PROJECT REPORT

on

Analyzing Speech to Detect Emotions using Deep Neural Networks

by

Deepak Hirawat	**BE/10485/2016**
Momojit Ghosh	**BE/10401/2016**
Satyabrat Bhol	**BE/10391/2016**

Submitted in partial fulfilment of requirement for the award in the degree of

Bachelor of Engineering

In

Electronics and Communication Engineering

Under the Guidance of

Dr. Rupesh Kumar Sinha

Department of Electronics and Communication Engineering

Birla Institute of Technology, Mesra, Ranchi – 835215

June 2020

DECLARATION

This is to certify that the work presented in this project entitled "**Analysing Speech to Detect Emotions**", in partial fulfilment of the requirement for the award of Degree of **Bachelor of Engineering in Electronics & Communication Engineering**, submitted to the Department of Electronics & Communication Engineering of Birla Institute of Technology, Mesra, Ranchi, Jharkhand is a bona fide work carried out by **Satyabrat Bhol, Deepak Hirawat** and **Momojit Ghosh** under my supervision and guidance.

To the best of my knowledge, the content of this project, either partially or fully, has not been submitted to any other institution for the award of any other degree.

Date:

Dr. Rupesh Kumar Sinha
Department of ECE
Birla Institute of Technology
Mesra, Ranchi.

CERTIFICATE OF APPROVAL

This is to certify that the project entitled "**Analysing Speech to Detect Emotions**" is hereby approved as a suitable design of an Engineering subject, carried out and presented in satisfactory manner to warrant its acceptance as prerequisite to the degree for which it has been submitted.

It is understood that by this approval, the undersigned do not necessarily endorse any conclusion drawn or opinion expressed therein but approve the project for which it is submitted.

Internal Examiner	External Examiner
Date:	Date:

Dr. Srikanta Pal
Head of Department
Department of Electronics and Communication Engineering
Birla Institute of Technology
Mesra, Ranchi

ACKNOWLEDGEMENT

We owe our deepest gratitude to our guide **Dr. Rupesh Kumar Sinha**, who helped us throughout the project, despite his extremely busy schedule. He made sure that we learnt by practice, always motivating us to think a step further, work a little harder. Our interactions with him always resulted in newer ideas and proved beneficial towards our work. Without his constant presence and supervision our work would not have been successful.

We are very grateful to **Dr. Srikanta Pal, Professor & Head, Dept. of Electronics and Communication Engineering, Birla Institute of Technology, Mesra** for extending all the facilities and always giving valuable suggestions for pursing this course.

And lastly our batch mates who have always been there with valuable suggestions and support in our endeavours.

Satyabrat Bhol	**Deepak Hirawat**	**Momojit Ghosh**
BE/10391/2016	**BE/10485/2015**	**BE/10401/2015**

1. ABSTRACT

The ability to understand people through spoken language is a skill that many human beings take for granted. On the contrary, the same task is not as easy for machines, as it is a consequence of many variables which vary the speaking sound wave while people are talking to each other. A sub-task of speech understanding is about the detection of the emotions elicited by the speaker while talking, and this is the focus of our contribution via this project. We are presenting a classification model of emotions elicited by speeches based on deep neural networks (CNNs). For the purpose, we focused on the audio recordings available in the Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS) dataset.

The model has been trained to classify eight different emotions (neutral, calm, happy, sad, angry, fearful, disgust, surprise) which correspond to the ones proposed by Paul Ekman (American psychologist and professor emeritus at the University of California, San Francisco who is a pioneer in the study of emotions and their relation to facial expressions) plus the neutral and calm ones. If we consider the evaluation metric as the F1 score, we obtain a weighted average of 0.91 on the test set and the best performances on the "Calm" class with a score of 0.95. Our worst results have been observed for the "Sad" class with a score of 0.87 but nevertheless better than many models.

It is known how each of us expresses more than one emotion at a time, but our opinion is that it is extremely difficult, both for the speaker and for the listener, to be able to recognize which and in what percentage of the emotions are mixed. In this regard we decided to create a model that aims to identify only the emotion that has a greater value in the audio track. In this work, our goal is to use pure audio data considering Mel-frequency cepstral coefficients (MFCC's).

TABLE OF CONTENTS

LIST OF FIGURES

2. INTRODUCTION

The human communication through the spoken language is the base for information exchange and it is the main aspect of the society since the first human settlements. In the same way, emotions go back to a primordial instinct prior to the spoken language that we know today and that can be considered as the first natural communication strategies. In today's common languages, the word is followed by emotions in order to make communication more direct, clear and understandable to all.

Speech and song are often considered overlapping forms of vocal expression. For example, in ancient Greek, the words for singing and speaking do not have the distinct meanings they do today, and emotion is expressed over both song and speech.

As humans, we consider facial expressions, volume and tone of the speaker' voice and many other factors, while our brain works on connecting all the different information to interact with other people and judge their emotions. In the area of human-machine interaction, we have tried for a long time to provide the ability to understand the language spoken on a computer. Today, many tools come close to this ability, even if the interpretation of emotions during the dialogue is a characteristic often overlooked. Due to the importance of including this aspect in a more complex and complete model of human-machine interaction, in this work we focused on an efficient strategy to be able to identify the main emotion expressed by a subject during the dialogue.

In the literature it is shown how each of us expresses more than one emotion at a time but our opinion is that it is extremely difficult, both for the speaker and for the listener, to be able to recognize which and in what percentage of the emotions are mixed. In this regard it was decided to create a model that aims to identify only the emotion that has a greater value in the audio track and is achieved considering Mel-frequency cepstral coefficients (MFCC's).

3. Previous Works

In this field of research, many classification strategies have been presented in the last years. One of the systems, proposed by Iqbal et al. used Gradient Boosting, KNN and SVM to work on a granular classification on the RAVDESS dataset used in this work to identify differences based on gender with approximately between 40% and 80% overall accuracy, depending on the specific task. In particular, the proposed classifiers performed differently in different datasets, but we considered only the work done on RAVDESS for the scope of this paper. In Iqbal et al. work, three types of datasets have been created including only male recordings, only female recordings and a combined one. In RAVDESS (male) SVM and KNN have 100% accuracy in both anger and neutral, but in happiness and sadness Gradient Boosting performs better than SVM and KNN. In RAVDESS (female) SVM achieves 100% accuracy in anger as same as male part. SVM has overall good performance except in sadness. Performance of KNN is also good in anger and neutral like 87% and 100% respectively. In anger and neutral, Gradient Boosting performs poorly. KNN performance is very poor in happiness and sadness comparing with other classifiers. In male and female combined dataset, performances of SVM and KNN are good in anger and neutral rather than Gradient Boosting. KNN’s performance is poor in happiness and sadness. Average performances of classifiers in male dataset are better than female dataset except SVM. In combined database, SVM get high accuracy than gender-based datasets.

Another approach presented by Jannat et al. achieved 66.41% accuracy on audio data and more than 90% accuracy mixing audio and video data. Given pre-processed image data that includes faces and audio waveforms, Jannat et al. trained 3 separately deep networks: one network only on image data, another only on the plotted audio waveforms, and the third on both image and waveform data. One of the first approaches that used the RAVDESS dataset, but classifying only some of the emotions available, was published by Zhang et al., reaching an overall accuracy score higher than the model proposed in this work but using less classes. Giving more details, in Zhang et al. three shared emotion recognition models for speech and song have been proposed: a simple model, a single-task hierarchical model and a multi-task hierarchical model. The simple model creates a single classifier, independent of domain. The two hierarchical models use domain during training. The single task model trains a separate emotion classifier for each domain. The multi-task model trains a multi-task classifier to jointly predict emotion across both domains. In the testing phase, the testing data are separated based on the predicted domain. The data are analysed using the classifier corresponding to the estimated domain. The work has been conducted adopting the directed acyclic graph SVM (DAGSVM).

Certain researchers have looked at the similarity between music and speech emotion communication. Juslin and Laukka conducted a meta-analysis of studies analysing speech and music performance. They found that some acoustic features play similar roles in both music and speech emotion communication. For example, anger is associated with fast tempo, high sound level and elevated high-frequency energy in both music and speech. Ilie and Thompson found that the manipulation of certain acoustic features of music and speech, such as loudness and rate, resulted in similar emotion perception. For example, loud excerpts were judged as more pleasant, energetic, and tense. The work of Weninger et al. investigated the shared acoustic features in speech, music and sound and introduced the cross-domain correlation coefficient as a measure of relevance. Their results indicate that there may be shared and divergent properties in emotion communication across domains of expression.

Support Vector Machine classifiers (SVM) were originally designed for binary classification. There are currently two types of approaches for multi-class SVM: one builds and combines multiple binary classifiers, while the other considers all data in one optimization problem. Typical methods of the former approach include one-against-all, one-against-one and directed acyclic graph SVM (DAGSVM). Experiments comparing these two approaches showed that the combinations of binary classifiers outperformed the all-in-one multi-class optimization in both classification accuracy and training/testing time and demonstrated that one-against-one method and DAG are more suitable for practical use. Evgeniou et al. extended single-task SVMs to multitask scenario. This extension assumes that there are T related tasks that share the same space. It solves multiple tasks together by imposing a regularization constraint on the average model and controlling the differences between the tasks using additional model parameters. Empirical results show that this multi-task SVM outperforms earlier multitask learning methods as well as single-task SVM. Evgeniou, Micchelli and Pontil improved the previous idea by proposing multi-task kernels. They developed multi-task kernels to extend the single-task learning methods that use kernels to multi-task learning. These methods all require that the tasks are label-compatible (tasks sharing the same set of labels).

4. About the Dataset

The dataset used in this work is the Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS). This dataset contains 7356 files made by 24 professional actors (12 females, 12 male), vocalizing two lexically matched statements in a neutral North American accent. Speech includes calm, happy, sad, angry, fearful, surprise, and disgust expressions, and song contains calm, happy, sad, angry, and fearful emotions. Each expression is produced at two levels of emotional intensity (normal, strong), with an additional neutral expression. All conditions are available in three modality formats: Audio-only (16bit, 48 kHz .wav), Audio-Video (720p H.264, AAC 48 kHz, .mp4), and Video-only (no sound). 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 interrater reliability were reported.

Each of the 7356 RAVDESS files has a unique filename. The filename consists of a 7-part numerical identifier (e.g., 02- 01-06-01-02-01-12.mp4). These identifiers define the stimulus characteristics:

- Modality (01 = full-AV, 02 = video-only, 03 = audio only)
- Vocal channel (01 = speech, 02 = song)
- Emotion (01 = neutral, 02 = calm, 03 = happy, 04 = sad, 05 = angry, 06 = fearful, 07 = disgust, 08 = surprised)
- Emotional intensity (01 = normal, 02 = strong). NOTE: There is no strong intensity for the ’neutral’ emotion
- Statement (01 = “Kids are talking by the door”, 02 = “Dogs are sitting by the door”)
- Repetition (01 = 1st repetition, 02 = 2nd repetition)
- Actor (01 to 24. Odd numbered actors are male, even numbered actors are female).

The emotion content of the dataset was evaluated by 350 human participants, each utterance receiving 10 ratings from 10 different participants. The participants were asked to identify the emotion expressed by the performer from the set of the target emotions or indicate none is correct. An agreement rate ranging from 0 to 1 was calculated for each utterance. A score of 1 means total consensus between evaluators and the target emotion, while a score of 0 means complete lack of consensus. The evaluators also rated the emotional intensity and the emotional genuineness of the performer.

4.1 Enrichment of training data: As deep learning models struggle with little amount of data and this dataset does not have a great number of features, a pipeline has been created to enrich the training and the test set. In particular, using the video available in the dataset we have extracted with the FFMPEG library a new set of features with a different frequency: the audio files of the dataset have a frequency of 48MHz, while the ones that have been extracted from the videos have a frequency of 44,1MHz, thus introducing some (necessary) noise but still being able to increment the dimension of the training and the test set.

4.2 Metrics, data splitting and experimental runs: The files have been randomly split into train and test datasets with a test set composed with the 33% of the original full dataset. Consequently, the training set is composed by a 3315 MFCC vector of 40 features obtained thanks to the LibROSA library. The test set shape is 1633 x 40. No cross-validation set has been used for this task. Both the sets are already labelled with the class value encoded as described before. We used as evaluation metric the F1 score as a compact indicator of the quality of the classifier and as standard for comparing our results with them at state of the art. The model has been trained using the sparse categorical cross entropy loss function and rmsprop optimizer for 1000 epochs and the best models have been used for the classification phase. The number of batches has been set to 16 for optimization reasons. During the training we validate the model using the accuracy score as common in deep learning architectures.

5. FEATURE EXTRACTION

5.1 Pre-Emphasis Filter

Pre-emphasis boosts the amount of energy in the high frequencies. For voiced segments like vowels, there is more energy at the lower frequencies than the higher frequencies. This is called spectral tilt which is related to the glottal source (how vocal folds produce sound). Boosting the high-frequency energy makes information in higher formants more available to the acoustic model. This improves phone detection accuracy. For humans, we start having hearing problems when we cannot hear these high-frequency sounds. Also, noise has a high frequency. In the engineering field, we use pre-emphasis to make the system less susceptible to noise introduced in the process later. For some applications, we just need to undo the boosting at the end.

A pre-emphasis filter is useful in several ways:

1. Balance the frequency spectrum since high frequencies usually have smaller magnitudes and lower frequencies,
2. Avoid numerical problems during the Fourier transform operation
3. Also improves the Signal-to-Noise Ratio (SNR).

5.2 Framing and Windowing

The short window of signal is called frame. In implementational view the windowing corresponds to what is understood in filter design as window-method: a long signal (of speech for instance or ideal impulse response) is multiplied with a window function of finite length, giving finite-length weighted (usually) version of the original signal. In speech processing the shape of the window function is not that crucial but usually some soft window like Hanning, Hamming, triangle, half parallelogram, not with right angles. After slicing the signal into frames, we apply a window function such as the Hamming window to each frame. Hamming window reduces side lobe and thus prevents signal from getting contaminated. We apply a window function to the frames to counteract the assumption made by the FFT that the data is infinite and to reduce spectral leakage.

5.3 Filter Banks

Filter banks are arrangements of low pass, bandpass, and high pass filters used for the spectral decomposition and composition of signals. The task performed by a filter bank are combinations of the common operations of spectral translation, bandwidth reduction, and sample rate changes. What we do in filter bank designs is coupling the processes, embedding one within another, and rearranging the order in which these operations are performed. We compute filter banks by applying triangular filters on a Mel-scale to the power spectrum to extract frequency bands.

The Mel-scale mimics the non-linear human ear perception of sound, by being more discriminative at lower frequencies and less discriminative at higher frequencies. Each filter in the filter bank is triangular having a response of 1 at the centre frequency and decrease linearly towards 0.

5.4 Mel Frequency Cepstral Coefficients (MFCC)

Mel-frequency cepstral coefficients (MFCCs) are coefficients that collectively make up an MFC. They are derived from a type of cepstral representation of the audio clip (a nonlinear "spectrum-of-a-spectrum"). The difference between the cepstrum and the mel-frequency cepstrum is that in the MFC, the frequency bands are equally spaced on the mel scale, which approximates the human auditory system's response more closely than the linearly spaced frequency bands used in the normal cepstrum. This frequency warping can allow for better representation of sound, for example, in audio compression. The reasons for choosing MFCC is that the frequency bands are equally spaced on the Mel scale. MFCCs approximates the human auditory response system and so it is a more logical choice.

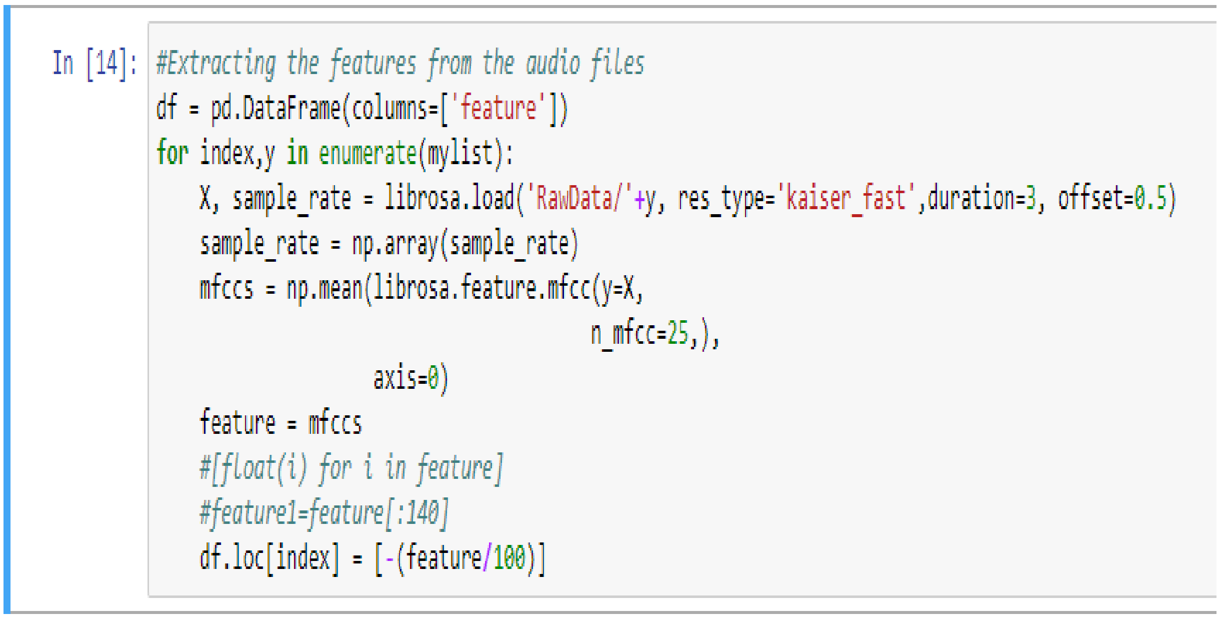

```
In [14]: #Extracting the features from the audio files
         df = pd.DataFrame(columns=['feature'])
         for index,y in enumerate(mylist):
             X, sample_rate = librosa.load('RawData/'+y, res_type='kaiser_fast',duration=3, offset=0.5)
             sample_rate = np.array(sample_rate)
             mfccs = np.mean(librosa.feature.mfcc(y=X,
                                                 n_mfcc=25,),
                             axis=0)
             feature = mfccs
             #[float(i) for i in feature]
             #feature1=feature[:140]
             df.loc[index] = [-(feature/100)]
```

Fig 1. Generating MFCC features using Python

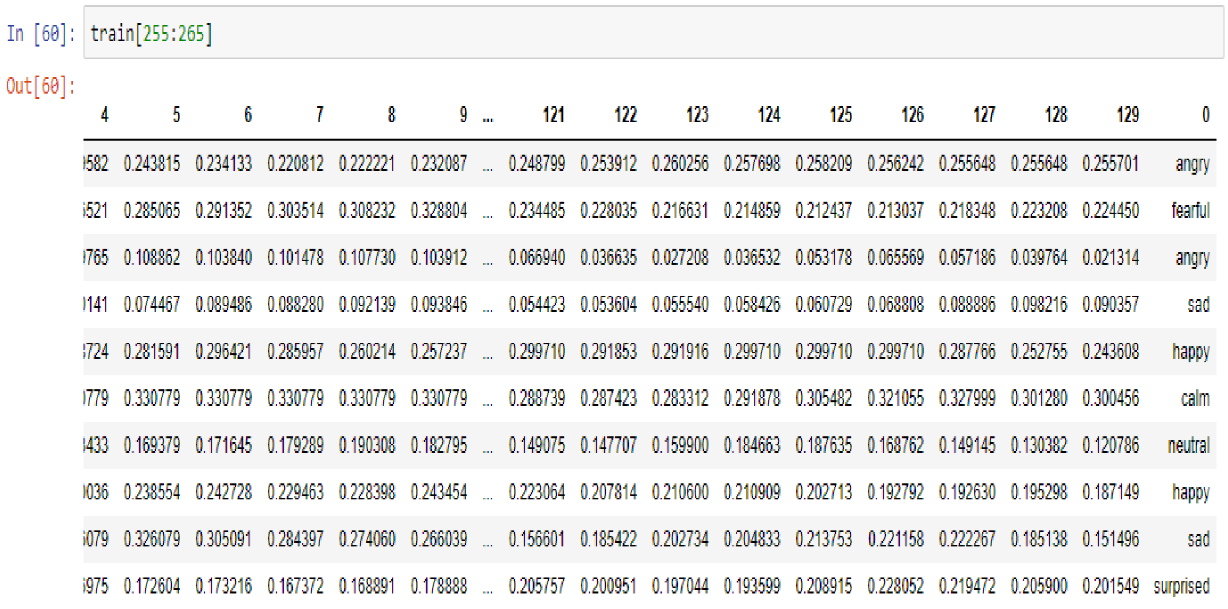

```
In [60]: train[255:265]

Out[60]:
    4        5        6        7        8        9 ...      121      122      123      124      125      126      127      128      129         0
  582 0.243815 0.234133 0.220812 0.222221 0.232087 ... 0.248799 0.253912 0.260256 0.257698 0.258209 0.256242 0.255648 0.255648 0.255701     angry
  521 0.285065 0.291352 0.303514 0.308232 0.328804 ... 0.234485 0.228035 0.216631 0.214859 0.212437 0.213037 0.218348 0.223208 0.224450   fearful
  765 0.108862 0.103840 0.101478 0.107730 0.103912 ... 0.066940 0.036635 0.027208 0.036532 0.053178 0.065569 0.057186 0.039764 0.021314     angry
  141 0.074467 0.089486 0.088280 0.092139 0.093846 ... 0.054423 0.053604 0.055540 0.058426 0.060729 0.068808 0.088886 0.098216 0.090357       sad
  724 0.281591 0.296421 0.285957 0.260214 0.257237 ... 0.299710 0.291853 0.291916 0.299710 0.299710 0.299710 0.287766 0.252755 0.243608     happy
  779 0.330779 0.330779 0.330779 0.330779 0.330779 ... 0.288739 0.287423 0.283312 0.291878 0.305482 0.321055 0.327999 0.301280 0.300456      calm
  433 0.169379 0.171645 0.179289 0.190308 0.182795 ... 0.149075 0.147707 0.159900 0.184663 0.187635 0.168762 0.149145 0.130382 0.120786   neutral
  036 0.238554 0.242728 0.229463 0.228398 0.243454 ... 0.223064 0.207814 0.210600 0.210909 0.202713 0.192792 0.192630 0.195298 0.187149     happy
  079 0.326079 0.305091 0.284397 0.274060 0.266039 ... 0.156601 0.185422 0.202734 0.204833 0.213753 0.221158 0.222267 0.185138 0.151496       sad
  975 0.172604 0.173216 0.167372 0.168891 0.178888 ... 0.205757 0.200951 0.197044 0.193599 0.208915 0.228052 0.219472 0.205900 0.201549 surprised
```

Fig. 2 MFCC features

6. MODELS TESTED

6.1 MLP (Multi-Layer Perceptron) Model

A **Multi-Layer Perceptron** (MLP) is a class of feedforward artificial neural network (ANN). Multilayer perceptrons are sometimes colloquially referred to as "vanilla" neural networks, especially when they have a single hidden layer. An MLP consists of at least three layers of nodes: an input layer, a hidden layer and an output layer. Except for the input nodes, each node is a neuron that uses a nonlinear activation function.

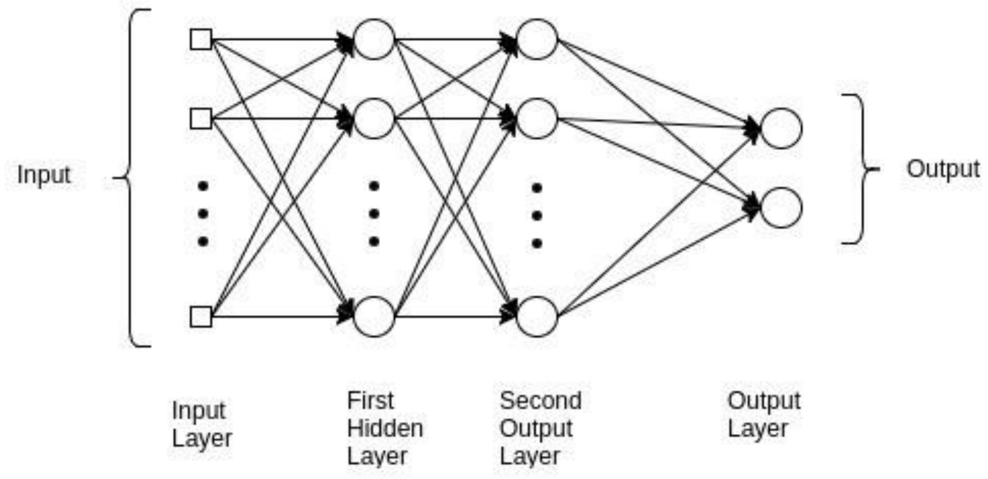


Fig. 3 Layers of perceptrons

Formally, a one-hidden-layer MLP is a function $f : R^D \rightarrow R^L$, where $D$is the size of input vector $x$and $L$is the size of the output vector $f(x)$, such that, in matrix notation:

$$f(x) = G(b^{(2)} + W^{(2)}(s(b^{(1)} + W^{(1)}x))),$$

with bias vectors $b^{(1)}$, $b^{(2)}$; weight matrices $W^{(1)}$, $W^{(2)}$and activation functions $G$and s.

The vector $h(x) = \Phi(x) = s(b^{(1)} + W^{(1)}x)$ constitutes the hidden layer. $W^{(1)} \in R^{D \times D_h}$ is the weight matrix connecting the input vector to the hidden layer. Each column $W^{(1)}_{\cdot i}$ represents the weights from the input units to the i-th hidden unit. Typical choices for s include $tanh$, with $tanh(a) = (e^a - e^{-a})/(e^a + e^{-a})$, or the logistic $sigmoid$function, with $sigmoid(a) = 1/(1 + e^{-a})$.

The initial values for the weights of a hidden layer $i$should be uniformly sampled from a symmetric interval that depends on the activation function. For $tanh$activation function results show that the interval should be $[-\sqrt{\frac{6}{fan_{in}+fan_{out}}}, \sqrt{\frac{6}{fan_{in}+fan_{out}}}]$, where fan_{in} is the number of units in the $(i-1)$-th layer, and fan_{out} is the number of units in the i-th layer. For the sigmoid function the interval is $[-4\sqrt{\frac{6}{fan_{in}+fan_{out}}}, 4\sqrt{\frac{6}{fan_{in}+fan_{out}}}]$. This initialization ensures that, early in training, each neuron operates in a regime of its activation function where information can easily be propagated both upward (activations flowing from inputs to outputs) and backward (gradients flowing from outputs to inputs).

Learning Rate

There is a great deal of literature on choosing a good learning rate. The simplest solution is to simply have a constant rate. Rule of thumb: try several log-spaced values ($10^{-1}, 10^{-2}, \ldots$) and narrow the (logarithmic) grid search to the region where you obtain the lowest validation error.

Decreasing the learning rate over time is sometimes a good idea. One simple rule for doing that is $\frac{\mu_0}{1+d\times t}$ where μ_0 is the initial rate (chosen, perhaps, using the grid search technique explained above), d is a so-called "decrease constant" which controls the rate at which the learning rate decreases (typically, a smaller positive number, $10^{-3}$and smaller) and $t$is the epoch/stage.

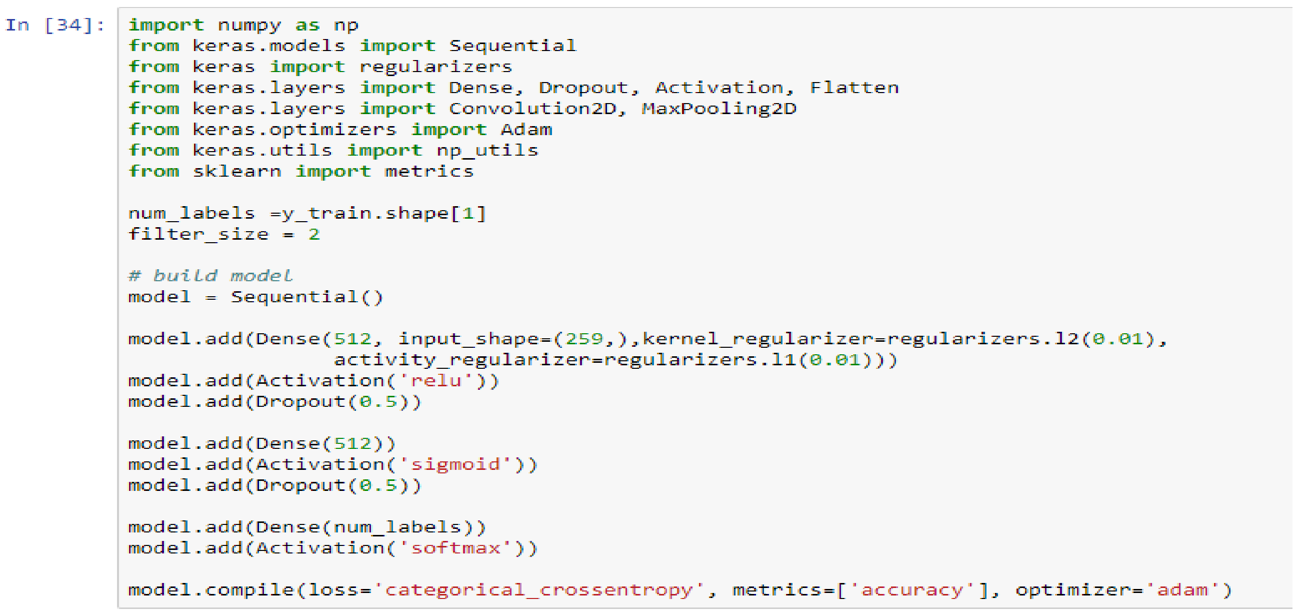
Hyper-parameter

Hyper-parameter is very much dataset dependent. Vaguely speaking, the more complicated the input distribution is, the more capacity the network will require to model it, and so the larger the number of hidden units that will be needed (note that the number of weights in a layer, perhaps a more direct measure of capacity, is $D \times D_h$ (recall D is the number of inputs and D_h is the number of hidden units).

Unless we employ some regularization scheme (early stopping or L1/L2 penalties), a typical number of hidden units vs. generalization performance graph will be U-shaped.

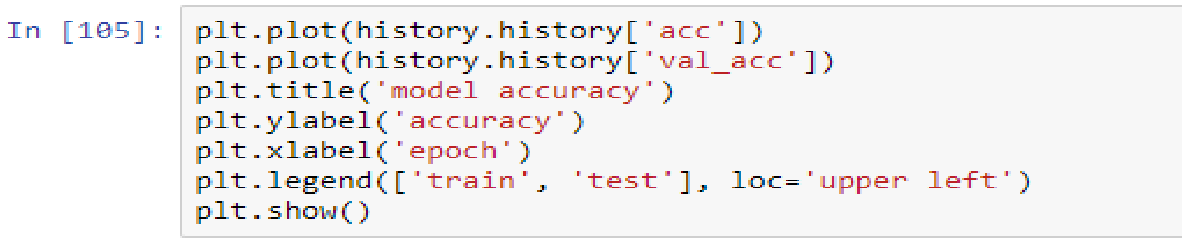
Typical values to try for the L1/L2 regularization parameter λ are $10^{-2}, 10^{-3}, \ldots$ etc.

MLP



```
In [34]: import numpy as np
         from keras.models import Sequential
         from keras import regularizers
         from keras.layers import Dense, Dropout, Activation, Flatten
         from keras.layers import Convolution2D, MaxPooling2D
         from keras.optimizers import Adam
         from keras.utils import np_utils
         from sklearn import metrics

         num_labels =y_train.shape[1]
         filter_size = 2

         # build model
         model = Sequential()

         model.add(Dense(512, input_shape=(259,),kernel_regularizer=regularizers.l2(0.01),
                         activity_regularizer=regularizers.l1(0.01)))
         model.add(Activation('relu'))
         model.add(Dropout(0.5))

         model.add(Dense(512))
         model.add(Activation('sigmoid'))
         model.add(Dropout(0.5))

         model.add(Dense(num_labels))
         model.add(Activation('softmax'))

         model.compile(loss='categorical_crossentropy', metrics=['accuracy'], optimizer='adam')
```

Fig. 4 Python code for MLP



```
In [105]: plt.plot(history.history['acc'])
          plt.plot(history.history['val_acc'])
          plt.title('model accuracy')
          plt.ylabel('accuracy')
          plt.xlabel('epoch')
          plt.legend(['train', 'test'], loc='upper left')
          plt.show()
```

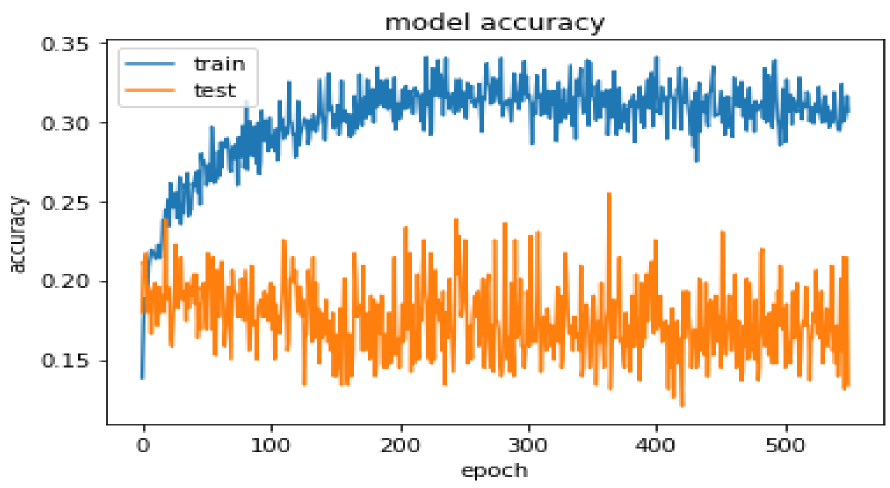


Fig. 5 MLP model results on test and train data

6.2 Long Short-Term Memory (LSTM) Model

Long Short-Term Memory (LSTM) is an artificial recurrent neural network (RNN) architecture used in the field of deep learning. Unlike standard feedforward neural networks, LSTM has feedback connections. It can not only process single data points (such as images), but also entire sequences of data (such as speech or video). For example, LSTM is applicable to tasks such as unsegmented, connected handwriting recognition, speech recognition and anomaly detection in network traffic or IDS's (intrusion detection systems).

A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three *gates* regulate the flow of information into and out of the cell. LSTMs were developed to deal with the vanishing gradient problem that can be encountered when training traditional RNNs.

Intuitively, the *cell* is responsible for keeping track of the dependencies between the elements in the input sequence.

Input gate — It discovers 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.**

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

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.**

There are connections into and out of the LSTM *gates*, a few of which are recurrent. The weights of these connections, which need to be learned during training, determine how the gates operate. An RNN using LSTM units can be trained in a supervised fashion, on a set of training sequences, using an optimization algorithm, like gradient descent, combined with backpropagation through time to compute the gradients needed during the optimization process, in order to change each weight of the LSTM network in proportion to the derivative of the error (at the output layer of the LSTM network) with respect to corresponding weight.

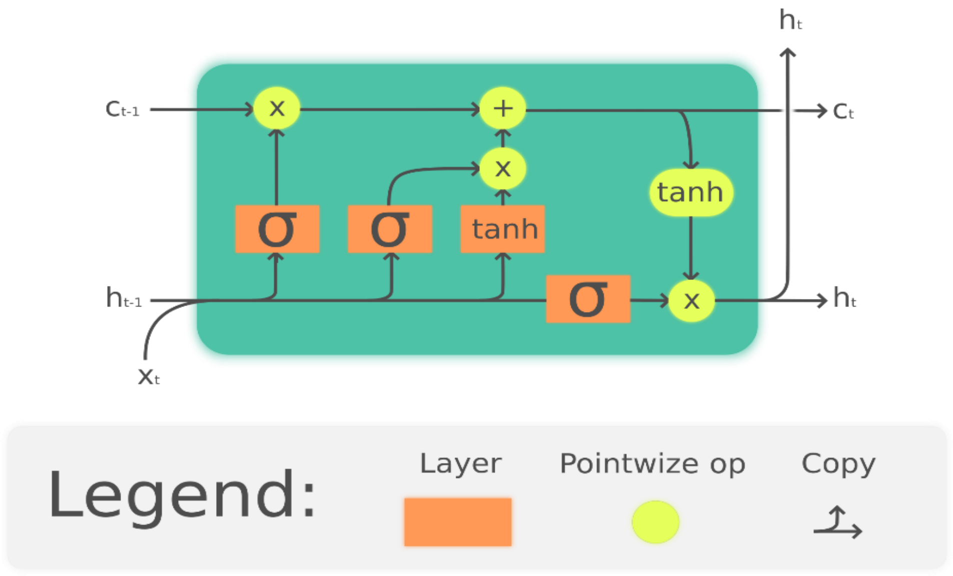
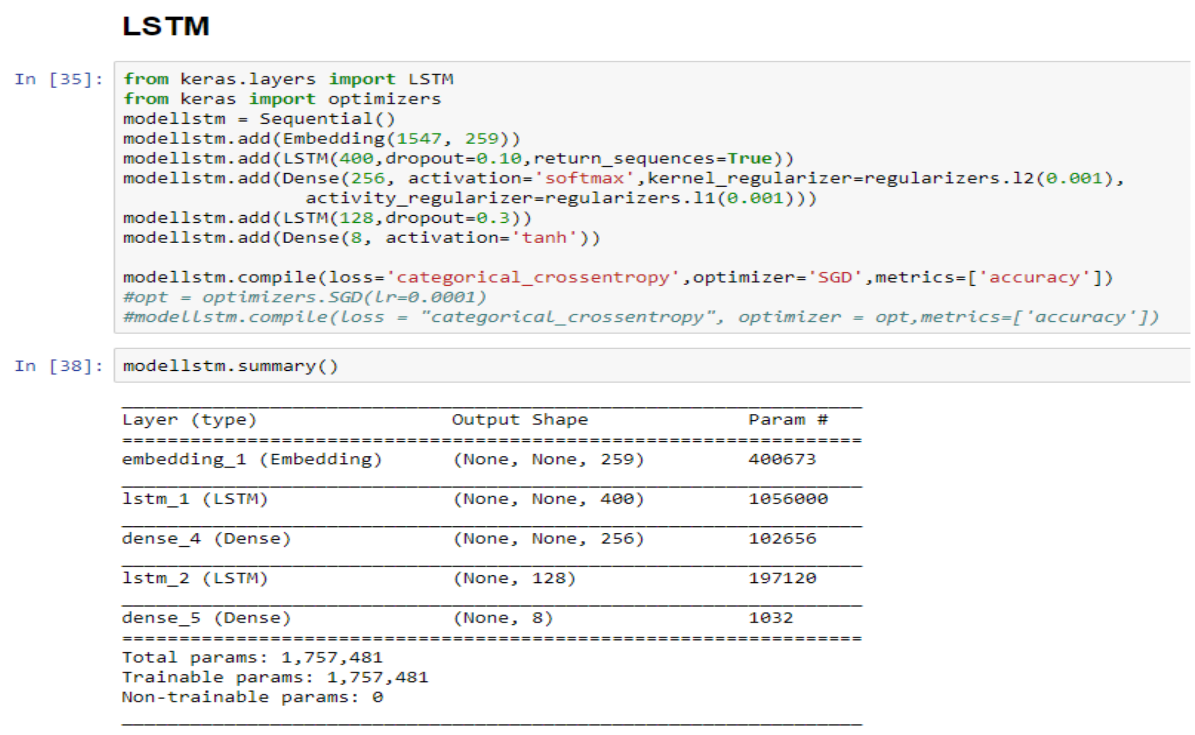


Fig. 6 A cell of LSTM

A problem with using gradient descent for standard RNNs is that error gradients vanish exponentially quickly with the size of the time lag between important events. This is due to if $\lim_{n\to\infty} W^n = 0$ the spectral radius of **W** is smaller than 1.

However, with LSTM units, when error values are back propagated from the output layer, the error remains in the LSTM unit's cell. This "error carousel" continuously feeds error back to each of the LSTM unit's gates, until they learn to cut off the value.



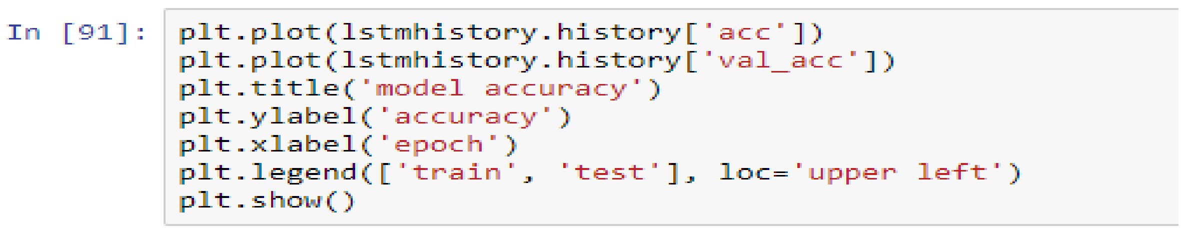
LSTM

```
In [35]: from keras.layers import LSTM
         from keras import optimizers
         modellstm = Sequential()
         modellstm.add(Embedding(1547, 259))
         modellstm.add(LSTM(400,dropout=0.10,return_sequences=True))
         modellstm.add(Dense(256, activation='softmax',kernel_regularizer=regularizers.l2(0.001),
                         activity_regularizer=regularizers.l1(0.001)))
         modellstm.add(LSTM(128,dropout=0.3))
         modellstm.add(Dense(8, activation='tanh'))

         modellstm.compile(loss='categorical_crossentropy',optimizer='SGD',metrics=['accuracy'])
         #opt = optimizers.SGD(lr=0.0001)
         #modellstm.compile(loss = "categorical_crossentropy", optimizer = opt,metrics=['accuracy'])
```

```
In [38]: modellstm.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, None, 259)         400673
_________________________________________________________________
lstm_1 (LSTM)                (None, None, 400)         1056000
_________________________________________________________________
dense_4 (Dense)              (None, None, 256)         102656
_________________________________________________________________
lstm_2 (LSTM)                (None, 128)               197120
_________________________________________________________________
dense_5 (Dense)              (None, 8)                 1032
=================================================================
Total params: 1,757,481
Trainable params: 1,757,481
Non-trainable params: 0
_________________________________________________________________
```

Fig. 7 Python code for LSTM



```
In [91]: plt.plot(lstmhistory.history['acc'])
         plt.plot(lstmhistory.history['val_acc'])
         plt.title('model accuracy')
         plt.ylabel('accuracy')
         plt.xlabel('epoch')
         plt.legend(['train', 'test'], loc='upper left')
         plt.show()
```

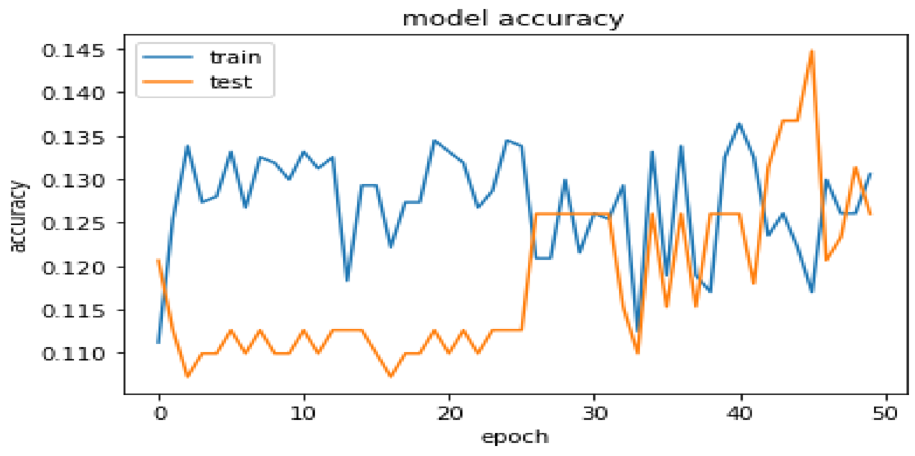


Fig. 8 LSTM model results on test and train data

6.3 Convolutional Neural Network (CNN)

In deep learning, a **Convolutional Neural Network** (**CNN**) is a class of deep neural networks, most commonly applied to analysing visual imagery. The name “convolutional neural network” indicates that the network employs a mathematical operation called convolution. Convolution is a specialized kind of linear operation. Convolutional networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers. CNNs are regularized versions of multilayer perceptrons. Multilayer perceptrons due to its virtue of "fully-connectedness" are prone to overfitting. CNNs take a different approach towards regularization: they take advantage of the hierarchical pattern and assemble complex patterns using smaller and simpler patterns.

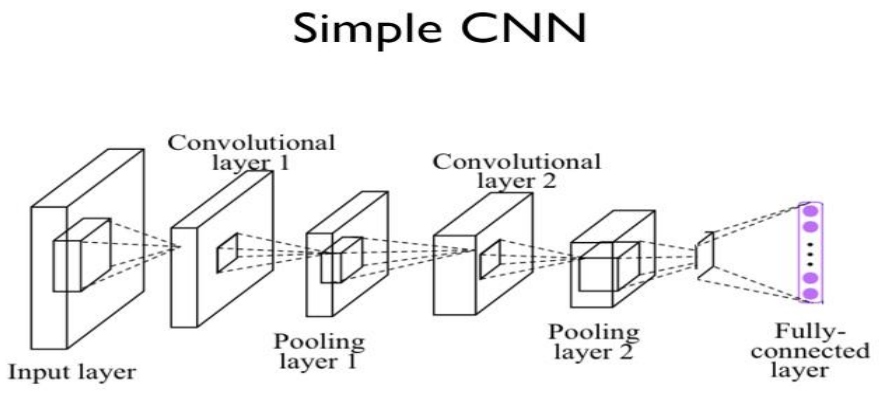


Fig. 9 A simple CNN model

Convolution

The main building block of CNN is the convolutional layer. Convolution is a mathematical operation to merge two sets of information. In our case the convolution is applied on the input data using a *convolution filter* to produce a *feature map*. We perform the convolution operation by sliding this filter over the input. At every location, we do element-wise matrix multiplication and sum the result. This sum goes into the feature map. The green area where the convolution operation takes place is called the *receptive field*. Due to the size of the filter the receptive field is also 3x3. This was an example convolution operation shown in 2D using a 3x3 filter. But these convolutions are performed in 3D. An image is represented as a 3D matrix with dimensions of height, width and depth, where depth corresponds to color channels (RGB). A convolution filter has a specific height and width, like 3x3 or 5x5, and by design it covers the entire depth of its input, so it needs to be 3D as well.

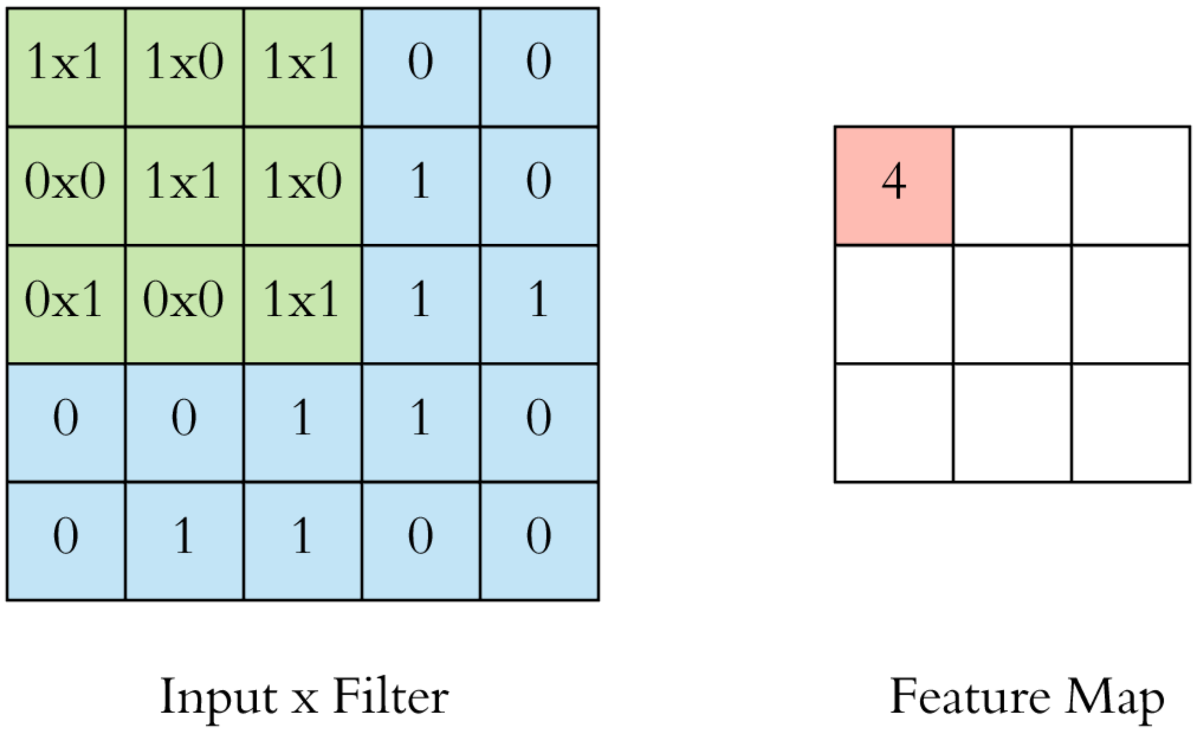


Fig. 10 Feature map generation

<u>Pooling</u>

After a convolution operation we usually perform *pooling* to reduce the dimensionality. This enables us to reduce the number of parameters, which both shortens the training time and combats overfitting. Pooling layers downsample each feature map independently, reducing the height and width, keeping the depth intact.

The most common type of pooling is *max pooling* which just takes the max value in the pooling window. Contrary to the convolution operation, pooling has no parameters. It slides a window over its input, and simply takes the max value in the window. Like a convolution, we specify the window size and stride.

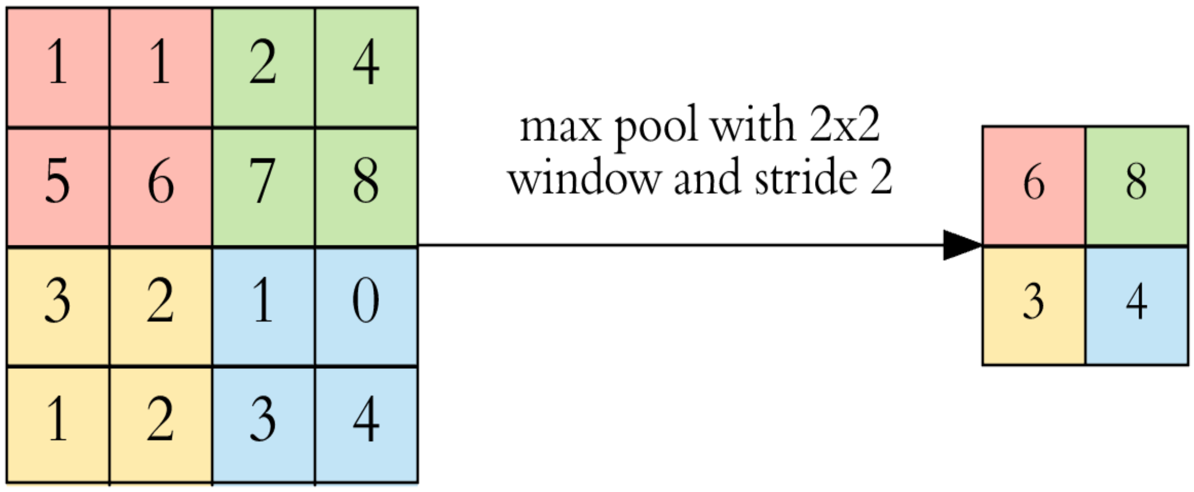


Fig. 11 Pooling Operation

After the convolution + pooling layers we add a couple of fully connected layers to wrap up the CNN architecture. The convolution layers are the main powerhouse of a CNN model. Automatically detecting meaningful features given only an image and a label is not an easy task. The convolution layers learn such complex features by building on top of each other. The first layers detect edges, the next layers combine them to detect shapes, to following layers merge this information to infer that this is an object.

Dropout

Dropout is by far the most popular regularization technique for deep neural networks. Even the state-of-the-art models which have 95% accuracy get a 2% accuracy boost just by adding dropout, which is a substantial gain at that level.

Dropout is used to prevent overfitting and the idea is very simple. During training time, at each iteration, a neuron is temporarily "dropped" or disabled with probability *p*. This means all the inputs and outputs to this neuron will be disabled at the current iteration. The dropped-out neurons are resampled with probability p at every training step, so a dropped-out neuron at one step can be active at the next one. The hyperparameter *p* is called the *dropout-rate* and it's typically a number around 0.5, corresponding to 50% of the neurons being dropped out.

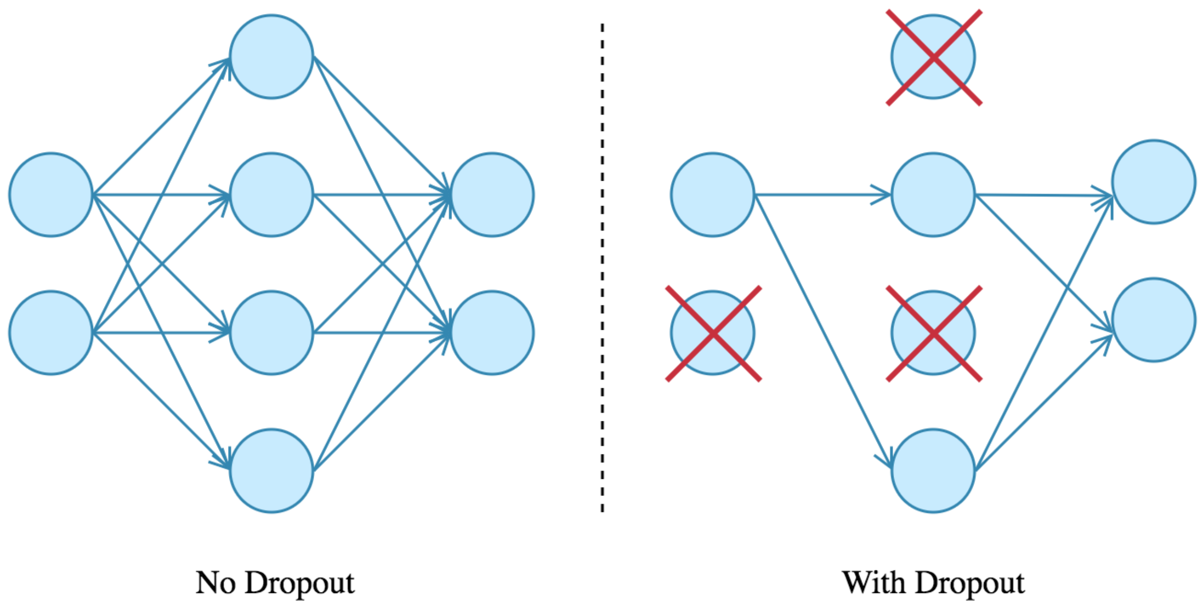


Fig. 12 Dropout comparison

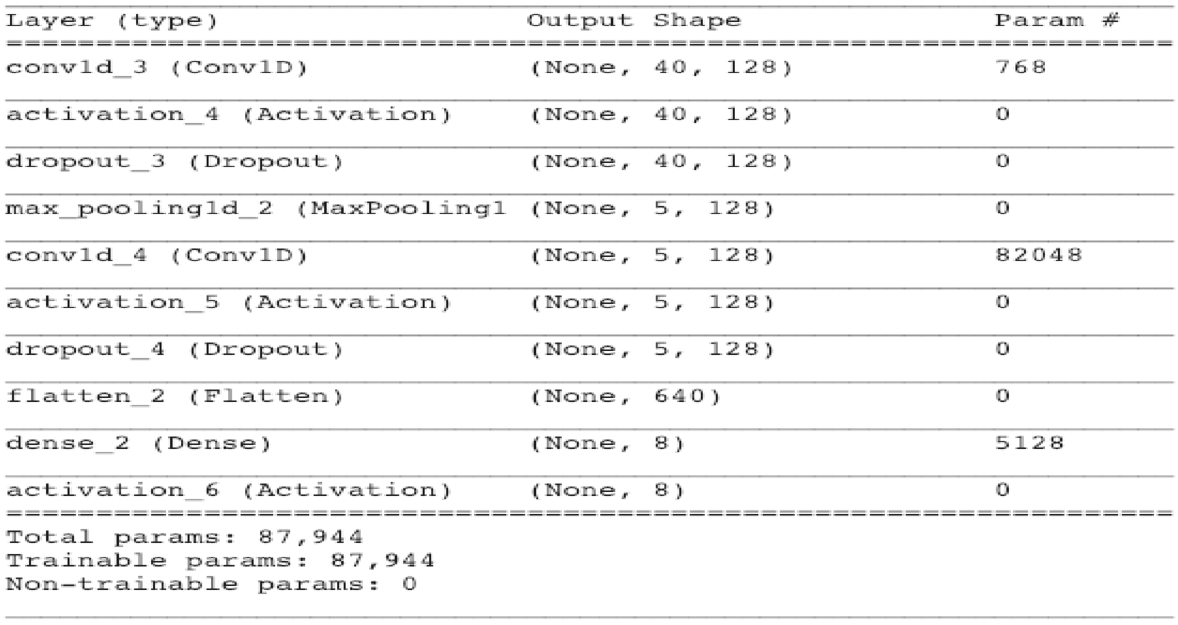
Dropout can be applied to input or hidden layer nodes but not the output nodes. The edges in and out of the dropped-out nodes are disabled. Remember that the nodes which were dropped out change at each training step. Also, we don’t apply dropout during test time after the network is trained, we do so only in training.

7. PROPOSED MODEL

The model of classification of emotions here proposed is based on convolutional neural networks (CNN) and dense layers. The key idea is considering the Mel-frequency cepstral coefficients (MFCC), commonly referred to as the "spectrum of a spectrum", as the only feature to train the model. MFCC is a different interpretation of the Mel-frequency cepstrum (MFC), and it has been demonstrated to be the state of the art of sound formalization in automatic speech recognition task. The MFC coefficients have mainly been used has the consequence of their capability to represent the amplitude spectrum of the sound wave in a compact vectoral form.

The audio file is divided into frames, usually using a fixed window size, in order to obtain statistically stationary waves. On the small frames obtained, the Discrete Fourier Transformation is applied, and only the logarithm of the amplitude spectrum is kept. The amplitude spectrum is normalized with a reduction of the "Mel" frequency scale. This operation is performed for empathizing the frequency more meaningful for a significant reconstruction of the wave as the human auditory system can perceive. For each audio file, 40 features have been extracted. The feature has been generated converting each audio file to a floating-point time series. Then, a MFCC sequence has been created from the time series. The MFCC array has been transposed and the arithmetic mean has been calculated on its horizontal axis.

The network can work on vectors of 40 features for each audio file provided as input. The 40 values represent the compact numerical form of the audio frame of 2s length. Consequently, we provide as input a of size < *number of training files* > x 40 x 1 on which we performed one round of a 1-dimensional CNN with a ReLu activation function, dropout of 20% and a max-pooling function 2 x 2. The rectified linear unit (ReLu) can be formalized as g(z) = max{0, z}, and it allows us to obtain a large value in case of activation by applying this function as a good choice to represent hidden units. Pooling can, in this case, help the model to focus only on principal characteristics of every portion of data, making them invariant by their position. We have run the process described once more by changing the kernel size.



```
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_3 (Conv1D)            (None, 40, 128)           768
_________________________________________________________________
activation_4 (Activation)    (None, 40, 128)           0
_________________________________________________________________
dropout_3 (Dropout)          (None, 40, 128)           0
_________________________________________________________________
max_pooling1d_2 (MaxPooling1 (None, 5, 128)            0
_________________________________________________________________
conv1d_4 (Conv1D)            (None, 5, 128)            82048
_________________________________________________________________
activation_5 (Activation)    (None, 5, 128)            0
_________________________________________________________________
dropout_4 (Dropout)          (None, 5, 128)            0
_________________________________________________________________
flatten_2 (Flatten)          (None, 640)               0
_________________________________________________________________
dense_2 (Dense)              (None, 8)                 5128
_________________________________________________________________
activation_6 (Activation)    (None, 8)                 0
=================================================================
Total params: 87,944
Trainable params: 87,944
Non-trainable params: 0
_________________________________________________________________
```

Fig. 13 CNN Architecture

Following, we have applied another dropout and then flatten the output to make it compatible with the next layers. Finally, we applied one Dense layer (fully connected layer) with a softmax activation function, varying the output size from 640 elements to 8 and estimating the probability distribution of each of the classes properly encoded (Neutral =0; Calm =1; Happy =2; Sad =3; Angry =4; Fearful =5; Disgust =6; Surprised=7).

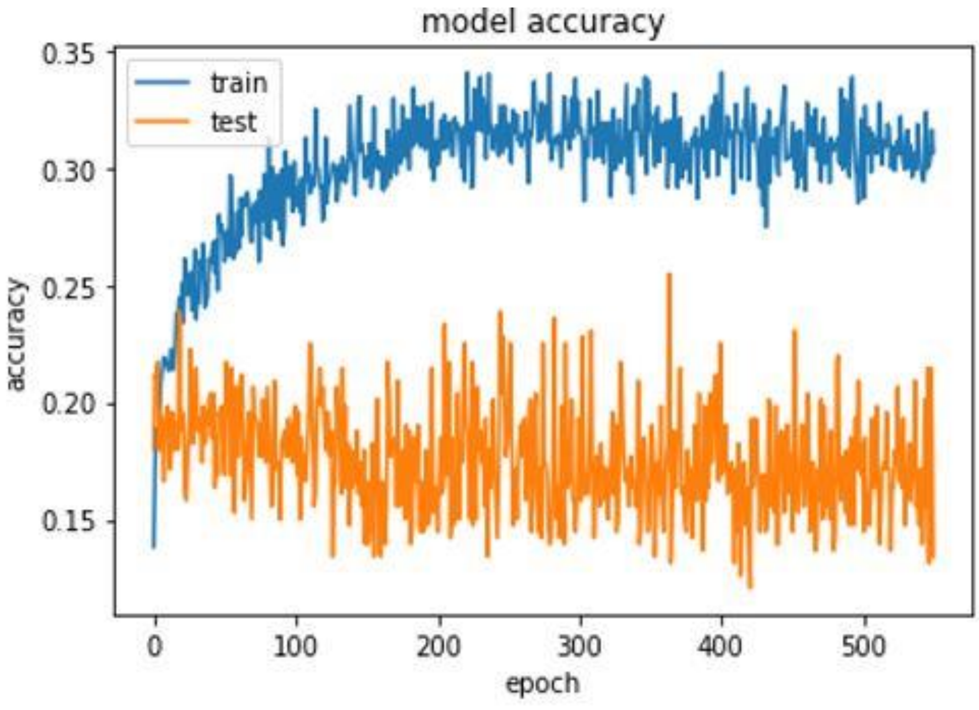
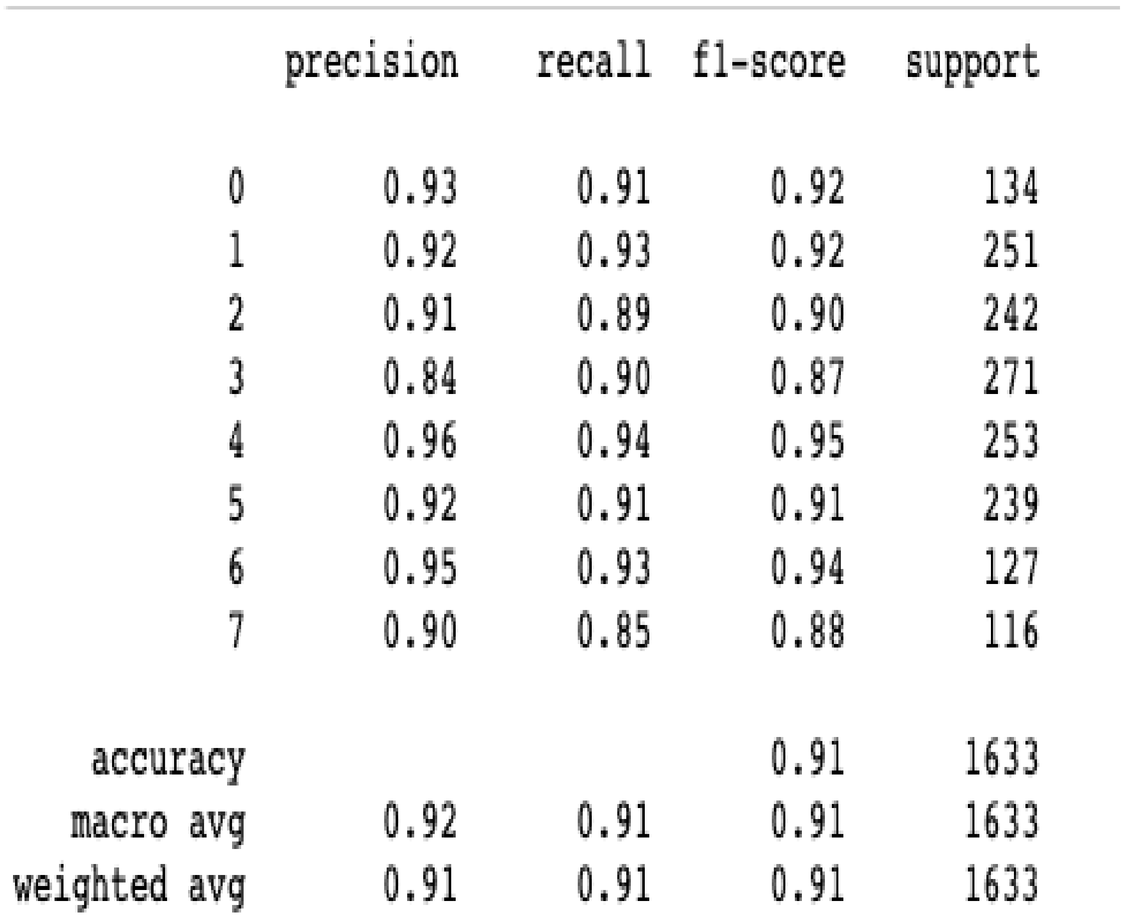


Fig. 14 Train-Test accuracy vs Epochs



```
              precision    recall  f1-score   support

           0       0.93      0.91      0.92       134
           1       0.92      0.93      0.92       251
           2       0.91      0.89      0.90       242
           3       0.84      0.90      0.87       271
           4       0.96      0.94      0.95       253
           5       0.92      0.91      0.91       239
           6       0.95      0.93      0.94       127
           7       0.90      0.85      0.88       116

    accuracy                           0.91      1633
   macro avg       0.92      0.91      0.91      1633
weighted avg       0.91      0.91      0.91      1633
```

Fig. 15 Results on the model on the test set per each class (Neutral =0; Calm =1; Happy =2; Sad =3; Angry =4; Fearful =5; Disgust =6; Surprised=7)

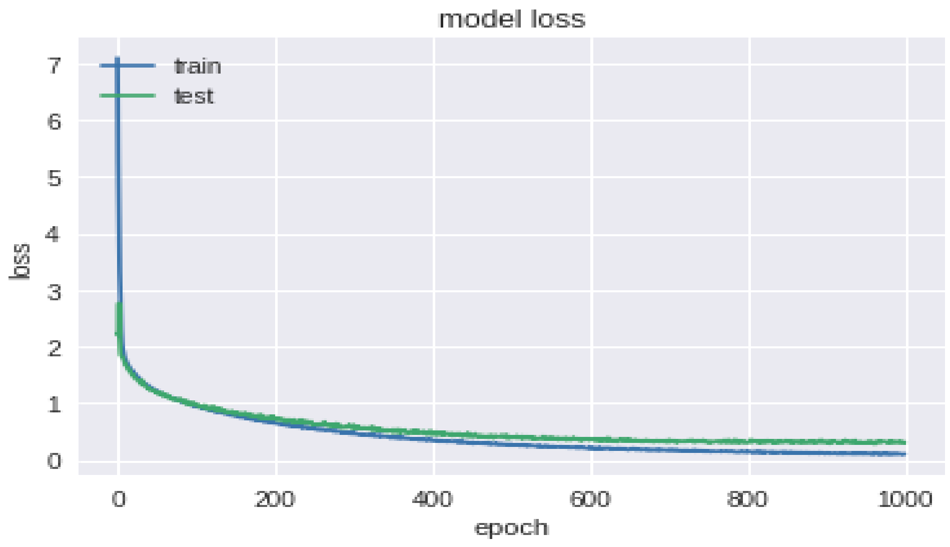


Fig. 16 CNN Model Loss

8. RESULT

In this work, we presented an architecture based on deep neural networks for the classification of emotions using audio recordings from Ryerson Audio-Video Database of Emotional Speech and Song (RAVDESS). The model has been trained to classify eight different emotions (neutral, calm, happy, sad, angry, fearful, disgust and surprised). We carried out the training using three different models – MLP model, LSTM model and CNN model.

The CNN model provided the best outcome with an overall F1 score of 0.91. Fig. 15 shows the values of precision, recall and F1 obtained for each of the emotional classes. These results show us that precision and recall are very balanced, allowing us to obtain F1 values distributed around the value 0.90 for almost all classes. We trained the CNN model that uses 1D CNNs, max-pooling operations and Dense Layers to estimate the probability of distribution of annotation classes correctly. The small variability of F1 results point out the robustness of the model that effectively manages to correctly classify emotions in eight different classes.

The classes "Sad" and "Surprised" are the ones in which the model is less accurate (if F1 score is considered as the parameter to check). It is to be noted that as the number of classes increase it becomes complex to determine the classes but nevertheless the model performs good given the number of classes to determine.

A further index of model reliability can be found in Fig. 16 and it is possible to observe how the value of loss (error in the accuracy of the model) tends to decrease both on the test set and on the training set up to the 1000th epoch. The decrease is less evident from the 400th epoch but still perceptible. It can also be observed that the difference between accuracy of train and test set is very less and same goes for loss between the training and test data set, hence it can be concluded that the model is not overfitting and can also be confirmed with the F1 values. To obtain such result, we extracted the MFCC features (spectrum of a spectrum) from the audio files used for the training.

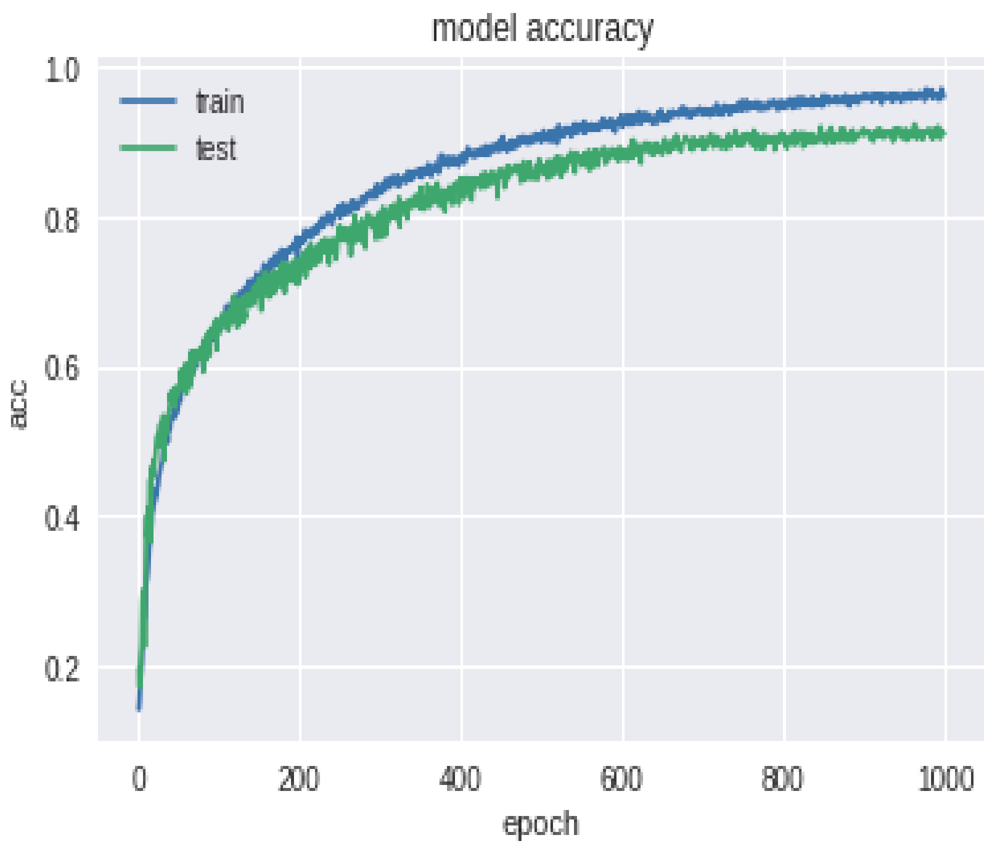


Fig. 17 Model Accuracy

9. CONCLUSION

In this work, we presented an architecture based on deep neural networks for the classification of emotions using audio recordings from the Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS). The model has been trained to classify seven different emotions (neutral, calm, happy, sad, angry, fearful, disgust, surprised) and obtained an overall F1 score of 0.91 with the best performances on the angry class (0.95) and worst on the sad class (0.87). To obtain such result, we extracted the MFCC features (spectrum of-a-spectrum) from the audio files used for the training. On the above representations of input data, we trained a deep neural network that uses 1D CNNs, max-pooling operations and Dense Layers to estimate the probability of distribution of annotation classes correctly. As baseline for our task, we considered the MLP model on the same dataset achieving an average F1 score of 0.75 over the 8 classes. After the MLP model, we trained the dataset using a LSTM model that achieved an F1 score of 0.78. Our final choice was a CNN model that obtained a F1 score of 0.91 on the test set.

The good results obtained suggested that such approaches based on deep neural networks are an excellent basis for solving the task. They are general enough to work in a real application context correctly.

10. Live Prediction

Since, the dataset used (RAVDESS) is made from actors from European and British accent, there might be certain differences to prediction based on the accent of the recorded voice. Much time was spent in finding dataset relevant to the Indian accent but since it is not available and making a dataset for the same would be a very difficult task given the number of audio clips required to train the model, this approach was not taken into consideration.

Initially, after training and testing the model, the model has been saved to the working directory, which can be later used to make predictions. The accuracy of the model can be confirmed by checking the prediction to the actual emotion of the recorded and voice.

The following have been performed to make live predictions using the model:-

1. Firstly, the audio is recorded using the PyAudio library from Python.
2. The recorded audio is sampled at 44100 Hz rate and is recorded for a period of 3 seconds matching with that of the dataset used for training.
3. The recorded audio is stored in the same directory with the name of “output10.wav” file.
4. MFCC of the recorded audio file is extracted.
5. The features extracted (MFCC) is then passed to the model to make predictions.
6. The output to the user is then given by converting the class to emotion (Neutral =0; Calm =1; Happy =2; Sad =3; Angry =4; Fearful =5; Disgust =6; Surprised=7).

11. References

[1] DAVIS, S., AND MERMELSTEIN, P. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. IEEE transactions on acoustics, speech, and signal processing 28, 4 (1980), 357–366.

[2] EKMAN, P. Basic emotions. Handbook of cognition and emotion 98, 45-60 (1999), 16.

[3] GOODFELLOW, I., BENGIO, Y., COURVILLE, A., AND BENGIO, Y. Deep learning, vol. 1. MIT press Cambridge, 2016.

[4] HAYNES, J.-D., AND REES, G. Neuroimaging: decoding mental states from brain activity in humans. Nature Reviews Neuroscience 7, 7 (2006), 523.

[5] HUANG, X., ACERO, A., HON, H.-W., AND FOREWORD BY-REDDY, R. Spoken language processing: A guide to theory, algorithm, and system development. Prentice hall PTR, 2001.

[6] IQBAL, A., AND BARUA, K. A real-time emotion recognition from speech using gradient boosting. In 2019 International Conference on Electrical, Computer and Communication Engineering (ECCE) (2019), IEEE, pp. 1–5.

[7] JANNAT, R., TYNES, I., LIME, L. L., ADORNO, J., AND CANAVAN, S. Ubiquitous emotion recognition using audio and video data. In Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers (2018), ACM, pp. 956–959.

[8] LECUN, Y., BENGIO, Y., ET AL. Convolutional networks for images, speech, and time series. The handbook of brain theory and neural networks 3361, 10 (1995), 1995.

[9] LIVINGSTONE, S. R., AND RUSSO, F. 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 13, 5 (2018), e0196391.

[10] LOGAN, B., ET AL. Mel frequency cepstral coefficients for music modeling. In ISMIR (2000), vol. 270, pp. 1–11.

[11] MUDA, L., BEGAM, M., AND ELAMVAZUTHI, I. Voice recognition algorithms using mel frequency cepstral coefficient (mfcc) and dynamic time warping (dtw) techniques. arXiv preprint arXiv:1003.4083 (2010).

[12] NAIR, V., AND HINTON, G. E. Rectified linear units improve restricted boltzmann machines. In Proceedings of the 27th international conference on machine learning (ICML-10) (2010), pp. 807–814.

[13] PEDREGOSA, F., VAROQUAUX, G., GRAMFORT, A., MICHEL, V., THIRION, B., GRISEL, O., BLONDEL, M., PRETTENHOFER, P., WEISS, R., DUBOURG, V., VANDERPLAS, J., PASSOS, A., COURNAPEAU, D., BRUCHER, M., PERROT, M., AND DUCHESNAY, E. Scikitlearn: Machine learning in Python. Journal of Machine Learning Research 12 (2011), 2825–2830.

[14] PLATT, J. C., CRISTIANINI, N., AND SHAWE-TAYLOR, J. Large margin dags for multiclass classification. In Advances in Neural Information Processing Systems 12, S. A. Solla, T. K. Leen, and K. Muller, Eds. MIT Press, 2000, pp. 547–553. ¨

[15] POLIGNANO, M., DE GEMMIS, M., NARDUCCI, F., AND SEMERARO, G. Do you feel blue? detection of negative feeling from social media, 2017.

[16] ZHANG, B., ESSL, G., AND PROVOST, E. M. Recognizing emotion from singing and speaking using shared models. In 2015 International Conference on Affective Computing and Intelligent Interaction (ACII) (2015), IEEE, pp. 139–145.

[17] Kim, Y.: Convolutional Neural Networks for Sentence Classification. In: Proceedings of the 2014 Conference on EMNLP, pp. 1746–1751 (2014).

[18] Zhang, X., Zhao, J., LeCun, Y.: Character-level Convolutional Networks for Text Classification. In: Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH, pp. 3057–3061 (2015).

[19] Hinton, G.: Deep neural networks for acoustic modeling in speech recognition. In: IEEE Signal Processing Magazine 29, pp. 82–97 (2012).

[20] Lee, J., Tashev, I.: High-level feature representation using recurrent neural network for speech emotion recognition. In: INTERSPEECH (2015).

[21] Jin, Q., Li, C., Chen, S., Wu, H.: Speech emotion recognition with acoustic and lexical features. In: IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 4749-4753 (2015).

[22] Ververidis, D., Kotropoulos, C.: Fast and accurate sequential floating forward feature selection with the bayes classifier applied to speech emotion recognition. In: Signal Processing, vol. 88, no. 12, pp. 2956– 2970 (2008).

[23] Mao, X., Chen, L., Fu, L.: Multi-level Speech Emotion Recognition Based on HMM and ANN. In: WRI World Congress on Computer Science and Information Engineering, pp. 225-229 (2009).

[24] Ntalampiras, S., Fakotakis, N.: Modeling the Temporal Evolution of Acoustic Parameters for Speech Emotion Recognition. In: IEEE Transactions on Affective Computing 3.99, pp. 116-125 (2012).

[25] Hao, H., Xu, M. X., Wu, W.: GMM Supervector Based SVM with Spectral Features for Speech Emotion Recognition. In: IEEE International Conference on Acoustics, Speech and Signal Processing – ICASSP, pp. 413-416 (2007).

[26] Neiberg, D., Laskowski, K., Elenius, K.: Emotion Recognition in Spontaneous Speech Using GMMs. In: INTERSPEECH (2006).

[27] Wu, C. H., Liang, W. B.: Emotion Recognition of Affective Speech Based on Multiple Classifiers Using Acoustic-Prosodic Information and Semantic Labels. In: IEEE Transactions on Affective Computing 2.1, pp. 10-21 (2011).

[28] Kim, Y., Lee, H., Provost, E. M.: Deep learning for robust feature generation in audiovisual emotion recognition. In: IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 3687-3691 (2013).

[29] Zheng, W. L., Zhu, J., Peng, Y.: EEG-based emotion classification using deep belief networks. In: IEEE International Conference on Multimedia & Expo, pp. 1-6 (2014).

[30] Han, K., Yu, D., Tashev, I.: Speech emotion recognition using deep neural network and extreme learning machine. In: INTERSPEECH (2014).

[31] Fayek, H. M., Lech, M., Cavedon, L.: Towards real-time Speech Emotion Recognition using deep neural networks. In: International Conference on Signal Processing and Communication Systems, pp.1-5, (2015).