LoopQ PRIZE 2022 Competition
Case A : Emotion Detection from speech
Solution Overview

Competitor: Biagini Diego

1 Introduction to the problem

Trying to understand how a person's observable actions and attitude correlate to their own inner emotional state is a problem that has been studied extensively in psychology, and of course we would like to automatically do this with the current technological tools at our disposal.
Such a technology has many applications in real world scenarios, it can be very informative from a human computer interaction point of view as well as an analytics and marketing one.

Examples of cues that can be considered in predicting one's emotional state are various, they can be written pieces of text (the famous text sentiment analysis task)[1], video and/or audio clips[2], gesture or facial expression based[3] and even neurological ones[4].
The objective of the challenge is an instance of those, it is **Speech Emotion Recognition(SER)**, this problem boils down to the recognition of a subject's emotional state given a relatively small audio clip of them speaking, either by themselves or in a conversation with someone else.

From a human point of view such a problem doesn't pose many challenges but as soon as we try to formalize it we run into fundamental problems. What is an emotion? How can emotions be represented?
As an answer to these questions formal systems have been devised to characterize one's emotional state, one of those is the representation of emotion over a continuous multidimensional axis, e.g. the **Valence-Arousal-Dominance(VAD)** representation[5].
What we are interested however is a conceptually simpler approach, that is one's emotional state corresponds to a single one from a set of possible categories.
This set can be of arbitrary cardinality, but in the task at hand it corresponds to the following emotional states: **anger, disgust, fear, happiness, sadness, surprise and neutrality**.
After we have defined the possible emotions we have to classify our data according to them, how can we achieve this? The only possible solution is manual human labeling and that is what has been done on all datasets created to study this task.

With a labeled dataset nothing stops us from taking the task into the well studied discipline of Machine Learning.
Attempts to solve this problem have been undertaken since the dawn of the field, with the earlier ones exploiting hand crafted features[6] and well engineered systems to squeeze every possible drop of data expressiveness available using classical models like Naive Bayes[7], Support Vector machines[8], Restricted Boltzmann Machines[9], Hidden Markov Models[10], Gaussian Mixture Models[11], etc.

However advances in **Deep Learning** have completely blown Machine Learning out of the water, especially in perceptive tasks like this one, and after the first years of the 2010s many deep learning architectures/innovations have been experimented on this task[12][13], like strictly **Convolutional NNs**[14] [15], **Recurrent NNs**[16][17], **Autoencoder** based[18][19] and **Generative**[20] approaches and more recently **Attention**[21][22] and **Transformer** based models[23][24].

With that said the approach taken in completing this challenge has been to explore these latest developments and see how the various deep learning models available in the literature fare on the dataset at hand.

2 Data and dataset analysis

2.1 Dataset composition

The data that was provided is divided into a labeled dataset and an unlabeled dataset used as challenge result assessment, all the data analysis has been performed on the former, which contains 10110 valid records.
After a quick integrity check for missing values, the first step in data analysis has been to analyze the higher level information, that is how the emotion labels and origin of the files (the original datasets they are from) are distributed.
The results are reported in the following table:

		Origin				
		CREMA	RAVDESS	SAVEE	TESS	
Emotion	angry	1058	163	98	333	1652
	disgust	1085	154	52	338	1629
	fear	1082	163	51	332	1628
	happy	1084	155	48	347	1634
	neutral	939	81	97	335	1452
	sadness	1058	162	0	335	1555
	surprise	0	163	51	346	560
		6306	1041	397	2366	10110

Table 1: Distribution of records according to emotion (vertical) and origin (horizontal)

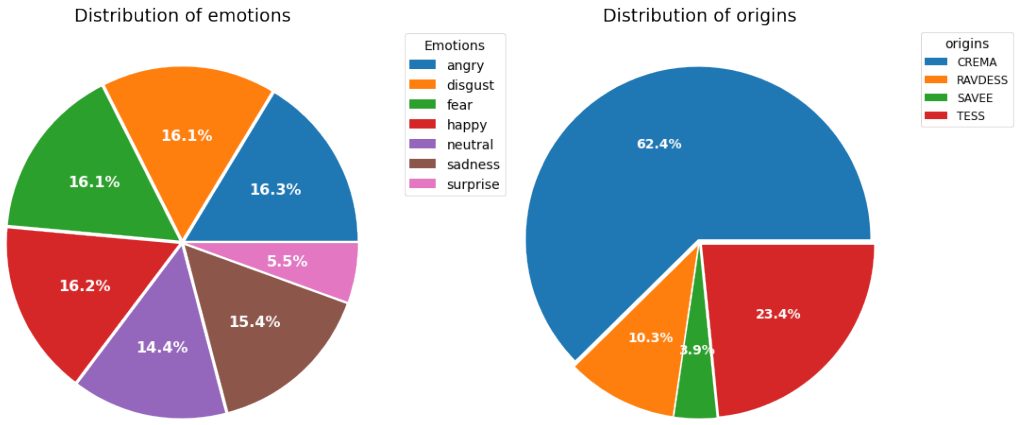

Figure 1: Individual emotion (left) and origin (right) percentages in dataset

We can observe that the distribution of emotions is quite balanced, except for "surprise".
The same can be said when we consider emotion categories within classes, except for origins which have no record of a certain class.
We can't say the same for the overall origin of the samples, in fact more than half of them came from CREMA.

2.2 Processing Audio Signals

Our SER task has at its core the processing of digital audio signals, so we shall define different ways of dealing with them.
The first parameter to take notice of is the **sample rate** of a particular digital signal, the sample rate corresponds to how many times each second the original analog signal has been sampled during the digitization process.
A raw digital audio signal is thus represented as a series of $sample\_rate \times duration$ measurements, these measures are the amplitudes and they are real valued, this is also called the waveform of the audio signal and can be processed as is.

However the signal processing field is able to give us more sophisticated representations, one of these is the **Spectrogram**.
The spectrogram of an audio signal is a 2 dimensional representation which is able to also give frequency information about the signal, in particular which frequencies are more present in a rectangular window of the waveform, this is done by taking its short time Fourier transform (STFT).
The parameters needed to obtain the spectrogram from a waveform are the window size and the hop length (distance between subsequent windows) both in terms of samples.

The spectrogram is a good representation for unspecified audio signals, however it's not as easily interpretable from the human point of view, due to the fact that a normal spectrogram is a linear representation in the frequency domain and humans perceive frequencies logarithmically, a better representation is the so called **Mel Spectrogram**.
The difference between the two is just a rescaling from Hertz to a new unit called Mel to increase the similarity of perceptually similar (at least for humans) frequencies in the spectrogram representation. To obtain a mel spectrogram from an audiowave we also have to set the number of mel filter banks in addition to the parameters needed for the creation of the spectrogram itself.

Another possible representation, which is actually obtained from the mel spectrogram, is the extraction of the so called **Mel Frequency Cepstrum Coefficients (MFCC)**.
Its main purpose is to highlight the more relevant information from a speech utterance, that is the one generated by the vocal tract instead of the glottal pulse, which empirically carries less information about the content (both linguistic and para linguistic) of the speech signal[25].
The parameter which defines the MFCC representation from a mel spectrogram is the number of coefficients we want to extract. In some cases these are decorated with first and second order time derivatives, the so called Δs and $\Delta\Delta$s.
This is actually one of the most used representations in many speech and music related tasks.

2.3 Audio Data properties

An analysis on the properties of the audio files themselves was also conducted using the **librosa** python library.
First of all the sampling rates were checked and it turns out that the dataset contains files recorded at different sampling rates, these are: 16000, 24414, 44100, 48000.

After intensity normalization we can extract information about the amplitudes of the audio signals, for example the average of the intensities of an audio over all the dataset is 0, while the average over the dataset of the maximum and minimum intensities of an audio are respectively 0.2796 and -0.2751.
Grouping with respect to the emotion and origin might also be interesting, results are in the table below:

	Grouping										
	angry	disgust	fear	happy	neutral	sadness	surprise	CREMA	RAVDESS	SAVEE	TESS
Max	0.575	0.211	0.286	0.337	0.156	0.105	0.219	0.301	0.192	0.569	0.210
Min	-0.559	-0.202	-0.284	-0.337	-0.151	-0.106	-0.224	-0.286	-0.177	-0.603	-0.231

Table 2: Average dataset statistics of amplitude when grouping according to a certain category

We can kind of deduce some properties of waveforms of particular emotions based on their amplitudes, for example it looks like in general anger and happiness are loud while sadness is much quieter (as can be expected). Furthermore we can see that the various datasets have relatively different waveform statistics, this could be due to recording conditions but also due to speaker peculiarities.

With regards to the length of the audios: the average length was 2.597 seconds with a standard deviation of 0.706 seconds, the shortest being 1.256 and the longest 7.139 seconds long. Furthermore the 95th percentile of audio length is 4.804 s.

We can visualize the waveforms and mel spectrograms of samples taken from different emotion categories:

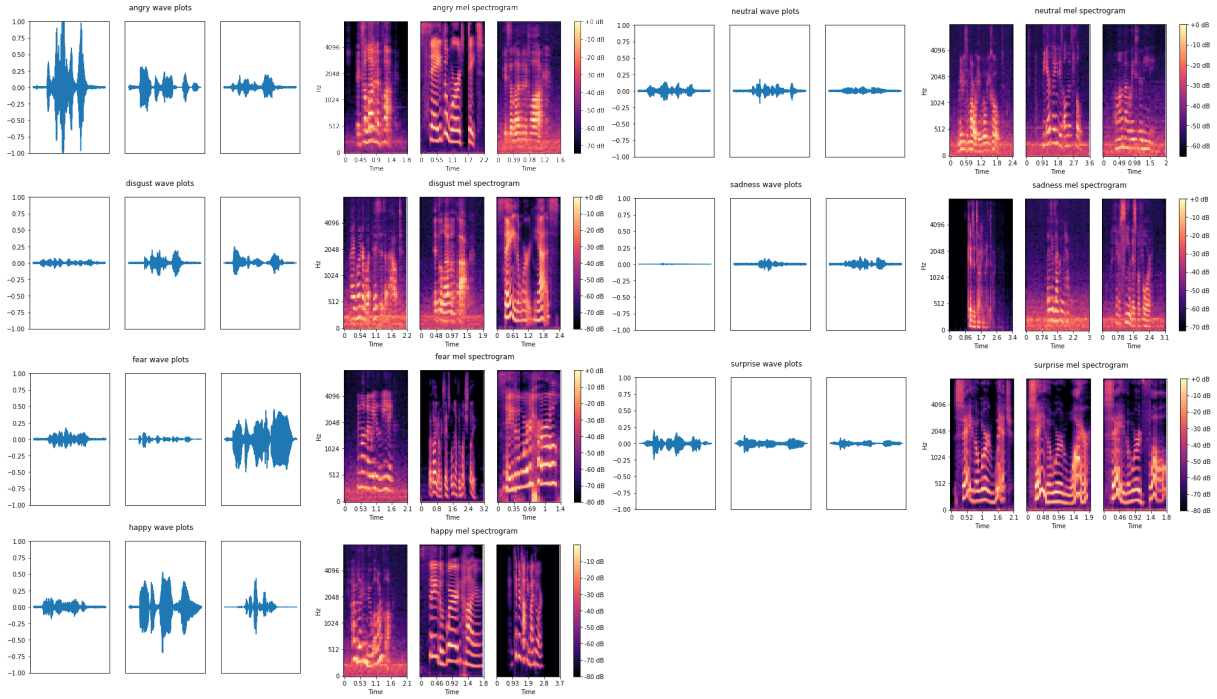

Figure 2: Samples of emotion waveforms (left) with their mel spectrogram representation (right) with a 2048 frames window, 1024 hop length and 128 mel filter banks

2.4 Decompositions to analyze the speech signal

There have been attempts to perform decomposition of the audio signal to differentiate between different speech properties like rhythm, content, pitch, timbre using unsupervised methods [26] and also using the information provided by a ML model trained for emotion classification [27], this would for sure give us more information (maybe even human interpretable) to notice differences between the expressed emotions.
However all of these methods exploited the fact that the individual speaker responsible for each utterance is known so we are able to disentangle both the timbre and other noise conditions (which should be the same for the same speaker), something that we don't have in our dataset.

Another general purpose method is performing NMF (non negative matrix factorization) over the magnitude spectrogram to obtain a matrix of components (expressing pitch and timbre) and activations (expressing time related information). This method was tested but no information was gleaned from it, we could argue due to the fact that emotion is too much of a high level information to be preserved in this representation.

So all we are left with to do an informal analysis are general signal analysis methods, like the mel spectrogram and actually we can notice some visual differences on it based on the emotion.
Looking at a lot of samples, some of them are actually very clear, for example in sadness spectrograms most of the energy is restricted in the lower frequency bands and surprise spectrograms contain a lot of high-energy bands evenly spaced along the frequency domain. Meanwhile disgust, fear and in particular anger and happiness have presence across all frequencies.
Other properties could be more questionable, for example it looks like that when an angry utterance starts the energy among the frequencies remains sustained for the entire duration of the utterance.

Once we have trained a machine learning model which uses the mel spectrogram as information we can actually verify if these intuitions were correct using an attribution method, that is a way to quantify how much a part of the input contributes to a certain prediction.
For sure a well trained model is able to learn a representation which allows it to predict its targets reliably, however such a representation is not human accessible so what we can do is project it towards what we can read, the inputs.
The simplest attribution method we can use, both in terms of resource usage and cognitive complexity, is **Saliency**[28] which was designed to work on images and that's what a spectrogram is

after all. Sampling the result of running it on the mel spectrogram part of the chosen final model (explained later) yields the following results:

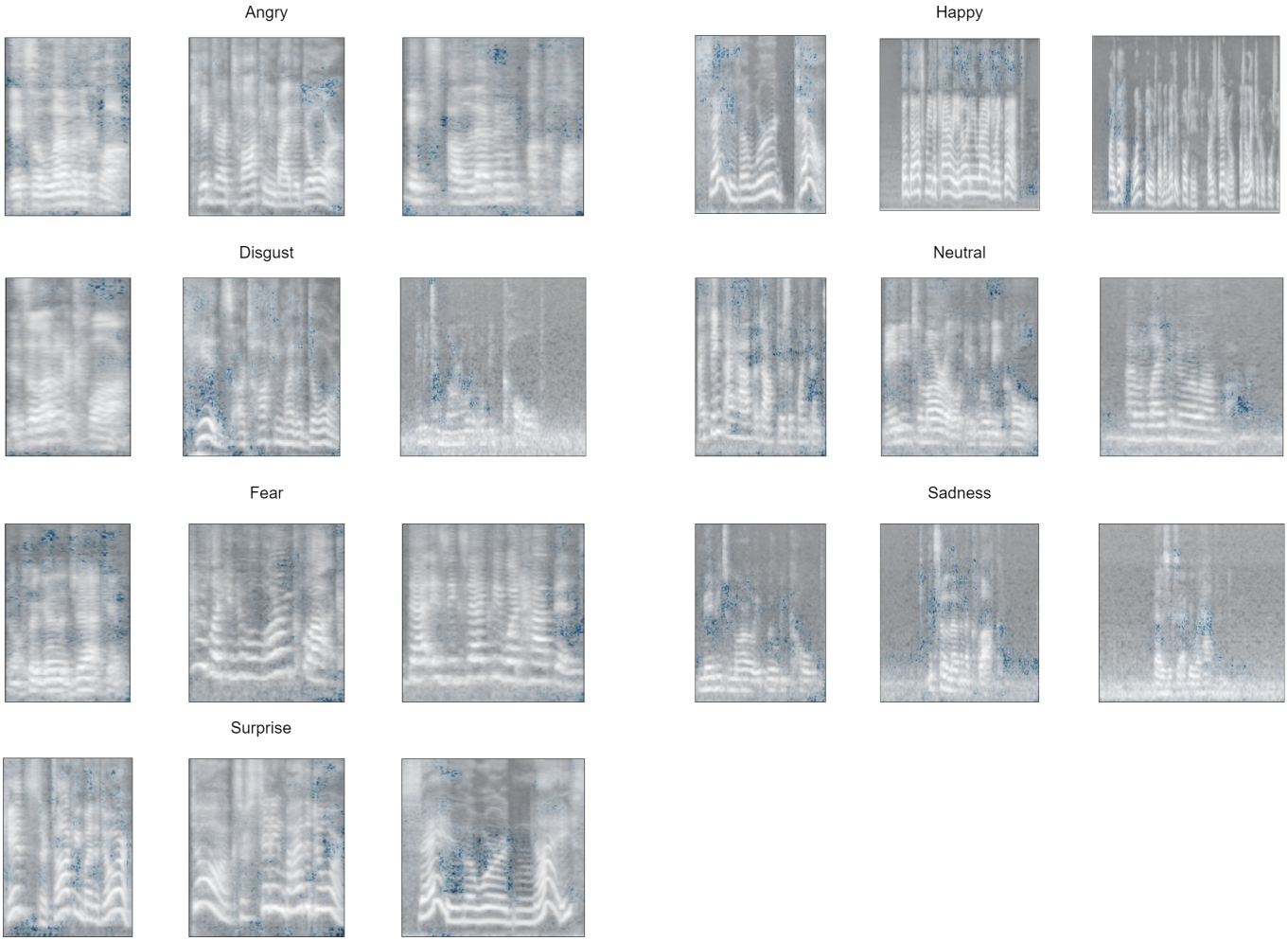

Figure 3: Attribution heatmaps using the saliency method on the mel spectrogram over some samples of different emotions, generated using the Captum library

Our suppositions on the sadness and surprise spectrogram properties seem to be indeed correct, for the sadness classification the low-middle frequencies are the only important ones and for the surprise classification a lot of importance is given to pixels near the quickly changing (over frequency) bands. Furthermore for happiness higher frequencies are much more relevant than lower ones.
The same method can be applied on the raw waveform and MFCC but the results weren't very illuminating, in the former case the only thing that was verified is that comparatively louder parts of an audio signal are more important for all emotions, while in the latter the resolution of MFCC is too low to see anything of value.

Anyway all of this is hardly generalizable and explainable through rules (like all perception tasks), that's were we have to rely on machine learning models instead of our confirmation-bias ridden intuition on hardly informative representations for our human senses.

3 Descriptions of the examined models

3.1 M1: Convolutional + LSTM network on Mel Spectrogram (CNNLSTM)

This model is based on the one described in [16].
The data which is provided to the network is the mel spectrogram, that is a 2 dimensional signal. The mel spectrogram is obtained by first fixing the raw waveform to be exactly 5 seconds long (either by cutting it or padding it) and then generating the mel spectrogram with a window size of 1024 frames, hop length 256 frames and 128 mel filter banks. With these settings the individual samples

sent to our model will have shape 128×313.

The network is based on two subsequent parts, a feature extraction one and global feature learning one.
The first is implemented through a sequence of so called **Local Feature Learning Blocks(LFLBs)**, which are just a sequence of a convolutional layer, a normalization one (batch normalization), activation function (ReLU) and finally a pooling layer (Max pooling).
Five LFLBs are deployed, their hyperparameters are:

- Convolutional layers: same kernel sizes of (3x3), unitary strides, "same" padding, different output channels respectively of (32,48,64,128,256)
- Max Polling layer: different kernel sizes and strides respectively of sizes (2x2,2x2,2x2,4x4,4x4)

The output is the flattened before being sent to the feature learning part .

The feature learning part makes use of an **LSTM (Long Short Term Memory)** layer,the most well known type of RNN which uses gated activations to decide when to preserve/forget past information about the input sequence, to perform sequence to sequence synthesis on the feature extracted by the convolutional part. The LSTM has 2 layers with a dropout layer between them and the dimension of the output sequence obtained from it is 256 channels.

The output of the LSTM is then passed through a dropout layer and fed into a fully connected layer and a softmax function for final classification.

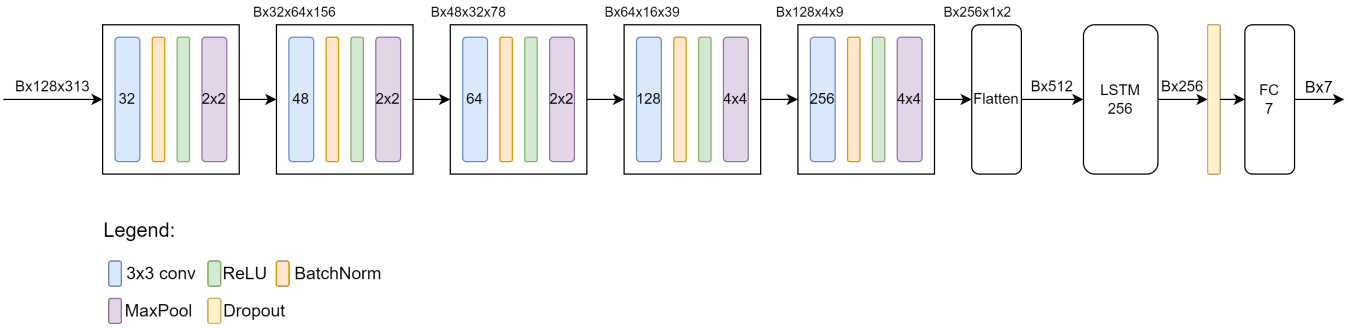

Figure 4: Graphical representation of the CNNLSTM model

3.2 M2: Fully convolutional network on MFCC (FCMFCC)

This model is based on the one described in [15].
The data provided to the network is the MFCC representation computed from the audio waveform, which is a 2 dimensional signal. The audio signal is first forced to be exactly 5 seconds long by either cutting or padding it.
The MFCC representation is then obtained using a 1024 frame window length, 256 frames hop length, 128 mel filter banks and 40 coefficients. With these settings the individual samples sent to our model will have shape 40×313.

The network is composed of an initial processing part implemented using parallel paths, followed by a feature learning part and finally a classification head.

The first part of the network uses a well known paradigm of convolutional networks, which was first coined **Inception Module** in [29], it consists in applying multiple kernels of different sizes to the same input, processing the individual results and finally stitching these parallel paths together by concatenating them.
A single path is composed of a convolutional layer, a batch normalization layer, ReLU activation and an average pooling layer.
The convolutional layers all have 32 output channels, they use "same" padding, their kernel sizes are respectively 3x3,9x1,1x11, this way we are able to synthesize feature maps along different spatial dimensions. The average pooling layers have a 2x2 kernel size.

The concatenated result from the previous part is fed into a series of 5 LFLBs, each one of them composed of a convolutional layer, batch normalization, ReLU and pooling layer.
The convolutional layers all have 3x3 kernel sizes and "same" padding with output sizes of respectively 64,96,128,160,320 channels.
The first two pooling layers are average pooling layers with 2x2 kernel sizes, the next two are the same with 2x1 kernel sizes and the last one is a global average pooling layer to prepare the data for the classifier head.

This is then passed through a dropout layer which is fed into a fully connected layer and a softmax function for final classification.

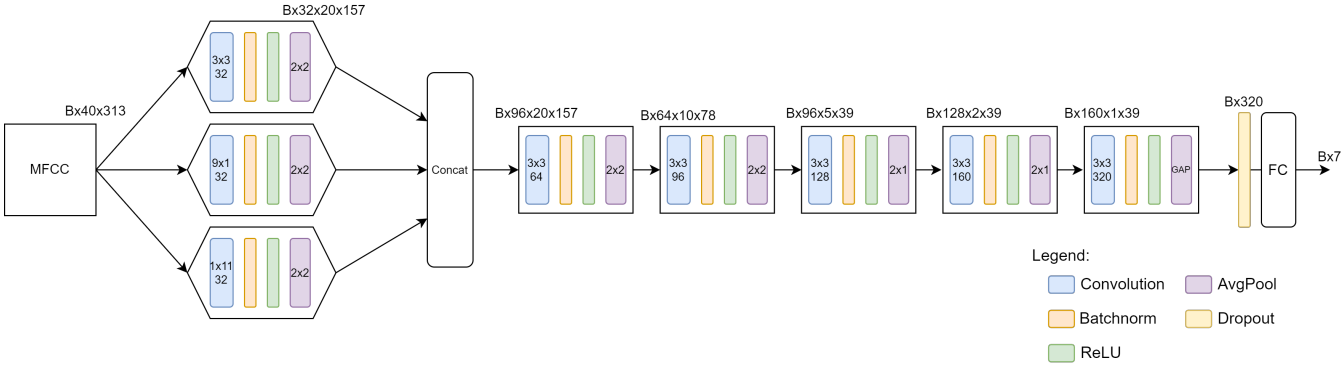

Figure 5: Graphical representation of the FCMFCC model

3.3 M3: Convolutional network with Attention on MFCC (CNNATT)

This model is based on the one described in [21].
The data provided to the network is the MFCC representation computed from the audio waveform. This model's architecture allows the processing of data of arbitrary length, so no length constraint are enforced by default. However a batch of samples has to have the same shape to be processed, so when collating a set of arbitrary length samples into a batch they are all padded to be of the same length as the longest one.

The MFCC representation is then obtained using a 1024 frame window length, 256 frames hop length, 128 mel filter banks and 40 coefficients. With these settings the individual samples sent to our model will have shape $40 \times (floor(\frac{batch\_max\_length}{256}) + 1)$, where $batch\_max\_length$ is the number of frames of the longest sample in the batch.

This model has an initial feature extraction part which resembles FCMFCC, followed by an attention mechanism and a classification head.

The feature extraction part is composed of a initial split path module followed by a sequence of 4 LFLBs.
The split path is composed of 2 paths whose convolutional layers have kernels of sizes respectively 11x3 and 3x9 and 8 output channels each. After the individual convolutions a batch normalization and ReLU activation function are applied and the two paths are concatenated.
The result is sent to a series of LFLBs each made up of a convolutional layer with a 3x3 kernel, batch normalization, ReLU and for the first two of these a 2x2 average pooling layer. The output channels of the convolutional layers are respectively 32,48,64,80.
The output of the LFLBs is flattened from the third dimension onward.

The next part is a modified version of the so called **Dot Self Attention** module. **Attention** is a general mechanism which allows the network to focus only on parts of a given sequence which are more relevant. There are countless possible implementations of an attention mechanism, the one in this model is similar to one of those presented in the famous paper "Attention is all you need"[30].

What is done in practice is: given an input tensor X of shape $d \times s$,where s is the length of the sequence and d is the input channel dimension, we want to obtain a tensor of shape $o \times s$, that is with the same sequence length and a fixed number of output channels o.

To do this three vectors called respectively the query Q, the key K and the value V (all of shape $o \times s$) are synthesized starting from X by matrix multiplication with a learnable weight matrix of shape $o \times d$:

$$Q = W_q \cdot X \quad K = W_k \cdot X \quad V = W_v \cdot X$$

The following operation is then performed: $X_{out} = softmax(\frac{K \cdot Q^T}{\sqrt{d}})V$
Any number of X_{out}s can be obtained in the same way using different weight matrices. These X^i_{out} are then averaged, this can be referred as multi head attention.
In the implementation the number of heads has been chosen to be 3 and the output size 128.

The result is then passed through a 1d global average pooling layer,a layer normalization layer (an alternative to batch normalization) before being sent to the fully connected head and the softmax function for classification.

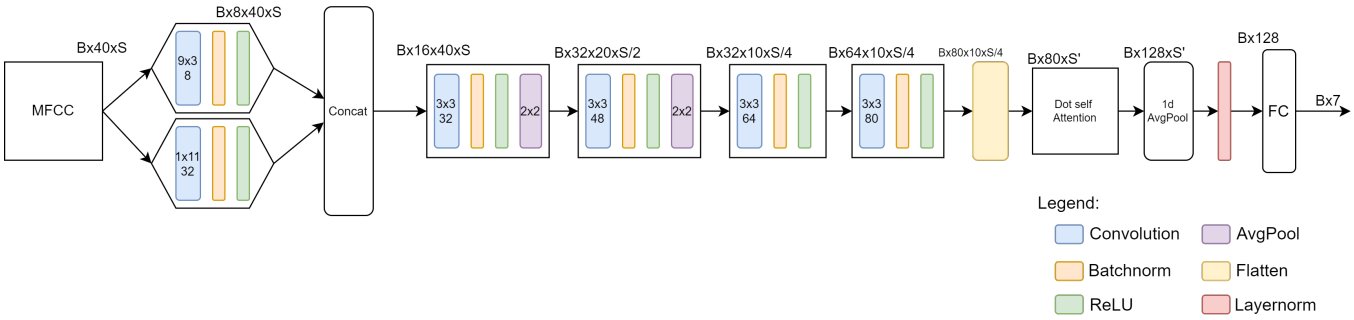

Figure 6: Graphical representation of the CNNATT model

3.4 M4: Finetuned HUBERT (HUBERTFT)

This model uses the methods described in [24].
The data provided to the network is the raw waveform which is forced to be at most 4.5 seconds long due to the high computational cost required during training with long sequences. No minimum length constraint is enforced since this network is able to process arbitrary length sequences, which when put together into a batch will be padded to be of the same length as the longest one in the batch.
As a result of this operation we can compute an attention mask, that is a binary mask which tells the model not to consider the padded part of the tensors

The model is composed of an initial large pretrained model whose outputs are fed into a classifier head.

The initial part uses the powerful paradigm of **Transfer Learning**, often also linked to the task of few/zero shot learning, it consists in using pretrained models originally trained on a big amount of data to specialize on another task for which we only have a relatively small dataset.
The pretrained model can either be kept completely frozen during training, that is none of its weights are subject to change, or only partially frozen.
The pretrained model incorporated in this model was HuBERT[31], in particular the HuBERT large version, a 300 million parameters model trained on 60 thousand hours of human speech.
This model can be divided into 3 parts: an initial convolutional part for feature extraction, a feature projection part and a transformer based encoder part. When training our entire model these first two parts were kept frozen while the encoder part was trained with a lower learning rate than the classifier head, this is the so called **Fine-Tuning** approach.
In addition to the batch tensor the tensor corresponding to the attention map was fed to HuBERT. The output of HuBERT large is a 2 dimensional tensor of size $1024 \times S'$, where S' depends on the original input sequence length.

This is fed into a 1d global average pooling layer along the sequence dimension, which is then sent to the head and through a softmax function for classification.

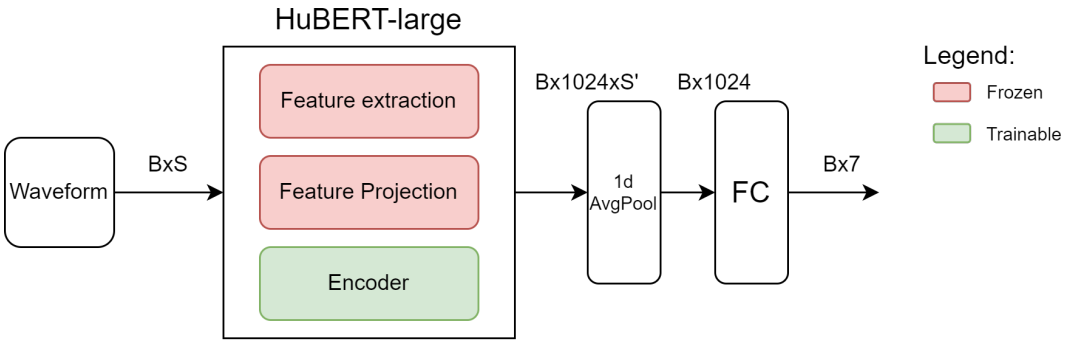

Figure 7: Graphical representation of the HUBERTFT model

3.5 M5: Multi modal network with fusion through Attention (MMFUSION)

This model is based on the modalities described in [32] and the fusion is implemented like in [33]. The data provided to the network is **Multi modal**, that is multiple inputs are fed to it and these inputs represent different points of view through which we can see the problem.
In our case there are three modalities, all obtained from the same source, they are:

- the original waveform
- the mel spectrogram, obtained using a 1024 frame window with a hop length of 256 frames and 128 mel filter banks
- the MFCC, obtained from the mel spectrogram above with 40 coefficients

Before computing the modalities the waveform is forced to be exactly 5 seconds long either by cutting it or padding it.

The network is composed of a modality processing part for each modality, an attention based fusion part and finally a classifier head.

The mel spectrogram is processed by first using a pretrained convolutional network without the classification head and finetuning it. The one that was chosen is ConvNeXt[34], a recently developed fully convolutional network for image processing which tries to replicate the mechanism of transformer based ones. The final pooling layer is removed from the network and replaced with a 2d flattening, in such a way we are trying to obtain a 2-dimensional 1024×36 feature map, where the second dimensionality is intended as the sequence length of features extracted by the CNN, we call this output x_s.

The MFCC is processed using a bidirectional LSTM with 2 layers and 256 output channels. With our settings we obtain a 512×313 tensor, we call this output x_m.

The raw waveform is processed using a finetuned HuBERT base model, a 90 million parameter self supervised audio model. The output has shape $768 \times S'$, where S' is the output sequence length obtained by HuBERT which with our settings was 249. We call this output x_w.

At this point the three modalities are fused using a particular implementation of an attention method, called unified attention, implemented with a so called **Unified attention block(UA)**.

The main component of the UA block is the **Gated Self Attention(GSA)** module.
Given an input sequence X of shape $d_{in} \times m$ where m is the sequence length and d_{in} is the input channel length; the Q,K,V of shape $d_{out} \times m$ transformations are computed, where d_{out} is the output channel length of the sequence.
The output of the GSA F is then computed as follows:

$$M = \sigma(f_g(f_g^q(Q) \odot f_g^k(K)))$$

$$A = softmax(\frac{(Q \odot \tilde{M}_q)(K \odot \tilde{M}_k)^T}{\sqrt{d_{out}}})$$

$$F = AV$$

Where f_g^q, f_g^k are linear projections of the sequence into a d_g channels space; f_g , whose output has shape $2 \times m$, tries to construct the 2 attention masks which after being passed through a softmax layer can be split into M_q, M_k which will be repeated d_{out} times to obtain $\tilde{M}_q, \tilde{M}_k$ of shape $d_{out} \times m$.

This mechanism can also be augmented with the multi attention approach, that is the d_{out} dimensional output space is obtained by concatenating k times $\frac{d_{out}}{k}$ dimensional GSA applications with different parameter weights, k is called the number of attention heads.
Furthermore after the concatenation another linear transformation which outputs once again a tensor of shape $d_{out} \times m$ is applied.

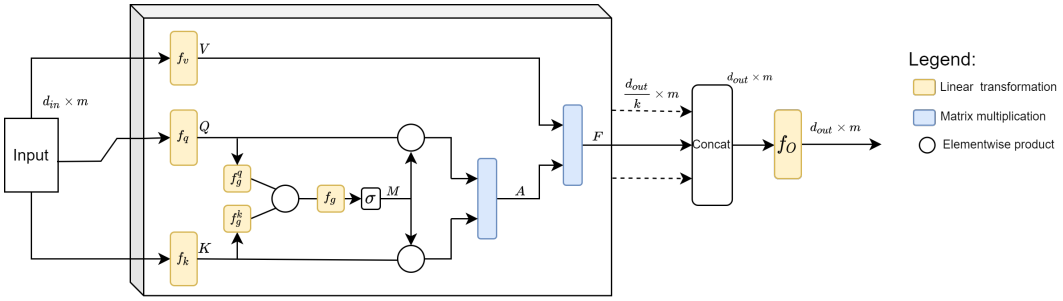

Figure 8: Multi headed gated self attention

The UA block is composed of two parts, an initial one which uses the GSA and a later part which uses a feed forward network(FF).
Given x_s, x_m, x_w, which are sequences of varying length and channel dimensions, they are first embedded in a d_z dimensional space as x'_s, x'_m, x'_w and concatenated along the sequence dimension into a unified feature matrix $Z = (x'_s \oplus x'_m \oplus x'_w)$
Z is first passed through the GSA module to obtain Z_{GSA}, before being sent to FF a skip connection is placed which computes $Z' = Z + Z_{GSA}$ and this is followed by layer normalization over the last 2 dimensions.
The feed forward network is composed of a dense layer with a $4d_z$ output channels dimension, followed by a ReLU activation, a dropout layer and finally another dense layer which resqueezes the dimension back to d_z channels. Like after the GSA part a skip connection and layer normalization is placed after the FF part as well.

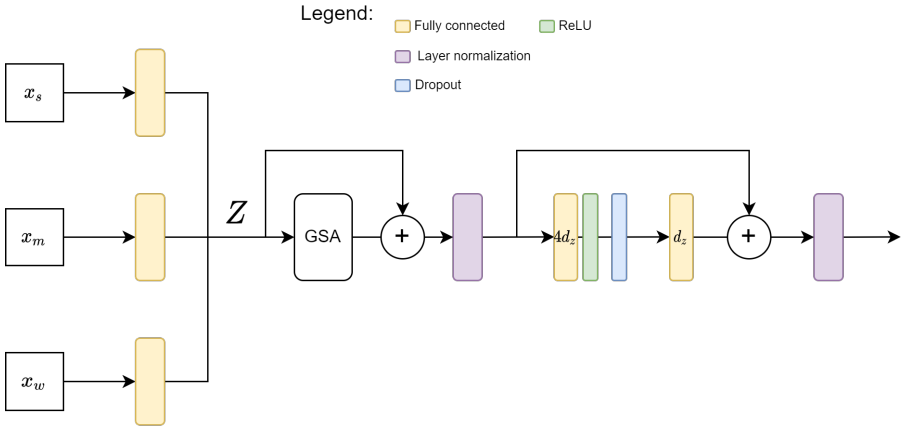

Figure 9: Unified attention block

In our architecture two UA blocks are placed after the modalities, with the second one not computing the initial linear transformation. Their hyperparameter is $d_z = 144$, and the GSAs inside them have 3 heads each one of them with $d_{out} = 48$ and $d_g = 48$.
To the output of the two UA blocks, which with our setting has shape 144×598, 1d average pooling is applied before being sent to the classifier head.
The classifier head is composed of 2 fully connected layers with output channel sizes respectively of 128,7 channels. The first layer is preceded by a dropout layer and followed by a ReLU activation.

The final result is passed through a softmax function for classification.

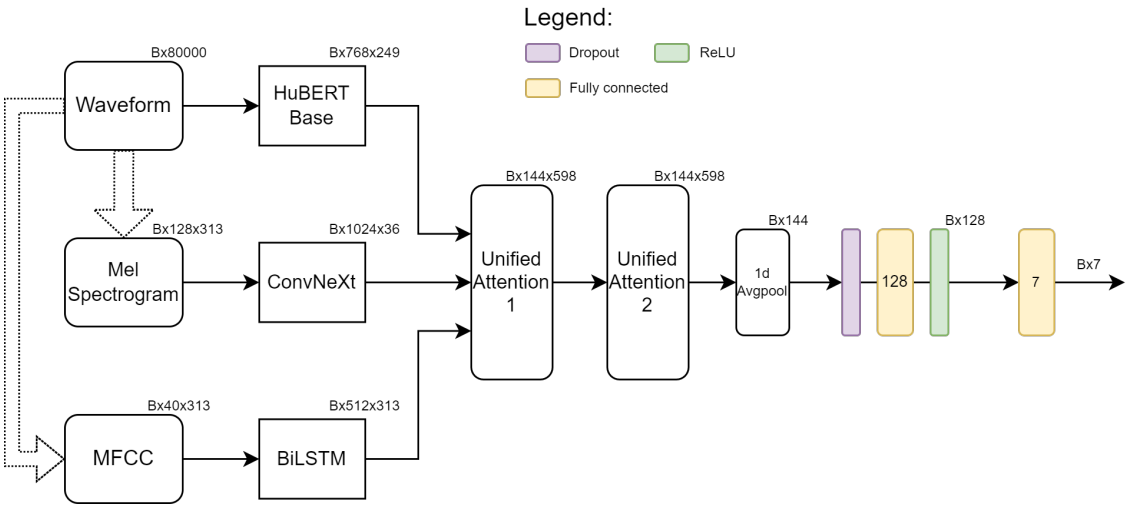

Figure 10: Graphical representation of the MMFUSION model

3.6 Data preprocessing and management

As described in section 2, the data comes from different sources, in particular the audio files had different sampling rates, so a standardization was needed for training our models.
The choice that was made was to downsample every file to a sample rate of 16000, this operation was done offline to speed up the training process.

Before processing the samples an additional operation which aimed at removing trailing silence was carried out.
From this cleaned data all the possible input data transformations like mel spectrogram and MFCC extraction where executed, with the parameters described for the individual models.

When working with a limited dataset an important task is data augmentation, that is synthesizing new data from the existing one.
This task is not so easy in a setting like this one since operators which transform the waveform (like amplitude or frequency stretching) might distort the data too much. A type of data augmentation on the waveform which was tested was time stretching, that is speeding up or slowing down slightly the audio, however no performance gains were found while the overhead needed in computing it is pretty high.
Other possible augmentations can be carried out on the Mel spectrogram, like frequency and time masking [35]. They did not show much promise either but at least they were computationally cheap.

A crucial aspect of data management is how to handle the provided data to train the model as best as possible.
To obtain such a result we decided to use the simple holdout method, that is the data was split into a training and validation set.
The split percentage was 80% of the 10110 records for training, which were 8085, and the remaining for validation.
The split was forced to be stratified with respect to the labels, that is the individual splits contain the same ratio of labels as the entire dataset.
A better approach than holdout would be cross-validation however this would incur in a much higher computational cost, especially with the more complex models.

3.7 Experiments and results

All the models were implemented using the python library **PyTorch**, the **librosa** library was used for some audio processing features. The training was carried out on a cloud hosted NVIDIA P5000 GPU.

The proposed models were trained on the training dataset while monitoring their generalization ability using the validation dataset.
For all the models categorical cross entropy was the loss function used.
As for hyperparameter optimization, both for the training routine and the models themselves, an informal search was carried out since a full grid search was computationally unfeasible, these results show the best configurations found.

Due to the big differences between the models the training parameters like batch size, learning rate, optimizer settings, number of epochs were different from model to model, they are listed in the following table:

Model name	**Optimizer**	**Initial learning rate**	**Weight decay**	**Schedule**	**Max epochs**	**Batch size**
CNNLSTM	Adam	10^{-4}	-	lr divided by 2 every 5 epochs	100	64
FCMFCC	AdamW	10^{-4}	10^{-6} for the feature learning part	lr divided by 10 every 10 epochs	150	64
CNNATT	AdamW	10^{-4}	10^{-7}	lr divided by 10 every 20 epochs	200	32
HUBERTFT	Adam	10^{-4} for the classifier 10^{-5} for the finetuned parts	-	lr divided by 10 every 3 epochs	30	16
MMFUSION	AdamW	10^{-5} for the classifier 10^{-6} for the finetuned parts	10^{-6}	lr divided by 10 every 5 epochs	30	16

Table 3: Training hyperparameter settings for the different models

The learning rate schedule that was used consisted in dividing the the learning rate by a parameter ρ after k epochs of no improvement on the validation loss.
The models were trained for a fixed number of epochs but weights of the best performing epoch were saved, this is an early stopping approach employed to keep the models from overfitting too much.

By tracking the accuracy of all the models during the training routine we obtain the following results:

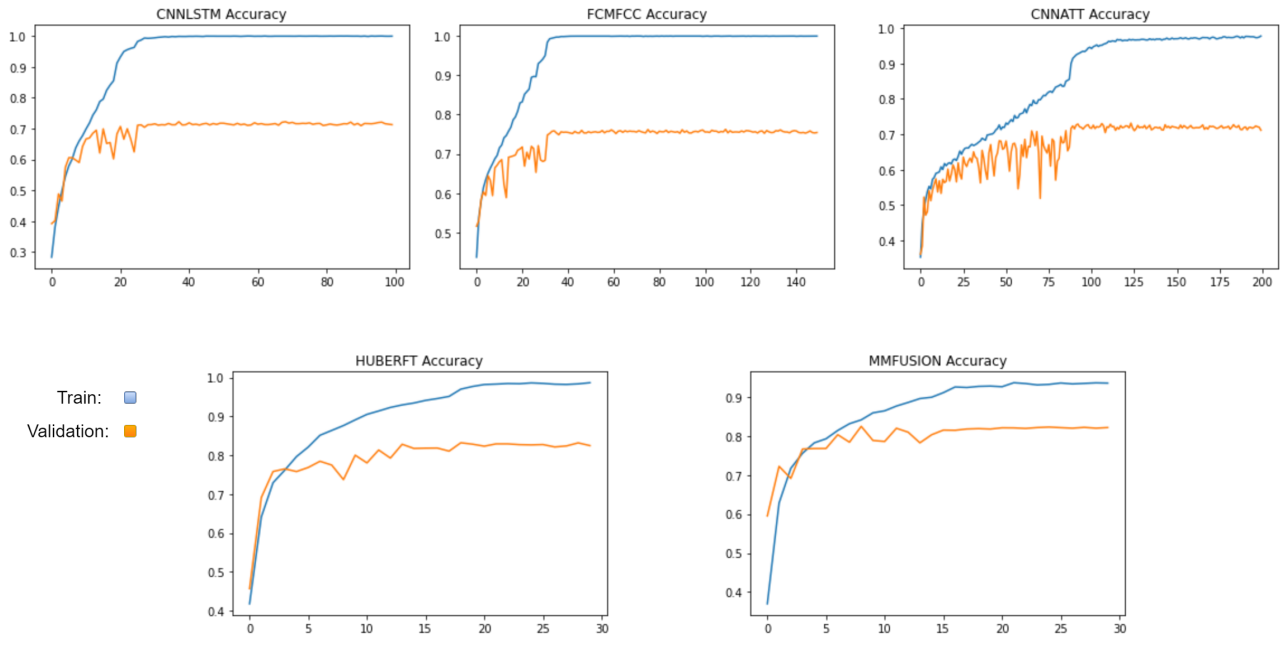

Figure 11: Training curves for the various models

The metrics used to assess the performance was multiclass accuracy and F1 macro, both of them are fairly similar since the dataset is quite balanced.

The final results can be found in the table below:

Model	**Accuracy**	**F1 Macro**
CNNLSTM	0.7220	0.7279
FCMFCC	0.7605	0.7683
CNNATT	0.7313	0.7291
HUBERTFT	0.8297	0.8353
MMFUSION	0.8244	0.8280

Table 4: Evaluation of the early stopped models on the validation set

4 Discussion on the chosen solution

4.1 Final model training

The model that was chosen to be evaluated on the provided test dataset was **MMFUSION**.
Despite the marginally better performance of HUBERTFT we chose MMFUSION due to the lower number of parameters (around 185 millions compared to 315 millions) and its more informative and malleable multimodal structure. Furthermore if we look at the training graphs it definetly looks like HUBERTFT is overfitting much more to the training dataset while having only a slightly better performance on the validation set.

Once a model has been chosen we can train it on the entire labeled dataset, however in doing so we don't have a validation dataset anymore to lead the learning rate reduction.
As a consequence a learning rate reduction schedule similar to the one followed during the model selection step was imposed. In practice the model was trained for 30 epochs with the same settings as in Table 3 and the learning rate was reduced by 10 at epochs number 15 and 25.

4.2 Model strengths and weaknesses

The MMFUSION model has many interesting properties due to its multimodal nature. A multimodal model is usually more robust to catastrophic failure, even in this case where two modalities are synthesized from an original one.
In particular such a model should be less susceptible to adversarial attacks[36] trying to fool the waveform processor, stuff like imperceptively altering (from the human point of view) an utterance of class A to be recognized as class B, since the Mel spectrogram and MFCCs processor would not necessarily be fooled in the same way.

The way in which multimodality fusion was implemented, using the Unified Attention block, is also extremely flexible; adding other interesting modalities like audio transcriptions, visual information would not require much work to be done.
Multimodality also allows parallelization of the computations, in fact the most computationally expensive part of the model is the initial one, which extracts features from the various modalities, however these parts are actually independent until the fusion part and can be computed even on different processing units/machines.

Talking about the generalization ability of the model, the fact that we are using pretrained models trained on a lot of data under the hood, like HuBERT and ConvNeXt, would for sure help the model with data which does not follow exactly the same distribution as the given datasets and so would definitely outperform models trained from scratch in a real world setting.

While the implemented model was forced to process fixed length utterances the only element which enforces this constraint is the Layer Normalization layer in the UA blocks, changing it to work on arbitrary length sequences would only slightly lower the performance of the model on this particular dataset where utterance length is almost entirely under 5 seconds, but such a loss would probably

be recouped in a more heterogeneous dataset.

A possible critique of the model is the fact that the audio information has to be processed to obtain first the mel spectrogram and then the MFCCs and this might be slow, however in a real world implementation such a task could also be carried out in hardware if speed is the utmost concern.

As for the weak points of the model it is still quite an heavy model, with around 185 millions parameters, and requires a fair amount of computing power even for doing inference, so it could not be deployed onto smart devices without a good GPU/TPU unit, unless the computation was done on the cloud.
As it will be explained in more details in the next section the "top-1 prediction" correctness of the model is also not reliable enough to base decisions on it, so in a real world scenario we would have to take soft decisions based on the output probability distribution over all emotions.

4.3 Dataset biases and problems

As touched upon in the introduction, the SER task and emotion recognition in general is hard, with a good chunk of the problems originating from the data side.

First of all we have to consider the fact that our classification model has to abstract from the information which is due to the individual speaker variability and secondly from the recording conditions.
Both of these biases have kind of been solved by ASR systems, however the speaker variability one could be harder in the SER task since emotional variability is less well recognizable.

Using a certain dataset can easily introduce biases into our models which heavily impacts the model's generalization capabilities. When our tasks deal with real world people we should be sure that the dataset is representative of the actual population.
Common biases introduced by datasets are gender, age, ethnicity, nationality and education biases and the dataset which is provided isn't exempt from them.
Gender is a glaring issue of the TESS part for example, were we have only female speakers.
Age is a problem for the entire dataset, in fact most of the clips come from adults in the 20-50 age range. In particular there are no audio clips of underage participants in the entire dataset.

While we are able to objectively recognize the fact that a set of emotions under which we can classify any data point exists, the boundary lines between classes are subjective.
This subjectivity originates even from the supposed ground truth of the labeled dataset, with most of them being labeled by a set of judges of different size.
Let's look at the originating datasets from which the challenge dataset was obtained to see how they differ in this regard:

- CREMA: 2042 judges of various ethnicity and age, each clip was rated around 10 times
- TESS: no judges, emotion based only on speaker intention
- RAVDESS: 247 judges of unspecified characteristics; each clip was rated 10 times
- SAVEE: 10 judges, 5 young males and 5 young females; each clip was rated by all the judges

Another relevant question from the labeling point of view is: how much should the linguistic content contribute to a classification towards a certain class? How much should instead be due to the para-linguistic content, like intonation, pitch etc.?
For example how should the phrase "I'm angry", spoken in a dejected and sad tone be classified from the human point of view, according to the content or expressiveness?

What makes everything worse is that these datasets are extremely small compared to classical datasets in other tasks, this is due to the high cost incurred in labeling them.

We have talked about problems related to our task which might make it impossible to have a perfect or even reasonable solution (by other fields SOTA standards at least), but still we can't deny the fact that being able to at least partially solve it would be useful.

A possible solution could be to consider a top-k accuracy along the possible emotional categories and when using a system which needs this emotional state information the system itself should weigh the top-k emotions based on the network's confidence.

4.4 Considerations on scalability and deployment in a real world scenario

If we were to use this model in a real world scenario it would probably be used as part of a reactive system where it is continuously or intermittently receiving auditory information and decisions have to be taken based on the predicted emotion.
Whether the model is able to process arbitrary length audio utterances or not we could use a sliding window paradigm to obtain a prediction every k seconds over which to guide our system's behaviour.

As for scalability it would be reasonable to deploy the model in the cloud to prevent having to use powerful devices on the edge. A partial cloud deployment was already implemented, in fact the model weights were saved on a Microsoft Azure Blob container.
Deploying an entire cloud pipeline would entail the following steps:

- Instantiate a cloud computing virtual device with a powerful GPU, even better if multiple GPUs are provided so we can parallelize our system to work at scale
- Design an input/output data pipeline through which the cloud service can receive the auditory data (which could be preprocessed on the device or not) and answer with a prediction

5 Closing remarks

Various architectures of different conceptual complexity were implemented, from simple recurrent/convolutional networks to more sophisticated attention models even over different modalities. It turned out that the models which performed the best were those which incorporated pretrained models and finetuned them on the given dataset.
In the end the results obtained were not spectacular, however this result wasn't unexpected due to the nature of the SER problem as a partially subjective task with many possible, and one could say equally valid, interpretations.

Technical considerations aside, this challenge has been an amazing opportunity for me to bridge the gap between the theoretical and practical part of the field of machine learning.
In particular it allowed me to experience what it means to be faced with a problem in a field you have never worked on (audio/speech processing) and thus being forced to read the scientific literature from the bottom up, building from the earlier results to the most recent advances. Following this thread it taught me how to implement the innovations presented in scientific papers, which are often presented quite theoretically.

References

[1] S. M. Mohammad. "Sentiment Analysis: Detecting Valence, Emotions, and Other Affectual States from Text". In: *CoRR* abs/2005.11882 (2020). arXiv: 2005.11882.

[2] M. Soleymani et al. "A survey of multimodal sentiment analysis". In: *Image and Vision Computing* 65 (2017). Multimodal Sentiment Analysis and Mining in the Wild Image and Vision Computing, pp. 3–14. ISSN: 0262-8856.

[3] J. Jayalekshmi and T. Mathew. "Facial expression recognition and emotion classification system for sentiment analysis". In: *2017 International Conference on Networks Advances in Computational Technologies (NetACT)*. 2017, pp. 1–8.

[4] X.-W. Wang, D. Nie, and B.-L. Lu. "Emotional state classification from EEG data using machine learning approach". In: *Neurocomputing* 129 (Apr. 2014), pp. 94–106.

[5] S. Buechel and U. Hahn. "Emotion Analysis as a Regression Problem — Dimensional Models and Their Implications on Emotion Representation and Metrical Evaluation". In: Aug. 2016.

[6] F. Eyben et al. "The Geneva Minimalistic Acoustic Parameter Set (GeMAPS) for Voice Research and Affective Computing". In: *IEEE Transactions on Affective Computing* 7.2 (2016), pp. 190–202.

[7] S. K. Bhakre and A. Bang. "Emotion recognition on the basis of audio signal using Naive Bayes classifier". In: *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. 2016, pp. 2363–2367.

[8] Y. Chavhan, M. Dhore, and P. Yesaware. "Speech emotion recognition using support vector machine". In: *International Journal of Computer Applications* 1.20 (2010), pp. 6–9.

[9] H. Zheng and Y. Yang. "An Improved Speech Emotion Recognition Algorithm Based on Deep Belief Network". In: *2019 IEEE International Conference on Power, Intelligent Computing and Systems (ICPICS)*. 2019, pp. 493–497.

[10] B. Schuller, G. Rigoll, and M. Lang. "Hidden Markov model-based speech emotion recognition". In: *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03)*. Vol. 2. 2003, pp. II–1.

[11] H. Hu, M.-X. Xu, and W. Wu. "GMM Supervector Based SVM with Spectral Features for Speech Emotion Recognition". In: *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07*. Vol. 4. 2007, pp. IV-413-IV–416.

[12] R. A. Khalil et al. "Speech Emotion Recognition Using Deep Learning Techniques: A Review". In: *IEEE Access* 7 (2019), pp. 117327–117345.

[13] T. M. Wani et al. "A comprehensive review of speech emotion recognition systems". In: *IEEE Access* 9 (2021), pp. 47795–47814.

[14] S. Latif et al. "Direct modelling of speech emotion from raw speech". In: *arXiv preprint arXiv:1904.03833* (2019).

[15] A. Aftab et al. "Light-SERNet: A lightweight fully convolutional neural network for speech emotion recognition". In: *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2022, pp. 6912–6916.

[16] J. Zhao, X. Mao, and L. Chen. "Speech emotion recognition using deep 1D & 2D CNN LSTM networks". In: *Biomedical signal processing and control* 47 (2019), pp. 312–323.

[17] C. Etienne et al. "Cnn+ lstm architecture for speech emotion recognition with data augmentation". In: *arXiv preprint arXiv:1802.05630* (2018).

[18] S. Latif et al. "Multi-task semi-supervised adversarial autoencoding for speech emotion recognition". In: *IEEE Transactions on Affective Computing* (2020).

[19] J. Deng et al. "Sparse autoencoder-based feature transfer learning for speech emotion recognition". In: *2013 humaine association conference on affective computing and intelligent interaction*. IEEE. 2013, pp. 511–516.

[20] S. Latif et al. "Variational autoencoders for learning latent representations of speech emotion: A preliminary study". In: *arXiv preprint arXiv:1712.08708* (2017).

[21] Y. Zhang et al. "Attention based fully convolutional network for speech emotion recognition". In: *2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. IEEE. 2018, pp. 1771–1775.

[22] P. Tzirakis et al. "Speech emotion recognition using semantic information". In: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2021, pp. 6279–6283.

[23] L. Pepino, P. Riera, and L. Ferrer. "Emotion recognition from speech using wav2vec 2.0 embeddings". In: *arXiv preprint arXiv:2104.03502* (2021).

[24] L.-W. Chen and A. Rudnicky. "Exploring Wav2vec 2.0 fine-tuning for improved speech emotion recognition". In: *arXiv preprint arXiv:2110.06309* (2021).

[25] M. Müller. *Information Retrieval for Music and Motion*. 2007.

[26] K. Qian et al. *Unsupervised Speech Decomposition via Triple Information Bottleneck*. 2020.

[27] H. Sun et al. *How Speech is Recognized to Be Emotional - A Study Based on Information Decomposition*. 2021.

[28] K. Simonyan, A. Vedaldi, and A. Zisserman. *Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps*. 2013.

[29] C. Szegedy et al. *Going Deeper with Convolutions*. 2014.

[30] A. Vaswani et al. "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017.

[31] W.-N. Hsu et al. *HuBERT: Self-Supervised Speech Representation Learning by Masked Prediction of Hidden Units*. 2021.

[32] H. Zou et al. "Speech Emotion Recognition with Co-Attention Based Multi-Level Acoustic Information". In: *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2022, pp. 7367–7371.

[33] Z. Yu et al. *Multimodal Unified Attention Networks for Vision-and-Language Interactions*. 2019.

[34] Z. Liu et al. *A ConvNet for the 2020s*. 2022.

[35] D. S. Park et al. "SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition". In: *Interspeech 2019*. ISCA, Sept. 2019.

[36] A. Kurakin, I. Goodfellow, and S. Bengio. *Adversarial Machine Learning at Scale*. 2016.