposed method is more effective than others.

Introduction

In today's world, speech recognition and especially speech emotion recognition is getting more comprehensive research projects which crosswise uses psychology, pattern recognition, speech signal processing and artificial intelligence.

Sequence prediction problems have been around for a long time. They are considered as one of the hardest problems to solve in the data science industry. These include a wide range of problems; from predicting sales to finding patterns in stock markets' data, from understanding movie plots to recognizing your way of speech, from language translations to predicting your next word on your phone's keyboard.

With the recent breakthroughs that have been happening in data science, it is found that for almost all of these sequence prediction problems, Long short Term Memory networks(LSTMs) have been observed as the most effective solution.

LSTMs have an edge over conventional feed-forward neural networks and RNN in many ways. This is because of their property of selectively remembering patterns for long durations of time. The purpose of this article is to explain LSTM and enable you to use it in real life problems.

Models

LSTM stands for Long short-term memory. They are a special kind of Neural Network called Recurrent Neural Networks. Neural Networks is a machine learning technique where you stack up layers containing nodes. Neural Networks is a very powerful technique and is used for image recognition and many other applications. One of the limitations is that there is no memory associated with the model. Which is a problem for sequential data, like text or time series.Also our project aim is to perform a model which has best results as lstm on our dataset.

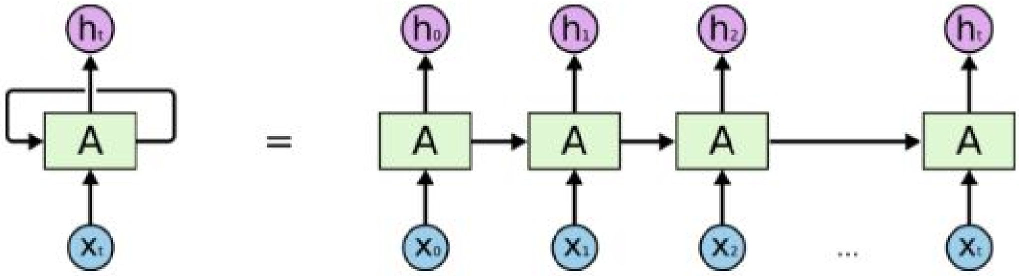
RNN addresses that issue by including a feedback look which serves as a kind of memory. So the past inputs to the model leave a footprint. LSTM extends that idea and by creating both a short-term and a long-term memory component.

Recurrent Neural Network (RNN)

Recurrent Neural Network is a generalization of a feedforward neural network that has an internal memory. RNN is recurrent in nature as it performs the same function for every input of data while the output of the current input depends on the past one computation. After producing the output, it is copied and sent back into the recurrent network. For making a decision, it considers the current input and the output that it has learned from the previous input.

Unlike feedforward neural networks, RNNs can use their internal state (memory) to process sequences of inputs. This makes them applicable to tasks such as unsegmented, connected handwriting recognition or speech recognition. In other neural networks, all the inputs are

Independent of each other. But in RNN, all the inputs are related to each other.



An unrolled recurrent neural network.

First, it takes the X(0) from the sequence of input and then it outputs h(0) which together with X(1) is the input for the next step. So, the h(0) and X(1) is the input for the next step. Similarly, h(1) from the next is the input with X(2) for the next step and so on. This way, it keeps remembering the context while training.

The formula for the current state is ;

$$h_t = f(h_{t-1}, x_t)$$

Applying Activation Function:

$$h_t = tanh(W_{hh}h_{t-1} + W_{xh}x_t$$

W is weight, h is the single hidden vector, W_{hh} is the weight at previous hidden state, W_{hx} is the weight at current input state, tanh is the activation function, that implements a Non-linearity that squashes the activations to the range[-1.1]

Output:

$$y_t = W_{hy}h_t$$

First, it takes the X(0) from the sequence of input and then it outputs h(0) which together with X(1) is the input for the next step. So, the h(0) and X(1) is the input for the next step.

Similarly, h(1) from the next is the input with X(2) for the next step and so on. This way, it keeps remembering the context while training.

y_t is the *output state*. W_{hy} is the *weight at the output state.*

Advantages of Recurrent Neural Network

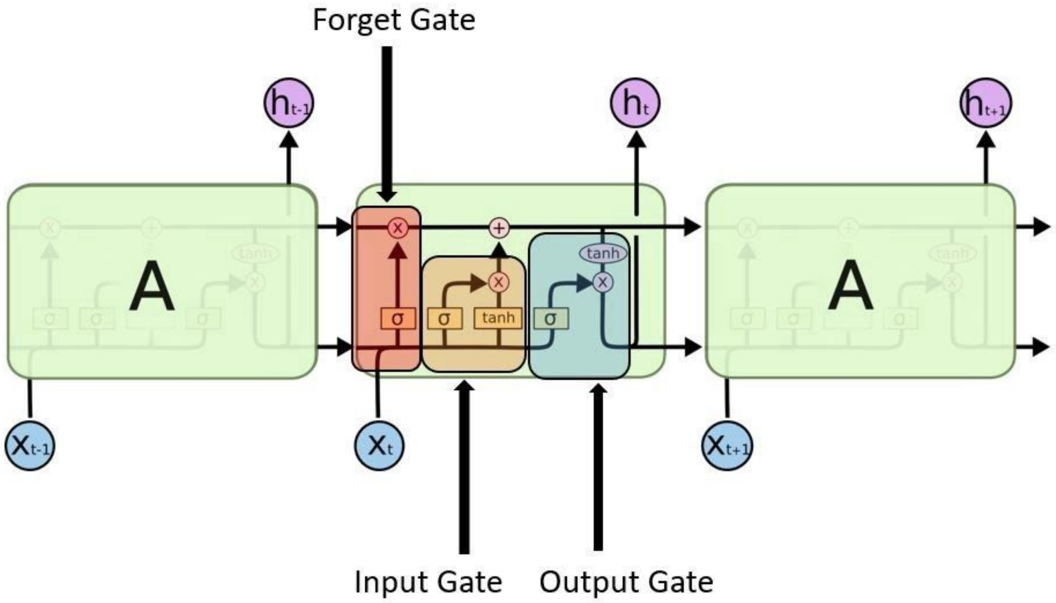
1. **RNN** can model sequence of data so that each sample can be assumed to be dependent on previous ones
2. Recurrent neural networks are even used with convolutional layers to extend the effective pixel neighbourhood.

Disadvantages of Recurrent Neural Network

1. Gradient vanishing and exploding problems.
2. Training an RNN is a very difficult task.
3. It cannot process very long sequences if using tanh or relu as an activation function.

Long Short Term Memory (LSTM)

Long Short-Term Memory (LSTM) networks are a modified version of recurrent neural networks, which makes it easier to remember past data in memory. The vanishing gradient problem of RNN is resolved here. LSTM is well-suited to classify, process and predict time series given time lags of unknown duration. It trains the model by using back-propagation. In an LSTM network, three gates are present:



Input gate — discover which value from input should be used to modify the memory. Sigmoid function decides which values to let through 0,1. and tanh function gives weightage to the values which are passed deciding their level of importance ranging from -1 to 1.

$$i_t = \sigma(W_i * [h_{t-1}, x_t] + b_i)$$

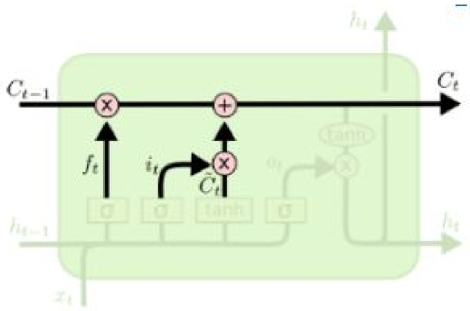
$$\acute{C} = tanh(W_C * [h_{t-1} , x_t] + b_C)$$

Forget gate — discover what details to be discarded from the block. It is decided by the sigmoid function. it looks at the previous state h_{t-1} and the content input x_t and outputs a number between 0(*omit this*)and 1(*keep this*)for each number in the cell state C_{t-1} .

$$f_t = \sigma(W_f * [h_{t-1},x_t] + b_f)$$

Cell State - Acts as a highway that transports relative information along the sequence chain.

Next, the cell state is calculated. First, the cell state is multiplied by f(result of step 1) which is then added to i(result of step 2), resulting in the new cell state(c).



$$C_f = f_t * C_{t-1} + i_t * \acute{C}_t$$

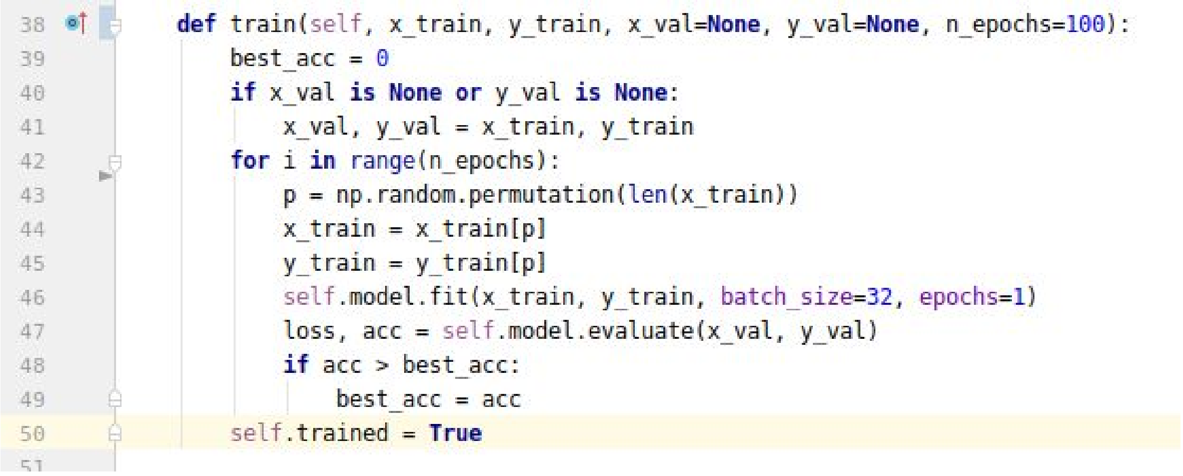
Output gate — the input and the memory of the block is used to decide the output. Sigmoid function decides which values to let through 0,1. and tanh function gives weightage to the values which are passed deciding their level of importance ranging from-1 to 1 and multiplied with output of Sigmoid.

$$o_t = \sigma(W_o * [h_{t-1}, x_t] + b_o)$$
$$h_t = o_t * tanh(C_t)$$

Development

For the development of our project, we needed to take some important steps. First of all, we needed to collect the dataset. For our project, it was mentioned to use CASIA emotional speech corpus. This corpus contains two hours of spontaneous emotional segments extracted from 219 speakers from films, TV plays and talk shows. The number of the speakers of the corpus makes this database a valuable addition to the existing emotional databases. In total, 24 non-prototypical emotional states are labeled by three first Chinese native speakers. In contrast to other available emotional databases, we provided multi-emotion labels and fake/suppressed emotion labels. To our best knowledge, this database is the first large-scale Chinese natural emotion corpus dealing with multi-modal and natural emotion. However, CASIA emotional speech corpus was paid and it was for companies, so we have talked with prof. Danilo Comminiello and he had mentioned using one similar database which is called Surrey Audio-Visual Expressed Emotion(SAVEE) Database. The SAVEE database was recorded from four native English male speakers (identified as DC, JE, JK, KL), postgraduate students and researchers at the University of Surrey aged from 27 to 31 years. Emotion has been described psychologically in discrete categories: anger, disgust, fear, happiness, sadness and surprise. This is supported by the cross-cultural studies of Ekman and studies of automatic emotion recognition tended to focus on recognizing these. We added neutral to provide recordings of 7 emotion categories. The text material consisted of 15 TIMIT sentences per emotion: 3 common, 2 emotion-specific and 10 generic sentences that were different for each emotion and phonetically-balanced. The 3 common and 2 × 6 = 12 emotion-specific sentences were recorded as neutral to give 30 neutral sentences. There are seven classes in this data such as anger, disgust, fear, happiness, sadness, surprise and neutral.

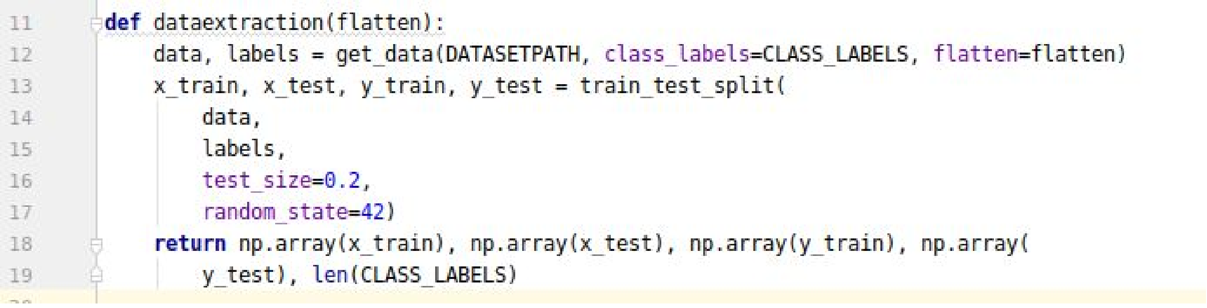
As we have mentioned earlier, LSTM network models are a type of recurrent neural network that are able to learn and remember over long sequences of input data. The model can support multiple parallel sequences of input data. The benefit of using LSTM for sequence classification is that they can learn from the raw time series data directly, and in turn do not require domain expertise to manually engineer input features. The model can learn an internal representation of the time series data and ideally achieve comparable performance to models fit on a version of the dataset with engineered features. Thus, we needed to make several steps. Firstly, we needed to load the voice dataset into our memory. Then within the dataset, we loaded each of the train and test datasets. Secondly, after loading the data into the memory we were ready for modeling which we define as fitting and evaluating the LSTM model. In our project, we have called it *train()* which takes train, values and epochs for the model. In our project, we wanted to use 100 for the epoch number.



```
def train(self, x_train, y_train, x_val=None, y_val=None, n_epochs=100):
    best_acc = 0
    if x_val is None or y_val is None:
        x_val, y_val = x_train, y_train
    for i in range(n_epochs):
        p = np.random.permutation(len(x_train))
        x_train = x_train[p]
        y_train = y_train[p]
        self.model.fit(x_train, y_train, batch_size=32, epochs=1)
        loss, acc = self.model.evaluate(x_val, y_val)
        if acc > best_acc:
            best_acc = acc
    self.trained = True
```

Fig n. Fitting the Data

It is also crucial to mention some things like how we extracted the data. For this case, we have initialized the data and labels within our project and added the path of dataset, class labels we mentioned above. From this one, we have used the train_test_split function from the sklearn *library* with the test size of 0.2 and train size of 0.8 which makes 20% for the test and 80% for the train data. At the end, we have returned the array with train and test sets for the future use to extract the model.

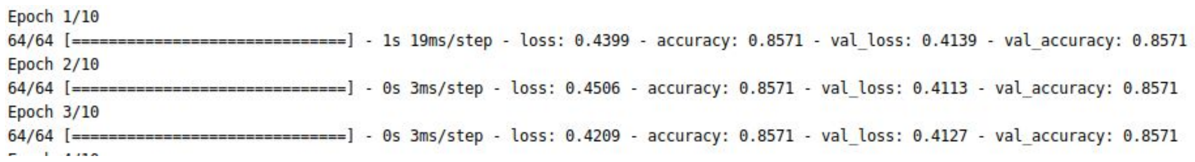


```
def dataextraction(flatten):
    data, labels = get_data(DATASETPATH, class_labels=CLASS_LABELS, flatten=flatten)
    x_train, x_test, y_train, y_test = train_test_split(
        data,
        labels,
        test_size=0.2,
        random_state=42)
    return np.array(x_train), np.array(x_test), np.array(y_train), np.array(
        y_test), len(CLASS_LABELS)
```

Fig n. Data Extraction

As mentioned in the project, we needed to make attention-gate bidirectional LSTM. Firstly, we need to mention that bidirectional LSTMs are an extension of traditional LSTMs that can improve model performance on sequence classification problems. In problems where all timesteps of the input sequence are available, Bidirectional LSTMs train two instead of one LSTMs on the input sequence. The first on the input sequence as-is and the second on a reversed copy of the input sequence. This can provide additional context to the network and result in faster and even fuller learning on the problem.

To make things more clear, made a parent class and made the LSTM the child class of that class. In our parent class, we have initialized several methods such as *save_model, load_model, train* and etc. We have used *save_model* method to save the weights in our path for later use. The load_model is taking the model we have saved and loads the weights from that. One of the important methods in our project is *train* method because it takes several arguments such as *x_train and y_train* and within the range of epochs (we have made the epochs as small as we could because our dataset is not that big so small epochs would be enough) and then we are getting the model and doing compiling and fitting the data and if the accuracy is bigger than previous accuracy it gives us the current accuracy as the best one. In our *train* method we are doing main plottings that we will be showing in the results section. As shown in *Fig.* we have tried to train the data and get better results after every iteration.



```
Epoch 1/10
64/64 [==============================] - 1s 19ms/step - loss: 0.4399 - accuracy: 0.8571 - val_loss: 0.4139 - val_accuracy: 0.8571
Epoch 2/10
64/64 [==============================] - 0s 3ms/step - loss: 0.4506 - accuracy: 0.8571 - val_loss: 0.4113 - val_accuracy: 0.8571
Epoch 3/10
64/64 [==============================] - 0s 3ms/step - loss: 0.4209 - accuracy: 0.8571 - val_loss: 0.4127 - val_accuracy: 0.8571
```

Fig n. Values for Training

In addition to this, we have also implemented speech recognition in order to take the voice input from the user and predict the class for our dataset. For instance, when a user says something like happy, the algorithm tries to predict the class for that. However, sometimes we get less accuracy for this implementation, but sometimes it detects the class correctly.

Results

As we have implemented our algorithm, the most important part is to check the results and see how accurate our algorithm is. As described enough above, we have implemented bidirectional LSTM (Long Short-Term Memory) to check the accuracy of our algorithm within the dataset we have downloaded. With the help of the *sklearn* library we have used *confusion_matrix and accuracy* to check how well our algorithm performs. As a result of training and testing our dataset we have got the accuracy of more than 90% for SAVEE dataset in our project. As you might see from the figure, we have got the accuracy of 99% for the training dataset and 0.95 for the test dataset. We have also checked the loss of the dataset with the help of fitting the data history and as you might see in the figures we have got the 0.12 for the test dataset and 0.07 for the training dataset which is good enough for our project.

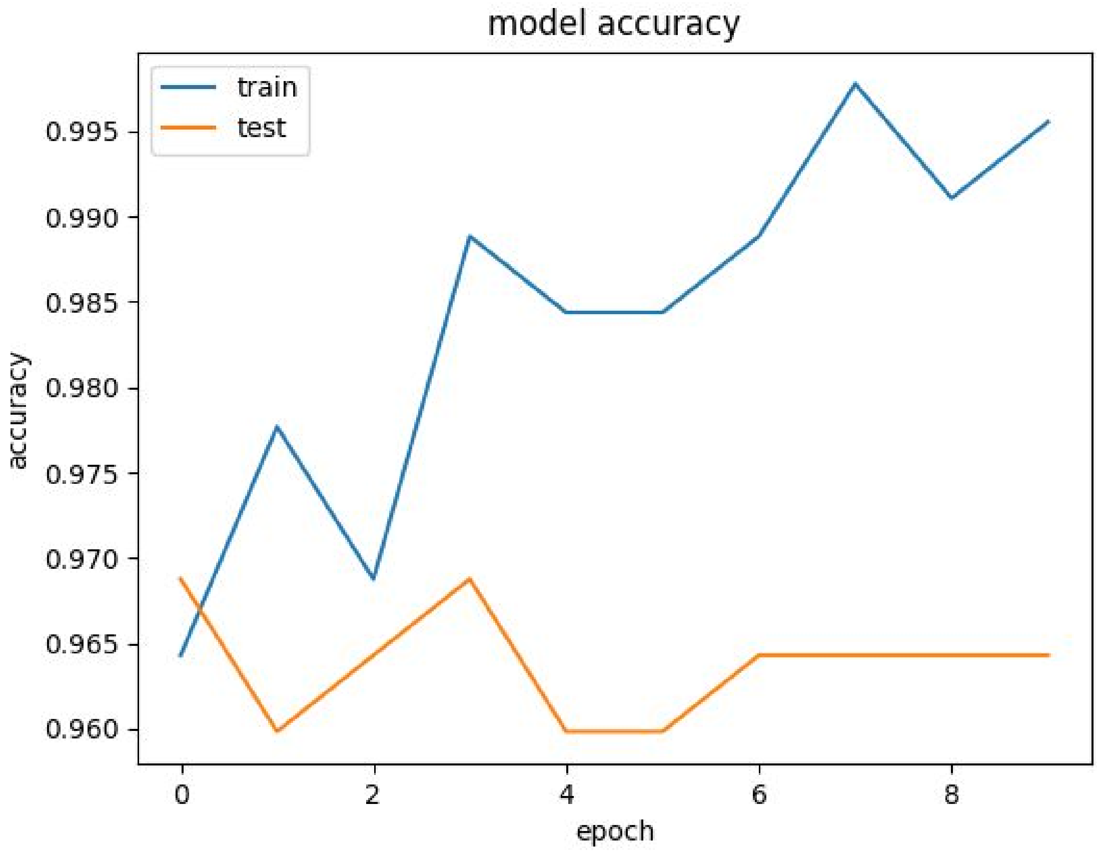
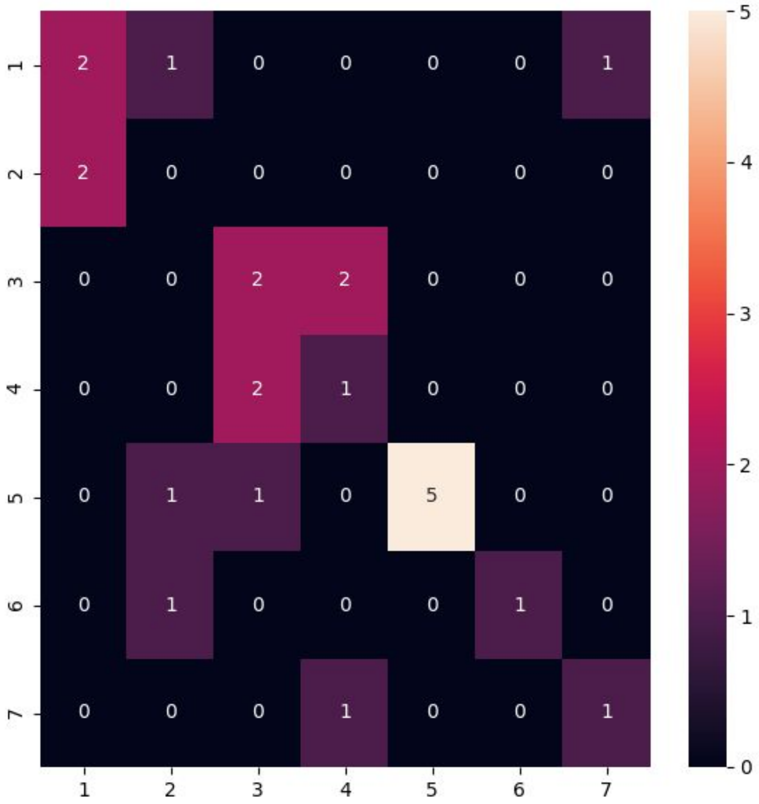


Fig. Plotting Model Accuracy

It is also crucial to mention the confusion matrix we have calculated in our project. As shown in *Fig.* our confusion matrix is like that:



	1	2	3	4	5	6	7
1	2	1	0	0	0	0	1
2	2	0	0	0	0	0	0
3	0	0	2	2	0	0	0
4	0	0	2	1	0	0	0
5	0	1	1	0	5	0	0
6	0	1	0	0	0	1	0
7	0	0	0	1	0	0	1

Fig. Confusion Matrix

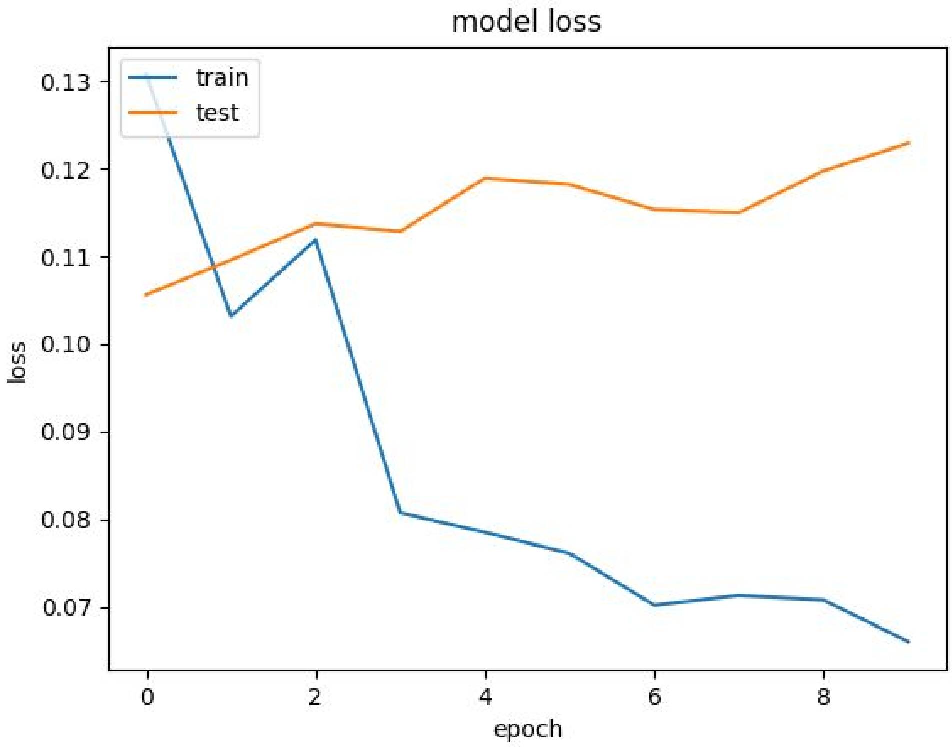


Fig. Plotting Model Loss

Conclusion

This paper has been created and made exhaustive research on implementing attention-based LSTM for speech emotion recognition and calculating the accuracy of the model after doing feature extraction. Till the end of this project, we have successfully made a lot of research on the topics of bidirectional and attention-based LSTM and tried to explore and understand how all of them work. In this report, we have presented how the algorithm exactly calculates the accuracy and confusion matrix using LSTM. The accuracy of models we have extracted are more than 90% which makes the algorithm work good. However, by the future work we will be trying to find ways to increase the accuracy of our algorithm.

References

1. Feature Extraction by DeepAI
    a. Feature Extraction Definition
2. SAVEE Database
    a. http://kahlan.eps.surrey.ac.uk/savee/
3. A Gentle Introduction to Long Short-Term Memory Networks by the Experts
    a. https://machinelearningmastery.com/gentle-introduction-long-short-term-memory-networks-experts/
4. How to Develop a Bidirectional LSTM For Sequence Classification in Python with Keras
    a. https://machinelearningmastery.com/develop-bidirectional-lstm-sequence-classification-python-keras/
5. Base of NN
    a. https://www.academia.edu/4305355/Stuart_Russell_and_Peter_Norvig_-_Artificial_Intelligence_-_A_Modern_Approach_3rd_ed._