CentraleSupélec

Deciphering Non Verbal Behaviours:

Speech Emotion Recognition

DAUMAS CARNEIRO Marina
GRANDIÈRE Gaelle
KEMPF Amélie
NORODOM Thomas
VALVERDE REIS ZREIK Ingrid
ZHANG Jiayi

Project supervisor: Samir Sadok

Date
31/03/2023

Contents

Abstract

This report explores the field of Speech Emotion Recognition (SER) and its practical implementation through various techniques. Emotions play a crucial role in human communication, and accurate recognition of emotions from speech has numerous benefits, including improving communication in healthcare, education, and business, and creating more realistic and immersive emotional experiences in the entertainment industry. The primary goal of this project is to classify emotions from speech samples acquired from the IEMOCAP datset using machine learning algorithms such as the MLP Classifier, LSTM, and Query2Label. These methods are compared and evaluated to determine which approach yields the most accurate results.

1 Introduction

Nonverbal communication is a fundamental aspect of human interaction. It involves the use of facial expressions, body language, tone of voice, and other nonverbal cues to convey meaning and emotion. Deciphering nonverbal behaviours is a complex and challenging task, as it involves interpreting subtle and often unconscious signals that are not always easy to understand. However, it is a crucial skill in many areas, including interpersonal relationships, business interactions, and law enforcement. In recent years, advances in technology and artificial intelligence have led to the development of new tools and techniques for analyzing nonverbal behaviour, including machine learning algorithms and computer vision systems. These technologies have the potential to revolutionize the way we understand and interpret nonverbal communication, providing new insights and improving our ability to communicate effectively.

The implementation of these technologies has become a commonplace across numerous industries, with many companies already specializing in the field. Emotion AI has various practical applications such as mental health monitoring through mobile apps such as CompanionMx, identifying high-risk patients in hospitals, and studying pregnancy-related risk factors. It also has financial benefits, with companies utilizing emotion recognition to determine how customers respond to their products and services. Other companies, such as audEERING or Affectiva, are specialised in this area and now offer their expertise.

Despite its increasing use and implementation, deciphering nonverbal behaviours is a complex and challenging task due to a variety of factors. Firstly, nonverbal cues can vary widely across different cultures, making it difficult to interpret them accurately without a deep understanding of cultural norms. Additionally, nonverbal cues can be highly dependent on the context in which they are used and may be ambiguous or difficult to decipher in certain situations. Moreover, people vary in their use of nonverbal cues, making it difficult to determine the true meaning of these behaviours without taking into account individual differences. Finally, while technological advances in computer vision and machine learning have enabled automated analysis of nonverbal behaviours, there is still a risk of inaccuracies or misinterpretations in these systems. Bypassing the challenges, the potential applications and insights that can be gained from deciphering nonverbal behaviours make this a critical area of research and development.

This report provides an implementation of different models and techniques to automatically determine the emotions of people in action, particularly focused on the study of their voices.

1.1 Motivation

As seen in the section above, Speech Emotion Recognition is a rapidly growing field of research in computer science. Emotions play a vital role in human communication and are an essential aspect of our daily lives. They are a medium by which one expresses their

feelings and state of mind.

Keeping in mind that recognizing and predicting emotions accurately from speech is a challenging task due to speech-related factors, the primary goal of this project is to classify emotions from a given speech sample in the most appropriate manner. In this paper, we aim to explore different methods of predicting emotions, including the MLP Classifier, the LSTM, and the Query2Label (Q2L). These methods will be compared and evaluated to determine which approach yields the most accurate results. By contributing to this exciting and rapidly evolving field of research, we hope to advance our understanding of speech emotion recognition and its potential applications in various domains.

The emotions we seek to classify in this research include happy, angry, neutral, sad, and excited. Each of these emotions has unique characteristics that make them distinct from one another. Developing an accurate and efficient method of classifying these emotions has numerous applications, including improving communication in various domains such as healthcare, education, and business, enhancing the user experience of virtual assistants and chatbots, and creating more realistic and immersive emotional experiences in video games and the entertainment industry.

1.2 State of the art

Speech emotion recognition (SER) has been a growing sector for the last few years and takes an important role in the improvement of human-computer interaction (HCI) application [6]. Many different SER have been developed using diverse methods and many focus on the use of audio data samples [10]. Analysing pitch, pause in sentences or small frequency variation, it is possible to detects emotions such as anger, joy or sadness[4].

To analyse the audio extract, feature treatment as MEL Frequency Cepstral Coefficient (MFCC), Contrast, Mel Spectrograph Frequency, Chromagram and Tonnetz [10],[3],[9] can be used. MFCC is especially efficient thanks to phonetic features that can be expressed as power cepstrum coefficient that increase the accuracy over utterance[13]. Wavelet transform, another feature treatment method can reveal more information than a Fourier transform. This method can be combined with MFCC to increase the precision of the result after going through different models[7].

The pre-processed date then go through different models to try detect emotion. One of the first efficient model was Markov model based on a probabilistic approach, associating a probability to each pattern, and computing a maximum to give a result among different emotion input[7]. But many other models using artificial intelligence exist. One of the most used is Convolutional Neural Network (CNN). They allow confrontation and pattern recognition through the different hidden layers of neurons with associated activation functions [14]. This method is efficient for both supervised (with label) and unsupervised (without label) learning. Using an incremental method, CNN can be developed in very deep neural network allowing a fine recognition pattern and an hight efficiency[9]. But it is only one among many others such as Gaussian mixture model (GMM), k-nearest neighbours (KNN), artificial

neural network (ANN) and support vector machine (SVM)[6], each with its advantages and its downsides. A last method is using a transformer decoder that can classify different emotions by extracting determinative features, attributing very specific information sample to each class[11]. It uses a backbone structure with a binary classification approach which allow it to be quite simple and efficient.

The choice of database is also an important factor and today, some of them being Interactive Emotional Dyadic Motion Capture (IEMOCAP) dataset , Ryerson Audio-Visual Database of Emotional Speech and Song dataset (RAVDESS) [10],[9]. It is important to use a large dataset for both training and testing. A wide range of them are available or you can create new one if necessary. Indeed, datasets are various and must be adapted for the targeted audience. But for work without lots of datasets such as dialects or specific ethnic patterns, a support vector machine can be the most efficient, being able to distinguish four different emotions with limited training due to limited data[3]. The model has to be adapted to the targeted audience because different languages do not interact the same with different models. A way to improve the training part allowing better result for CNN could be by using Stochastic Gradient Descent (SGD)[1]. Ruder shows that you can then optimize a CNN training using this method, allowing the CNN to be more efficient at the end and overall give better result in term of efficiency.

Finally, even if speech recognition alone can be quite effective, Blatliner and al. show that combining different kinds of features (audio and text) allow to achieve higher accuracy result than only using one of the features[5]. It combines the advantages of the different feature analysis and let them complement each other.

1.3 Approach

1.3.1 Version 0

In order to get ourselves acquainted with the libraries and models available to recognize emotion in speech, we ran the first test which we will call Version 0. In this version, we have used the libraries *librosa*, *soundfile*, and *sklearn* (among others) to build a model using a *MLPClassifier*.

The first step that followed was the pre-processing, that consisted in loading the sound files from the IEMOCAP dataset and extracting features from it. A function was defined to extract the *mfcc* (Mel frequency Cepstral Coefficient), *chroma*, and *mel* (Mel Spectrogram Frequency) features from the sound files. For each feature of the three, it calls the corresponding function from *librosa.feature* (eg- *librosa.feature.mfcc* for mfcc), and gets the mean value. Those methods for feature extraction are specified in the Section 2 of this document.

We created a data loading function that extracts features and appends them to a list x, while adding the corresponding emotion (e.g., angry, happy, sad, neutral, frustrated, excited, fearful, surprised, disgusted, or other) to a separate list y. Thus, x contains the features and y contains the emotions.

Next, we split the feature and emotion lists x and y into training and testing sets. Specifically, we used four sessions of the IEMOCAP dataset for training, and one session for testing. Afterwards, we initialized an MLPClassifier and trained the model on the training data.

We used cross-validation to access our model's performance on unseen data. We partitioned the dataset into 5 folds, using one session for testing and the others for training in each fold. We computed the performance on each fold and then averaged the results for an overall estimate.

This version showed weak results, as displayed in the Section 4, specially due to the fact that we used the mean operation to reduce the temporal lenght.

The complete pipeline is shown in the Figure 1.

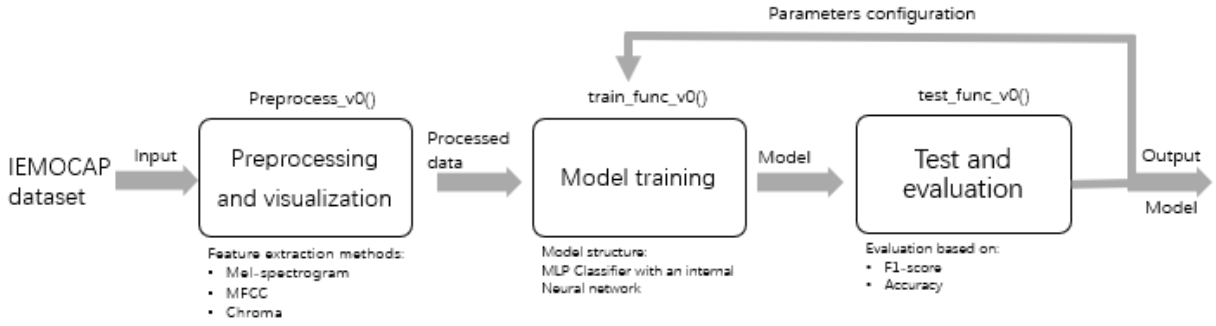


Figure 1: Pipeline of the approach Version 0.

1.3.2 Version 1

To improve the Version 0, we made several changes inside Version 1. First, we decided to pre-define the sequence length separately for each of the four feature extraction methods: *mfcc*, *spectrogram*, *melspec*, and *w2v*. Next, we used a Long Short-Term Memory (LSTM) neural network to find the features, followed by MLP layers to classify the emotions. Since *sklearn* lacks an implementation of LSTM models, we used *PyTorch* instead.

We also decided to focus on only five emotions instead of the nine found in the dataset due to the lack of available data for other emotions. We also hoped to achieve better results by concentrating on a smaller number of emotions. The selected emotions were: happiness, sadness, neutral, anger and excitement. We put happiness and excitement under the same label since they are closely related.

To reduce processing time, we opted to pre-process the data and store it in *npy* files. This way, during training, we can load the pre-processed data and save significant time. We

saved the pre-processed data in separate files for the training and test sets. Different feature extraction methods were used to create different pre-processed datasets.

We conducted multiple tests to optimize certain parameters, particularly the learning rate and the sequence length of input data. We also compared the results using different feature extraction methods. Although cross-validation and calculating average results would yield the most accurate results, time constraints limited our approach. We used session one as the test dataset and the remaining sessions as the training set. This simplification did not significantly impact our results.

The complete pipeline is shown in the Figure 2.

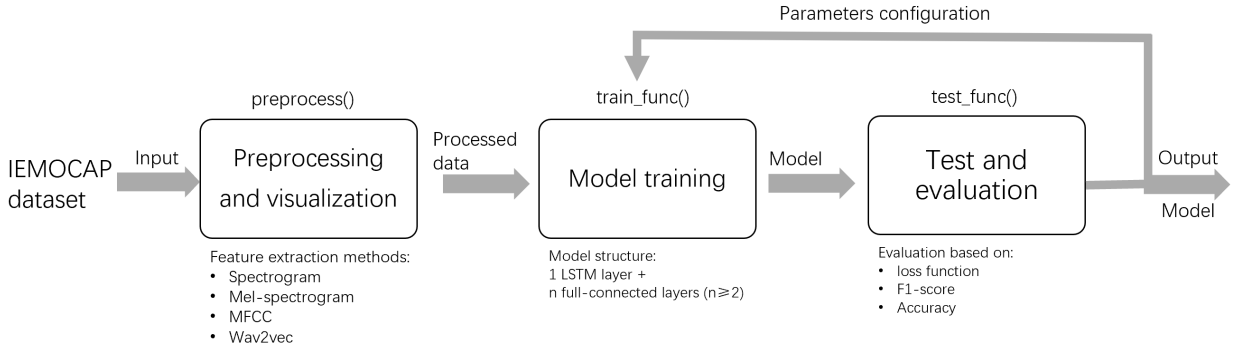


Figure 2: Pipeline of approach Version 1.

1.3.3 Version 2

The approach proposed in this version uses the Query2Label (Q2L) for speech emotion recognition. Q2L is a machine learning approach that uses the Transformer encoder-decoder architecture to learn label representations and classify data. This approach leverages Transformer decoders to query the existence of a class label. As shown in the Figure **??**, after extracting features of an input audio signal, by either *melspec* or *wav2vec* as done in the Version 1, we use learnable label embeddings as queries to probe and pool class-related features via the cross-attention module in Transformer encoders.

The built-in cross-attention mechanism, which is called encoder-decoder attention in [15], makes Transformer decoder a perfect choice for adaptively extracting desired features. By treating each label class as a query in a Transformer decoder, we can perform cross-attention to pool related features for the subsequent binary classification. Another advantage of the Transformer is its multi-head attention mechanism, which can extract features from different parts of an object class [11].

The Query2Label approach has been shown to achieve state-of-the-art results on various benchmark datasets, particularly in situations where the number of labels is large and the class imbalance is severe.

The complete pipeline is shown in Figure 3.

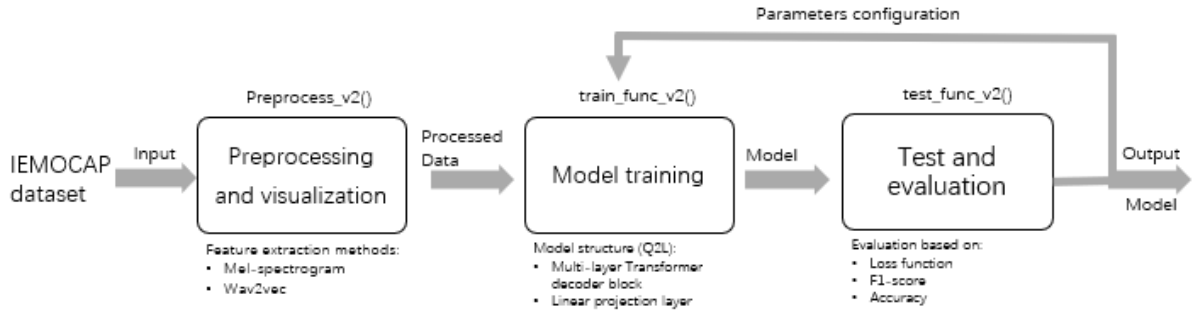


Figure 3: Pipeline of approach version 2.

2 Speech audio representation

To analyse the different sounds provided by the database, we must first pre-process the data so that it can be put as entry to our model. This part will describe the different methods to represent features that were used and compared to try to find the most efficient and easy to use for our models.

2.1 Spectrogram

The first method used to pre-process the data is the spectrogram. It is a widely used method for pre-processing audio recordings in sound analysis. It represents a visual display of the frequency spectrum of a sound signal as it changes over time.

To generate a spectrogram, we first divide the audio signal into small time frames, typically on the order of a few milliseconds. We then calculate the frequency spectrum of each interval using a Fourier transform. The resulting frequency spectrum is represented as a two-dimensional plot, with frequency on the vertical axis and time on the horizontal axis. As can be seen in Figure **??**, the intensity of each frequency component is represented by a color and shading, with higher intensity components appearing brighter.

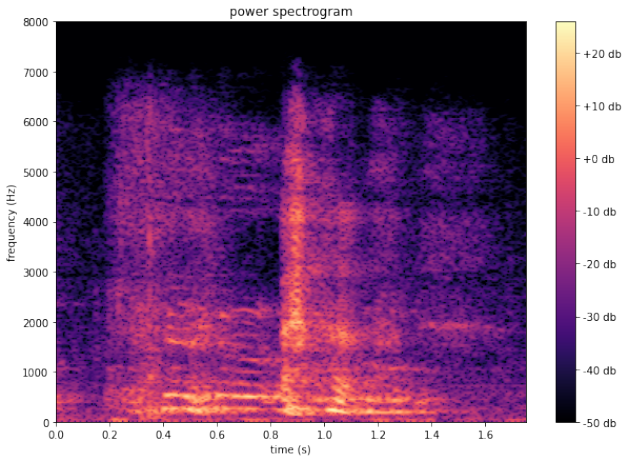


Figure 4: Spectrogram representation example.

2.2 Mel-Spectrogram

The Mel-Spectrogram is a modified version of the traditional spectrogram that uses the Mel frequency scale to better reflect the way humans perceive sound. This method is particularly used for speech and music recognition.

To generate a Mel-Spectrogram, we start the way we did for the classic spectrogram by dividing the audio signal into small time frames. The main difference is that we then apply a

series of filters to the frequency spectrum of each interval. These filters are designed to mimic the non-linear nature of human hearing, emphasizing lower frequencies and de-emphasizing higher frequencies. The result looks like a classic spectrogram with the same axis and system of representation (Figure 5).

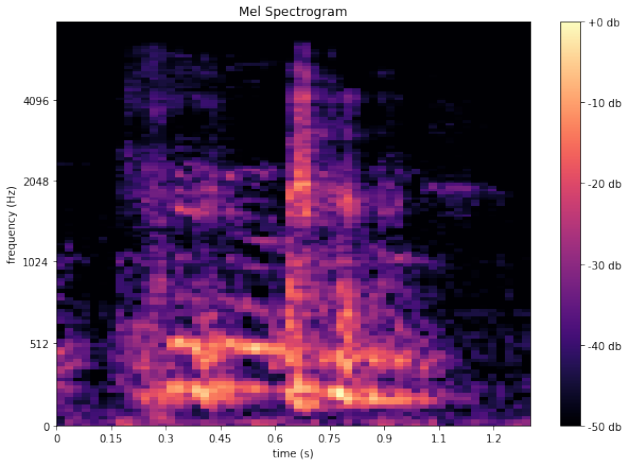


Figure 5: Mel-spectrogram representation example.

2.3 MFCC (Mel-Frequency Cepstral Coefficients)

The mel-frequency cepstrum (MFCC) is a commonly used representation of the short-term power spectrum of a sound in sound processing. It is derived from a linear cosine transform of a log power spectrum that is computed on a nonlinear mel scale of frequency. The frequency bands used in the MFCC are equally spaced on the mel scale, which approximates the human auditory system's response more accurately than the linearly-spaced frequency bands used in the normal spectrum.

What we use here are mel-frequency cepstral coefficients (MFCCs), which are a set of coefficients that are used to compute the MFCC. These coefficients are obtained from a cepstral representation of the audio clip that is based on a "spectrum-of-a-spectrum" concept. The process for deriving MFCCs involves taking the Fourier transform of a signal, mapping the powers of the spectrum onto the mel scale using overlapping windows, taking the logs of the powers at each mel-frequency, and finally, taking the discrete cosine transform of the resulting list of mel log powers to obtain the MFCCs.

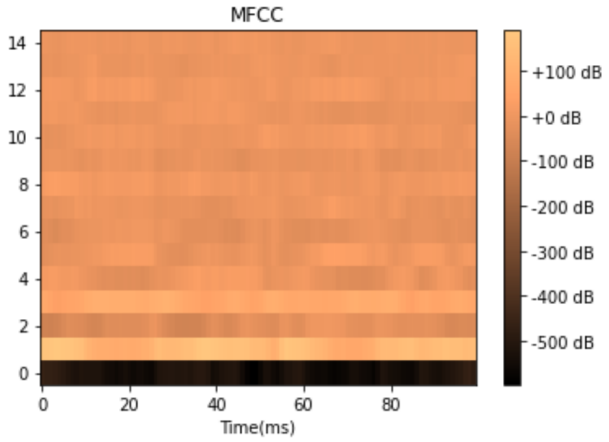


Figure 6: MFCC representation example

2.4 Wav2vec

Wav2vec stands for ".wav" (for raw audio) to "vec" (vector). In opposition to the other feature extraction methods, Wav2vec is a method using AI algorithms to extract the features of the raw audio, instead of mathematical operations. This method aims to extract new types of input vectors for acoustic models from raw audio, using pre-training and self-supervised learning.

It is largely used in speech processing, and several versions exist, such as vq-wav2vec and wav2vec 2.0

Wav2vec is a pre-trained model. Pre-training is the fact of training a first neural network on a task where lots of data are available, saving the weights, and creating a second neural network by initializing the weights as the ones saved from the first one.

The wav2vec model aims to predict the next observations of a speech sample (which requires a very good understanding of the audio sample) : it is self-trained.

The model is made of two CNNs. The first one is an encoder network and lowers the dimensionnality of the speech sample, to make it easier to work with. The second one is the context network, wich tries to predict the next values of the audio sample.

2.5 Comparison among representations

In the Table 1, there are displayed advantages and drawbacks of the audio representa-

tion methods explained above.

Feature	**Advantages**	**Disadvantages**
Spectrogram	Captures time-frequency characteristics, useful for visualizing frequency content, identifies temporal and spectral patterns	May not capture finer details important for speech recognition, can be computationally expensive
Mel-spectrogram	Logarithmic frequency axis approximates human perception, captures important spectral features for speech	Can result in high-dimensional feature representation, computationally expensive
MFCC	Captures important spectral features of speech, reduces feature dimensionality compared to spectrogram/mel-spectrogram	Transformation process can be sensitive to hyperparameters, may not be optimal for non-speech audio tasks
Wav2Vec	Self-supervised learning approach, can learn language/accent/speaker specific features, can be used for a wide range of audio tasks	Requires large amount of training data, computationally expensive, requires specialized hardware for large models

Table 1: Comparative table of advantages and drawbacks of pre-processing methods.

Overall, the choice of audio feature extraction technique depends on the specific task, the nature of the audio data and the computational resources available for feature extraction and processing. In the Section 4 we are gonna display the results in terms of $f1$ score, loss function and accuracy found when training and testing our models for each technique as the data pre-processing step.

3 Model

3.1 Version 0 - MLPClassifier

The model used in the Version 0 was the Multi-layer Perceptron Classifier. MLP classifiers can be a good model choice for recognizing emotions from speech because they are capable of learning complex, non-linear relationships between input features and target labels. Speech data typically consists of high-dimensional features that are difficult to model with simple linear classifiers. MLPs can handle these high-dimensional feature spaces and learn non-linear relationships between the input and output [2].

The MLPClassifier is a feedforward ANN model. A feedforward artificial neural network (ANN) is a type of neural network where the data flows only in one direction, from the input layer to the output layer. In this architecture, the input layer receives data and passes it to the hidden layers, which perform computations and then pass the results to the output layer [8].

It optimizes the log-loss function using the stochastic gradient descent [1]. We initially chose it because it works pretty well on relatively large datasets, like the IEMOCAP one, in terms of both training time and validation score.

Unlike some other machine learning models, such as decision trees or logistic regression, the MLPClassifier has an internal neural network architecture that can learn complex relationships between input features and output labels for the purpose of classification. The MLP Classifier implements a Multi-Layer Perceptron (MLP) algorithm and trains the Neural Network using Backpropagation [10].

In the proposed methodology for Speech Emotion Recognition, the Multi-Layer Perceptron Network will have one input layer, one hidden layer with 300 neurons and one output layer. The input layer will take as input, the features that are extracted from the audio file. The hidden layer uses an activation function to act upon the input data and to process the data. The activation function used is the RELU function. The output layer brings out the information learned by the network as output. This layer classifies and gives output of the predicted emotion, according to the computation performed by the hidden layer. This network is represented in Figure 7.

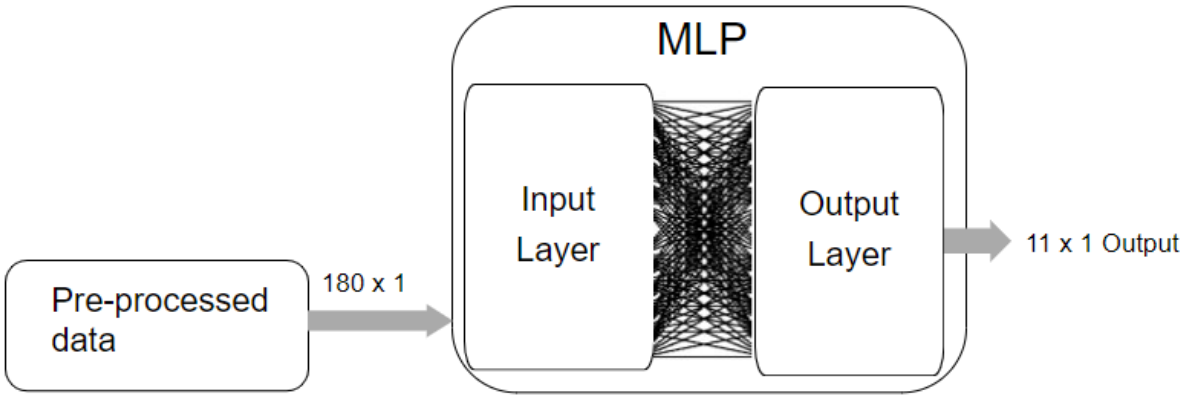


Figure 7: Network architecture for the MLP model.

3.2 Version 1 - LSTM+2 layers of MLP

Since audio signals are sequential, we decided to add a LSTM layer in our model. Long Short-Term Memory (LSTM) is a type of recurrent neural network that is particularly well-suited for tasks that involve sequential data, and its performance has been proved by other researchers' work [16]. Contrast to feedforward neural networks which only takes the current state as input, recurrent neural networks (RNN) take the current and previous states as input, thus allowing the network to maintain a memory of previous inputs and computations. For our project, the audios are cut into small pieces thus the changes of emotions may be ignored by analysing data by data using the feedforward neural network. Instead, by applying the LSTM, changes of emotions over time can be effectively captured by the model.

To do the emotion classification task, the output of LSTM is then fed as input to classification layers, which are MLP classifiers in our model. In the context of speech emotion recognition, MLP classifiers can be trained to classify emotions based on various acoustic features, such as pitch, loudness, and spectral information. By using multiple hidden layers, MLPs can capture higher-order features and dependencies in the data, allowing for more accurate emotion recognition [2].

This network architecture was identical across all feature extraction techniques, with the only difference being the size of the input. The number of features per frame varied depending on the specific used feature extraction method (mfcc=15, melspec=80, spec=513, wav2vec=512).The frames containing the extracted features were fed as input to the LSTM. Each block of the LSTM received a frame along with the output of the previous block as input. The final output of the last LSTM block was passed to the MLP, which consisted of two fully connected layers. The second fully connected layer of the MLP produced a 4×1 vector that represented the probabilities of each emotion being the correct one. In the training step, the *argmax()* is thus applied on the output vector to get the predicted emotion.

Additionally, one drawback of LSTM is that it is very time-consuming, so it is necessary to utilize batching during the training process. Specially, we set the $batch\_size = 32$, which divided the whole training set into smaller subsets. This approach made the training process more efficient by allowing the model to upgrade its parameters more frequently.

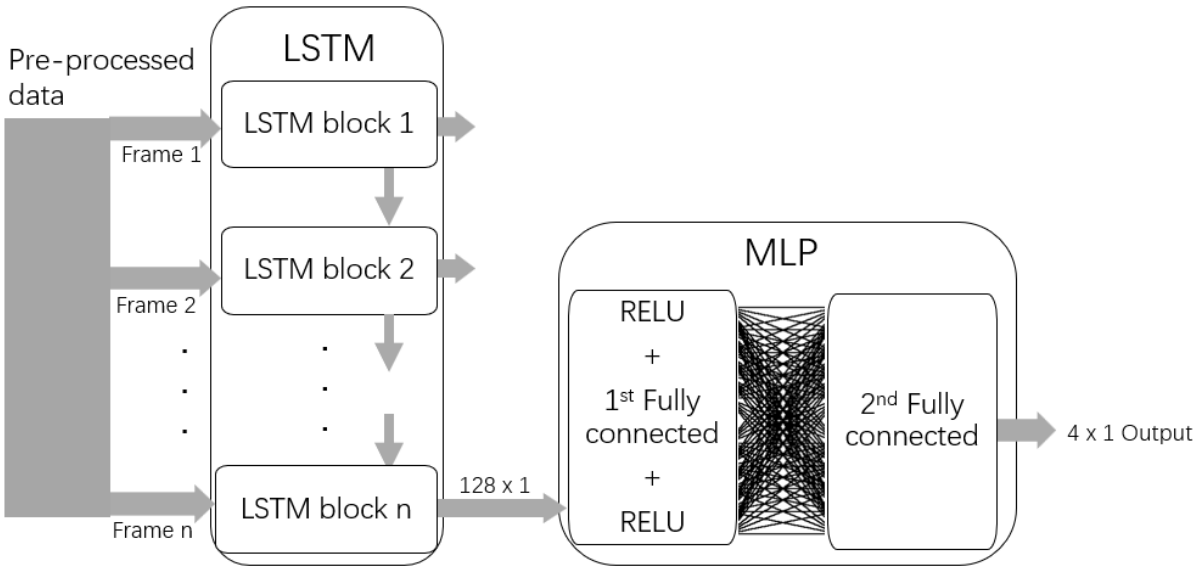


Figure 8: Network architecture for the LSTM+MLP model.

3.3 Version 2 - Q2L

After extracting the features of the input audio signal at each time frame, each label embedding is sent to Transformer decoders to query (by comparing the label embedding with features at each frame to generate attention maps) and pool the desired feature adaptively (by linearly combining the frame features based on the attention maps). The pooled feature is then used to predict the existence of the queried label, by using a linear projection layer. This process can be seen in the Figure 9.

In the query updating step, we usually use label embeddings as queries and perform cross-attention to pool category-related features from the frame features using multi-layer Transformer decoders. However, for our application we only used one decoder. We use the standard Transformer architecture, which has a self-attention module, a cross-attention module, and a position-wise feed-forward network (FFN). As we do not need to perform autoregressive prediction, we do not use attention masks. Thus each category can be decoded in parallel in each layer. The label embeddings are treated as learnable parameters. In this way, the embeddings can be learned end to end from data and model label correlations implicitly [11].

In the feature projection step, we will get queried feature vectors for all the categories

in the last layer (unique). To perform multi-label classification, we treat each label prediction as a binary classification task and project the feature of each class to a logit value using a linear projection layer followed with a sigmoid function [11].

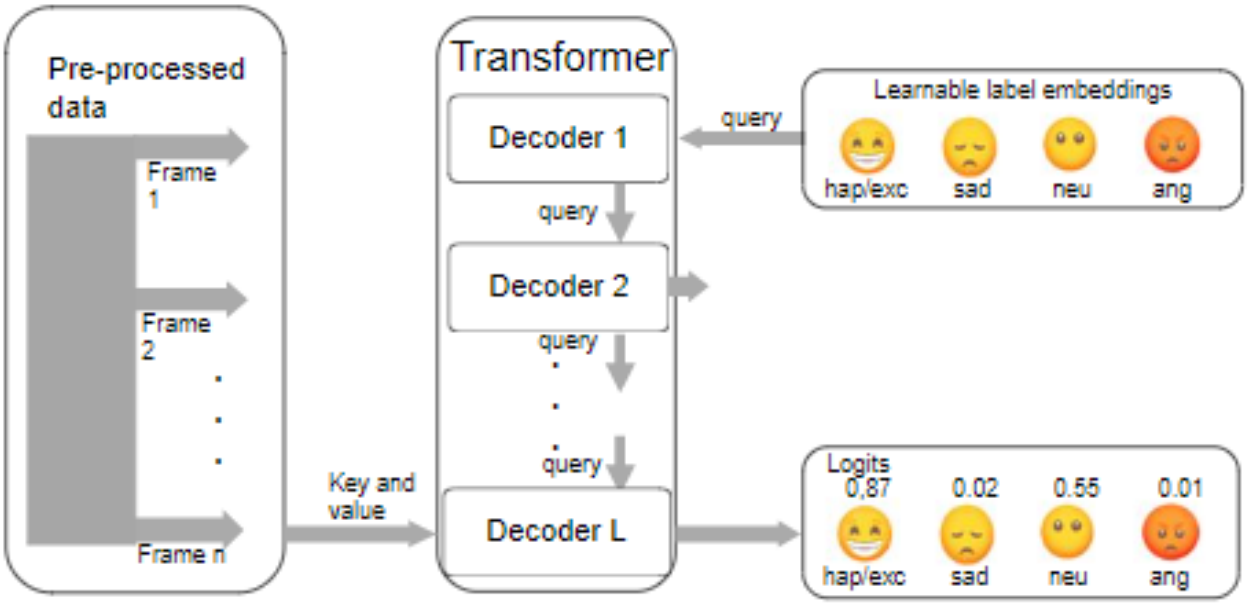


Figure 9: Framework of Query2Label (Q2L).

4 Experiments

4.1 Dataset

The data at the IEMOCAP database is divided into 5 groups called sessions. In each session the actors, always a man and a woman, would change. Because of this configuration it was easy to use the sessions as the groups for cross validation. When working with speech it is recommended that the training and testing sets contain audio files from different people so the model can be trained to be as generic as possible instead of adapting to the way of speaking of a specific person or group.

In addition to pre-processing the data using one of the feature extraction techniques already described, we used the *dataloader* from *pytorch* before feeding it to the network. It menages batches and iterates through the data in the dataset.

4.2 Data setup

The pre-processing of the data before the training was an essential step. For each of the representations used there were parameters we had to set so that the resulting data would have as much useful information as possible. The code below was used to setup the functions that generated the audio representations. The sample rate was kept as 1600 samples per second for all of the audio representations.

```
def melspec(self, signal, sample_rate):
    mel = librosa.feature.melspectrogram(y=signal,
                                         sr=sample_rate,
                                         n_mels = 80,
                                         hop_length=512,
                                         win_length = 1024)
    return mel

def mfcc(self, signal, sample_rate):
    mfcc = librosa.feature.mfcc(signal,
                                sr=sample_rate,
                                n_mfcc=15,
                                n_fft=1024,
                                hop_length=256,
                                win_length = 1024)
    return mfcc


def spec(self, signal, sample_rate):
        X = librosa.stft(signal,
                         n_fft=1024,
                         center=False,
                         hop_length=256,
                         win_length = 1024)

```

```
    X= np.abs(X)**2
    return X

def wav2vec(self, signal, sample_rate):
    wav2vec = self.model_wav2vec.feature_extractor(signal)
    return wav2vec
```

4.3 Speech emotion recognition

In order to find the best possible model we did tests with each of the audio representations. Within a set of tests for one type of representation we performed a second round of tests to define the optimal hyperparameters of the model.

There are many different parameters that could be tuned in order to increase the accuracy of the neural network. The most obvious ones are the number of epochs, learning rate, sequence length, and number of fully connected layers.

An epoch refers to a complete pass through the training dataset during training. The number of epochs controls the duration of training and affects the ability of the model to generalize to new, unseen data. Setting the number of epochs too low can result in underfitting, where the model is not able to learn enough from the training data and has poor performance on both the training and validation datasets. On the other hand, setting the number of epochs too high can result in overfitting, where the model has learned to fit the training data too closely and has poor generalization to new data. Our approach was to set an initial number of epochs and monitor the performance of the model on a validation dataset during training and stop the training when the validation performance stops improving or starts to deteriorate.

The learning rate is a hyperparameter that controls the step size during gradient descent optimization. A high learning rate can lead to unstable training, where the model jumps around the parameter space and fails to converge. On the other hand, a low learning rate can result in slow convergence and may require a large number of epochs to train. Therefore, finding the optimal learning rate is critical for training our speech emotion recognition model.

The sequence length is the number of time steps in an audio recording that the model processes at once. Longer sequence lengths can capture more contextual information from the audio, but they also require more memory and computation. Therefore, choosing the optimal sequence length depends on the trade-off between model performance and computational resources.

The number of fully connected layers in a neural network determines the complexity and capacity of the model. Adding more fully connected layers can increase the model's ability to learn complex representations of the input data, but it can also lead to overfitting if the model is too complex relative to the size of the training data. Therefore, choosing the optimal number of fully connected layers depends on the complexity of the problem and the

size of the training data.

In order to evaluate the performance of our model, we decided to use cross-validation as a technique. There are several reasons why cross-validation is particularly important for speech emotion recognition using the MLPClassifier. Since the MLPClassifier is a flexible model that can learn complex non-linear relationships between the input features and the target labels, it can lead to overfitting, particularly if the model is too complex relative to the size of the dataset. So, cross-validation can help us choose the optimal model complexity by comparing the performance of models with different hyperparameters on the validation folds.

Besides, in speech emotion recognition, we are often interested in building models that can generalize to new speakers who are not included in the training dataset. Cross-validation can help us evaluate the model's ability to generalize to new speakers by splitting the dataset into folds that contain data from different speakers.

4.3.1 Changing the learning rate

The learning rate determines how fast or slow the model will move towards the optimal weights. However, there is a trade-off between the convergence and overshooting. A too high learning rate will make the learning jump over minima but a too low learning rate will either take too long to converge or get stuck in an undesirable local minimum[12]. The range of considered learning rate is generally from 0.1 to 10^{-6}.

For our model, we performed separate tests for each audio representation as the optimal learning rate may be different for different representations. We used a learning rate schedule that started with a high learning rate and decreased it over time to achieve better convergence.

The results found for the Version 1 and 2 are showed in the Table 2 and 3.

The Table 2 presents the results of tests performed on two different versions of a model for audio classification using different representations of audio signals, i.e., spectrogram, melspectrogram, MFCC, and Wav2Vec, under different learning rates. The table shows the accuracy and $f1$ score obtained for each representation under different learning rates.

Version 1

	Learning Rate			
Representation	1e-3	1e-4	1e-5	1e-6
Spectrogram	49,24 (47,40)	50,83 (40,66)	49,48 (39,67)	- (-)
Melspectrogram	46,78 (46,82)	42,42 (34,04)	54,79 (44,28)	- (-)
MFCC	57,81 (46,27)	57,60 (45,67)	54,79 (44,38)	22,30 (43,85)
Wav2vec	55,00 (42,60)	56,60 (46,20)	53,00 (43,50)	- (-)

Table 2: Accuracy[%] (F1 Score[%]) of Version 1, under different learning rates.

Version 2

	Learning Rate			
Representation	1e-3	1e-4	1e-5	1e-6
Melspectrogram + Q2L	53,13 (42,36)	50,73 (34,01)	50,94 (38,25)	51,46 (33,31)
Wav2vec + Q2L	39,90 (14,16)	55,10 (45,41)	54,90 (44,63)	52,08 (41,31)

Table 3: Accuracy[%] and F1 Score[%] of Version 2, under different learning rates.

Figures of loss function are available in appendix A(5). The data suggests that the optimal learning rate may vary for different representations of audio signals. For instance, in Version 1, the optimal learning rate for MFCC representation was found to be 1e-3, while for melspectrogram it was 1e-5 and for spectrogram and wav2vec, it was 1e-4.

In Version 2, the performance of the model was evaluated using melspectrogram and Wav2Vec representations, along with query-to-label (Q2L) embeddings. The results show that the accuracy and F1 score for melspectrogram + Q2L were highest at a learning rate of 1e-5, while for Wav2Vec + Q2L, the optimal learning rate was found to be 1e-4.

In conclusion, the data suggests that the optimal learning rate for different audio representations may vary, and it is essential to perform separate tests for each representation to achieve better convergence. Furthermore, incorporating query-to-label embeddings can improve the performance of the model, as seen in Version 2.

4.3.2 Changing the sequence length

During the test, we observed that the loss, $f1$ score, and accuracy curves for the training set were not behaving as expected. We initially expected them to converge to zero loss and 100% accuracy, but they stabilized at other values. Upon reviewing the code, we formed the hypothesis that the sequence length might be the cause of this behavior.

The sequence length is a parameter that determines the size of the window applied to the data. At each iteration, the model processes a piece of data that is randomly selected from the dataset. This means that even if the training set is the same, the specific pieces of data used in each iteration will vary, which prevents the model from converging to 100% accuracy. In other words, the randomness introduced by the sequence length ensures that the model is exposed to a diverse range of examples, which can improve its ability to generalize to new data.

We tested different sequence lengths and feature representation methods, and the results are presented in Table 4. For a short sequence length, we observed that the accuracy curves tended to oscillate more. However, for a larger sequence length, we noticed some slight overfitting with certain feature representation methods. For example, in Figure 18, we can see that the model begins to overfit when using MFCC feature representations with a

sequence length of 200. More detailed accuracy curves for each experiment are provided in Appendix B (5).

However, the training results were not significantly improved with changed sequence length. The best accuracy result on training set remained under 60%. Thus the sequence length may not be the decisive factor in the lower-than-expected performance of training sets.

	Sequence Length				
Representation	10	50	100	150	200
Melspectrogram	42,71 (33,18)	53,44 (41,25)	54,79 (44,28)	53,33 (41,42)	54,27 (43,40)
MFCC	49,69 (40,17)	52,98 (42,99)	54,79 (44,38)	58,75 (47,01)	54,61 (41,72)
Wav2vec	52,30 (43,20)	51,90 (42,50)	53,00 (43,50)	53,80 (44,90)	52,80 (44,00)

Table 4: Accuracy[%] and F1 Score[%] of Version 1, under different sequence lenghts.

4.3.3 Changing the number of fully connected layers

In the Version 1 of our model, we used a MLP with 2 fully connected layers. In an attempt to improve its accuracy, we increased the number of layers to 3. While this resulted in a slight improvement of less than 1% on average, we noticed that the processing time for each epoch also increased. As a result, we decided to focus our testing on the 2-layered model. Despite training the model with more epochs, we observed that it stabilized around the same values for loss and accuracy. The graphs related to these experiments can be found in Appendix C (5).

Number of FC layers	**F1 Score [%]**	**Average accuracy [%]**
2	44,34	56,12
3	44,54	56,32

Table 5: Accuracy[%] and F1 Score[%] of Version 1, under different number of FC layers.

4.3.4 Applying cross validation

In cross-validation, the dataset is split into multiple folds, and the model is trained and evaluated on each fold separately. This approach allows us to estimate the generalization performance of the model on new, unseen data. The results obtained for each fold (using one of the IEMOCAP dataset sessions as testing sets) are displayed in the Table 6.

We observe that the values obtained for accuracy and $f1$ score don't change a lot between folds, which is likely due to the fact that the speakers and emotions in the different sessions are similar enough that the model is able to generalize well across them. In other

words, the data is homogeneous enough that the model is able to learn generalizable patterns across speakers and sessions. This is a common assumption in speech emotion recognition, where it is often assumed that the emotional content of speech is largely independent of the speaker and the language used.

Version 0 - MLPClassifier

Test Session	$f1$ score [%]	Accuracy [%]
Session 1	15,03	27,43
Session 2	13,52	26,67
Session 3	14,96	25,56
Session 4	15,93	27,58
Session 5	12,85	27,14
Average	14,46	26,88

Table 6: Results using different sessions as test set.

4.3.5 Final results

After having tested different variables we found the best model for each feature extraction method. The best $f1$ scores and accuracy of each method are shown in the Table 7. We can see that the results show a significant improvement from Version 0 to Version 1, but the results of Version 2 are worse than Version 1. However, results of different feature representation methods are similar, which is not what we had expected.

	Num. of parameters	$f1$ [%]	Accuracy [%]
V0	597611	14,46	26,88
V1 - spec	346244	40,66	50,83
V1 - mels	124548	44,28	54,27
V1 - mfcc	91268	47,01	58,75
V1 - w2v	345732	46,20	56,60
V2 - mels	740100	42,36	53,13
V2 - w2v	7368708	45,41	55,10

Table 7: Comparison of the best results for each model.

The Version 0, despite being the initial model to help us learn to use the audio libraries and computer learning tools, demonstrated promising results by recognizing all 9 emotions (diregarding *xxx* and *oth*) in the dataset.

With Version 1, we achieved results mostly within our expected range. The best accuracy of the four feature representation methods are over 50%, but we did not expect that the MFCC was the best among them. Since MFCC representation does not include the voiced sounds, we believed it would not be the best method. It is, indeed, very widely-used for emotion classification, and researchers have found similar results to the ones we obtained in the

paper *Arabic Speech Emotion Recognition From Saudi Dialect Corpus*[3]. One explanation is that MFCC features mimic to a certain extent the reception pattern of sound frequency intrinsic to a human[9].

It was also expected that Version 2 of the network would surpass Version 1 in performance. This, however, did not happen. One possible explanation for this would be the larger amount of parameters that this model has to train. To suuccessfully train networks with more parameters, we need a larger dataset and more varied examples. We suspect our dataset was not big enough to support this amount of parameters. This could also be one of the factors that contributes for the good results with MFCC representation since it has less parameters than the other representations.

5 Conclusion

In conclusion, this project has explored the topic of deciphering nonverbal behavior, with a specific focus on speech emotion recognition. We have presented three different approaches to tackle this task, each using different combinations of audio feature extraction methods and machine learning models.

The first approach employed concatenated *chroma*, *melspec*, and *mfcc* as audio features, and an MLPClassifier as the model. The second approach used separated *melspec*, *spectrogram*, *mfcc*, and *wav2vec* as audio features, and a combination of LSTM and MLP layers as the model. The third and final approach utilized separated *melspec* and *wav2vec* as audio features, and a query2label (Q2L) model.

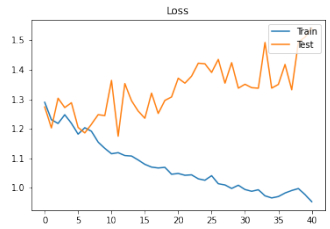
Overall, our experiments showed that the second approach outperformed the other two in terms of accuracy, with an accuracy of $58,75\%$ and a $f1$ score of $47,01\%$ for the *mfcc* on the IEMOCAP dataset. However, the third approach showed promising results with an accuracy of $55,10\%$ and a $f1$ score of $45,41\%$, despite using fewer features and a simpler model.

Cross-validation is a crucial step in machine learning model development that helps to evaluate the generalizability of the model's performance on new, unseen data. In this article, due to limitations on RAM and GPU usage, we were not able to perform cross-validation on all of our models. Eventhough the results obtained from each folder with the cross-validation done in Version 0 did not change a lot, it can vary depending on the specific dataset and the model being used. If the emotional content of speech varies significantly between speakers or if the dataset is particularly small or unrepresentative, then the results of cross-validation may be less consistent.

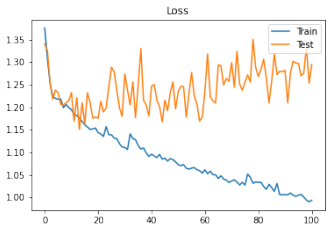
Therefore, if we had more available resources, we could have implemented it. Performing cross-validation would enable us to obtain a more reliable estimate of the model's performance, as it would be evaluated on different subsets of the data, thus reducing the risk of overfitting or underfitting.

Our findings suggest that speech emotion recognition can be achieved through various methods, and that the choice of audio features and machine learning models can significantly impact the accuracy of the predictions. Future research in this field could explore the use of other feature extraction techniques and models, as well as the application of these methods to real-life scenarios, such as speech-based emotion recognition in human-robot interactions or virtual assistants.

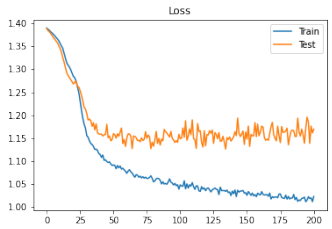
Appendix A: Loss function for different learning rates



(a) lr=1e-3



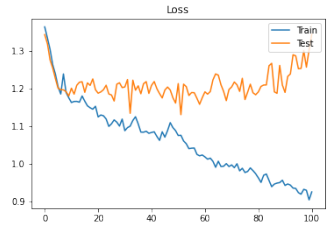
(b) lr=1e-4



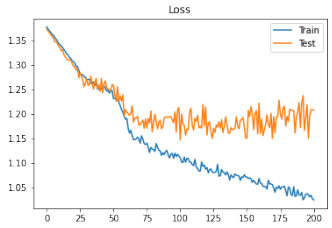
(c) lr=1e-5

Figure 10: Loss function of Mel-spectrogram in Version 1.

(a) lr=1e-3

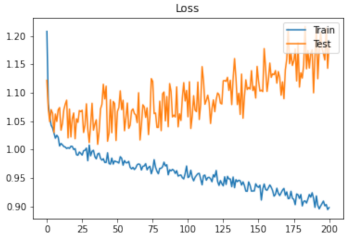


(b) lr=1e-4

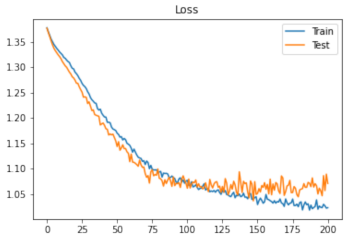


(c) lr=1e-5

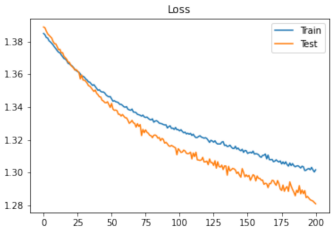
Figure 11: Loss function of Spectrogram in Version 1.



(a) lr=1e-3

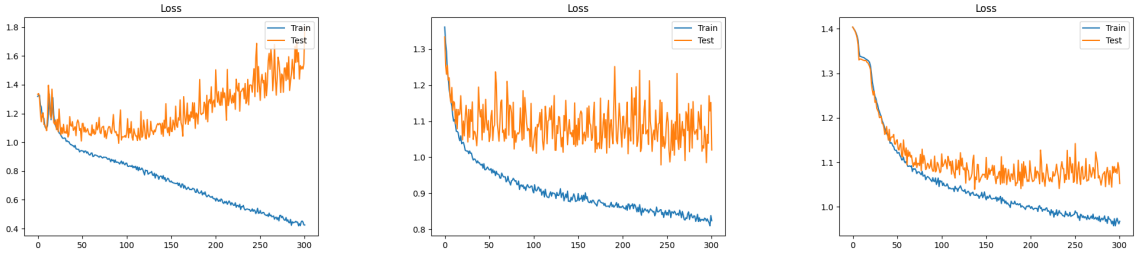


(b) lr=1e-5



(c) lr=1e-6

Figure 12: Loss function of MFCC in Version 1.

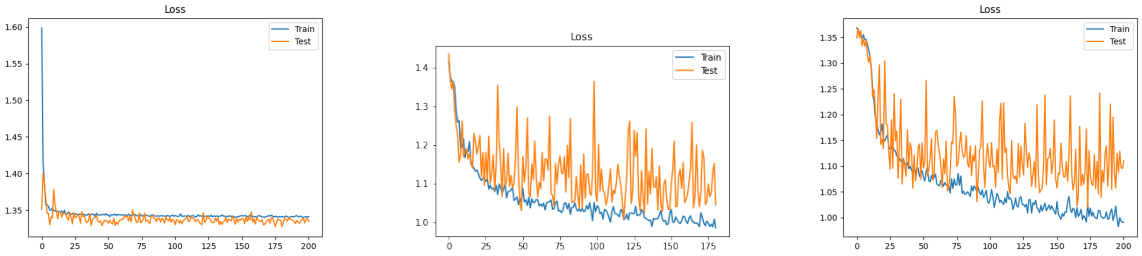


(a) lr=1e-3 (b) lr=1e-4 (c) lr=1e-5

Figure 13: Loss function of Wav2vec in Version 1.

(a) lr=1e-3 (b) lr=1e-4 (c) lr=1e-5

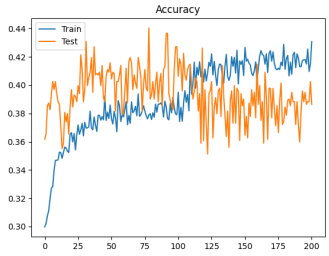
Figure 14: Loss function of Mel-spectrogram in Version 2.



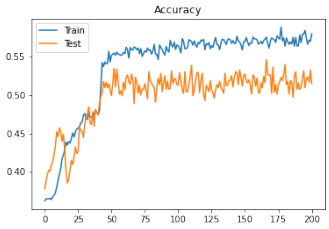
(a) lr=1e-3 (b) lr=1e-4 (c) lr=1e-5

Figure 15: Loss function of Wav2vec in Version 2.

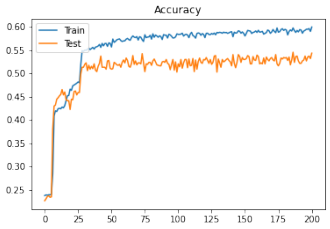
Appendix B: Accuracy for different sequence lengths



(a) sl=10

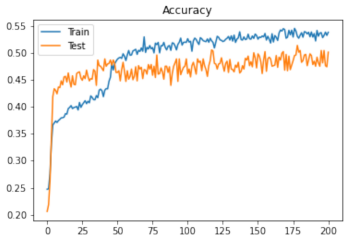


(b) sl=50

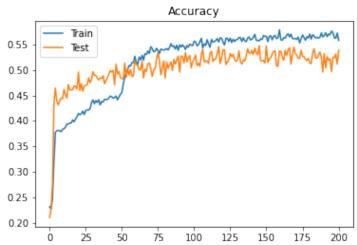


(c) sl=200

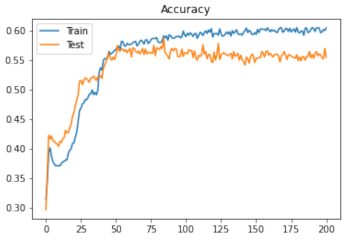
Figure 16: Accuracy of Mel-spectrogram in Version 1.



(a) sl=10

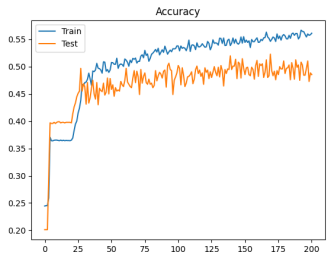


(b) sl=50

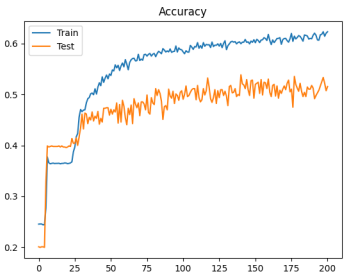


(c) sl=200

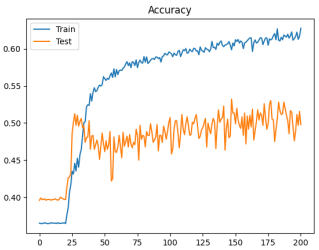
Figure 17: Accuracy of MFCC in Version 1.



(a) sl=10



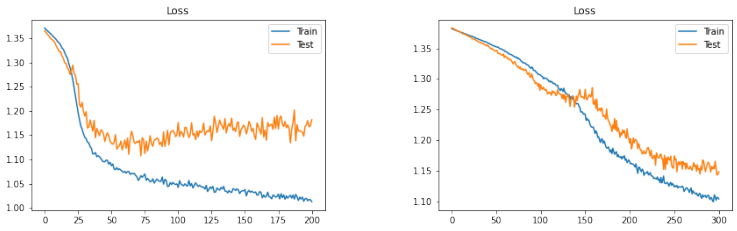
(b) sl=150



(c) sl=200

Figure 18: Accuracy of Wav2vec in Version 1.

Appendix C: Loss Function for different number of fully connected layers



(a) fc layers = 2 (b) fc layers = 3

Figure 19: Loss function of Melspetrogram with numbers of fully connected layers in Version 1.

References

[1] M. Alagözlü. Stochastic gradient descent variants and applications. 02 2022. doi: 10.13140/RG.2.2.12528.53767.

[2] M. Alhaisoni. Speech emotion recognition using multilayer perceptron neural network. *International Journal of Advanced Computer Science and Applications*, 10(5):124–129, 2019.

[3] R. H. Aljuhani, A. Alshutayri, and S. Alahdal. Arabic speech emotion recognition from saudi dialect corpus. *IEEE Access*, 9:127081–127085, 2021. doi: 10.1109/ACCESS.2021.3110992.

[4] S. T. Z. S. Ameya Ajit Mande, Sukrut Dani. Emotion detection using audio data sample. *International Journal of Advanced Research in Computer Science*, 10(6), 2019.

[5] D. S. Anton Blatliner, Björn Schuller and S. Steidl. The automatic recognition of emotions in speech. pages 71–99, 2011.

[6] R. J. Eva Lieskovska, Maros Jakubec and M. Chmulik. A review on speech emotion recognition using deep learning and attention mechanism. *Human Computer Intercation for Intelligent Systems*, 2021.

[7] E. S. Georgios Drakopoulos, George Pikramenos and S. J. Perantonis. Emotion recognition from speech: A survey. 2019.

[8] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[9] D. Issa, M. Fatih Demirci, and A. Yazici. Speech emotion recognition with deep convolutional neural networks. *Biomedical Signal Processing and Control*, 59:101894, May 2020. ISSN 17468094. doi: 10.1016/j.bspc.2020.101894. URL https://linkinghub.elsevier.com/retrieve/pii/S1746809420300501.

[10] J. Joy, A. Kannan, S. Ram, and S. Rama. Speech emotion recognition using neural network and mlp classifier. *Department of Computer Science and Engineering SRM Institute of Science and Technology, Vadapalani Campus, Chennai, India*, 2021.

[11] S. Liu, L. Zhang, X. Yang, H. Su, and J. Zhu. Query2label: A simple transformer way to multi-label classification. *arXiv preprint arXiv:2105.06355*, 2021.

[12] B. Nikhil and L. Nicholas. *Fundamentals of Deep Learning : Designing Next-Generation Machine Intelligence Algorithms*. O'Reilly Media, 2017.

[13] Y. O. Nobuo Sato. Emotion recognition using mel-frequency cepstral coefficients. 2007.

[14] K. O'Shea and R. Nash. An introduction to convolutional neural networks. 2015.

[15] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

[16] M. Wöllmer, F. Eyben, S. Reiter, B. Schuller, C. Cox, E. Douglas-Cowie, and R. Cowie. Abandoning emotion classes - towards continuous emotion recognition with modelling of long-range dependencies. In *Interspeech 2008*, pages 597–600. ISCA, Sept. 2008. doi: 10.21437/Interspeech.2008-192. URL https://www.isca-speech.org/archive/interspeech_2008/wollmer08_interspeech.html.