Multilingual Speech Emotion Recognition via Stacked Convolutional Autoencoders

Alice Schiavone

Project developed for course of "Audio Pattern Recognition"
at Università degli Studi Statale di Milano, held by Professor Stavros Ntalampiras

Abstract

This study aims to find a solution to the problem of Speech Emotion Recognition on multiple languages. We try to find a different approach from the standard feature-extraction method by implementing a Stacked Convolutional Autoencoder, which is supposed to learn a latent vector that catches the most important features of a raw input signal. While doing so, we compare its results with standard models and with different classifiers. Comparing performance and training history of these models is important to see what can be the best way to move in such an difficult task. While some emotions were easier to identify for some languages, we noticed that some bias was also "transferred" from one model to the other, that learned to wrongly classify the same emotion more than once. While not reaching outstanding results, this works shows that unsupervised feature extraction could be the way to follow to help classifiers to learn audio features, without relying entirely on knowledge from domain experts.

Keywords: Stacked Convolutional Autoencoder, Multilingual Speech Emotion Recognition, Semi-supervised Learning

1 Introduction

The problem of Speech Emotion Recognition is one of the many that machine learning is trying to tackle. Like other pattern recognition tasks applied to audio signals, the standard approach before the recent deep learning trend, was to extract features to reduce the dimensionality of the input space and help classifiers to focus on the important information, which requires knowledge from experts dedicated to study that specific field and its applications. Since the increasing availability in more powerful machines that can today compute and parallelize the easy, but many, computations required to train a deep neural network, many have tried to solve Speech Emotion Recognition and related tasks with neural networks and raw input signals. [7] [5]

Because one solution doesn't fit all problems, other problems have emerged: first of all, the lack or variety of data. In fact, most datasets available today do not in any way meet the large number of samples required to adjust the all the weights a deep neural network, that often requires tens of thousands of labeled examples to meet state of the art results. As we will see, most datasets about Speech Emotion Recognition have only some hundreds of examples. Another important concern could be about the language of the given signals: can one be sure that a model trained on English, would reach the same accuracy at classifying emotions of a Romance language like Spanish? In order to trying to find a solution to these, we try to approach the task with a multilingual approach: we merge multiple and diverse datasets of different language to train a single model that we hope achieves decent results on all of the involved languages.

This, of course, raises multiple concerns: how to normalize the data so that the model learns as it should? How to train a model in the high dimensionality of a raw signal, which is only made more complex by having a higher number of examples?

In this study we try to solve the task by implementing a model that, given a dataset made by three different languages (English, Italian and German), can recognize the emotion of the speaker.

2 Previous work

Many studies in the area of Speech Emotion Recognition are able to achieve nice results of classification accuracy when the authors' methods are applied to a mono-linguistic dataset. The methods implemented by the many authors varies a lot, and they tackle many different characteristics and fields of study. This results get better as the complexity and training time of the

models increases, but many problems still remain. One is that of the multi-linguistic and multi-emotional approach, and the other that of extracting features from a raw and noisy signal. We try to achieve is a method that, with few resources, reaches a relevant result in recognising a multitude of emotions in 3 different languages. While doing so, we compare the proposed model to different simple models to notice the differences and possible reasons for bad performance.

Usually experts choose to handle the problem with LSTM networks and other similar architectures that take into consideration the natural speech linearity in time. This is clearly the right approach to handle something that for humans would not make sense, if not played in the right sequence. However, during tests with LSTM and others, no significant improvement in performance was noticed. A standard CNN achievied the same, if not better, results in much lower computational time. To avoid clutter, LSTM experiments have been removed from this study.

3 Data

Reading through the papers that present the chosen datasets, we can read the following:

- EMOVO, an Italian Emotional Speech Database that is "built from the voices of up to 6 actors who played 14 sentences simulating 6 emotional states (disgust, fear, anger, joy, surprise, sadness) plus the neutral state. [...] In total 588 records have been archived: for each of the 6 actors 98 sentences were recorded, corresponding to 14 sentences spoken in 6 emotional states plus the neutral state. [...] The recordings were performed with a sampling frequency of 48 kHz, 16 bit stereo, wav format." [3]

- EMODB, " Ten actors (5 female and 5 male) simulated the emotions, producing 10 German utterances (5 short and 5 longer sentences) which could be used in everyday communication and are interpretable in all applied emotions. [...] neutral (neutral), anger (Ärger), fear (Angst), joy (Freude), sadness (Trauer), disgust (Ekel) and boredom (Langeweile). [...] The speech material comprises about 800 sentences (seven emotions * ten actors * ten sentences + some second versions). [...] Recordings were taken with a sampling frequency of 48 kHz and later downsampled to 16 kHz. " [1]

- RAVDESS, "contains 24 professional actors (12 female, 12 male), vocalizing two lexically-matched statements in a neutral North American accent. Speech includes calm, happy, sad, angry, fearful, surprise, and disgust expressions. [...] Each expression is produced at two levels of emotional intensity (normal, strong), with an additional neutral expression. All conditions are available in three modality formats: Audio-only (16bit, 48kHz .wav), Audio-Video (720p H.264, AAC 48kHz, .mp4), and Video-only (no sound)." [6] Of course, for the purpose of this study, we discard the video data to keep the audio-only data. We also aggregate the two intensity levels as one to match the other datasets.

We can summarize the available information in Table 1 and Table 2.

Name	EMOVO	EMODB	RAVDESS
Language	Italian	German	English
Gender	M/F	M/F	M/F
Emotions	6+1	6+1	6+1
Acted	yes	yes	yes
Sampling	48kHz	16kHz	48kHz
Samples	588	800+	2880

Table 1: Information about the datasets.

Name	EMOVO	EMODB	RAVDESS
Neutral	✓	✓	✓
Sad	✓	✓	✓
Angry	✓	✓	✓
Happy	✓	✓	✓
Fearful	✓	✓	✓
Disgusted	✓	✓	✓
Surprised	✓		✓
Bored		✓	

Table 2: Present emotions in each dataset.

EmoDB doesn't match the standard Big Six emotions, and the 'Surprised' and 'Bored' emotion can not be the substitute of one another. The final dataset is made of 1454 samples, because it excludes the "Surprised" and "Bored" emotions, given that not all of the 3 datasets have them; several examples from selected actors from RAVDESS have also been excluded, because otherwise the training would we greatly unbalanced towards the English language. Testing data is approximately the 30% of the total data. In the end, the training dataset turns out to be balanced in terms of emotions [Img 1], languages [Img 2] and gender [Img 3].

Because we want to evaluate our model on people that are not present in the training dataset, training is done on:

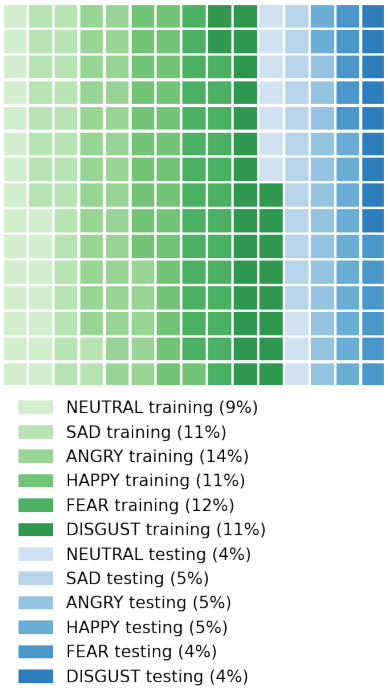


Figure 1: Emotion composition.
Total of 1486 (1050 training, 436 testing). One square stands for 99 samples.

- Actors f1, f2 (F) and actors m1, m2 (M) for EMOVO (Italian)
- Actors 09, 13, 14, 16 (F) and actors 10, 11, 12, 15 (M) for EMODB (German)
- Actors 02, 04, 06, 08 (F) and actors 01, 03, 05, 07 (M) for RAVDESS (English)

and testing is done on:

- Actor f3 (F) and actor m3 (M) for EMOVO (Italian)
- Actor 08 (F) and actor 03 (M) for EMODB (German)
- Actors 10, 12 (F) and actors 09, 11 (M) for RAVDESS (English)

The duration of the samples [Img 4] clearly shows where the dataset divides into 3 parts, and we can see that RAVDESS (the top of the plotted line) has files with a similar duration for its samples, while EMOVO has some outliers with a very long duration and a general variance like EMODB.

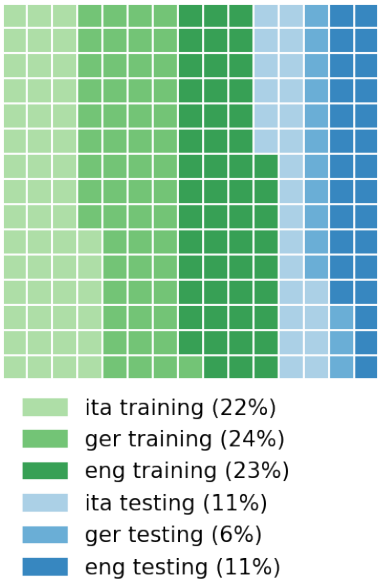


Figure 2: Language composition.
Total of 1486 (1050 training, 436 testing). One square stands for 99 samples.

4 Methodology

To compare the result of the proposed model, we start by defining two other standard supervised learning models. The first is a standard Dense Neural Network , the second is a Convolutional Neural Network, both trained on standard knowledge-base features. The two supervised learning architectures are those presented in [Img 17b] and [Img 17a] (See Section 8), respectively.

In many applications it has been proved that a initial unsupervised learning phase followed by a supervised learning one can improve the model classification accuracy. [4] The proposed model in this study is that of a Convolutional Autoencoder to handle the extraction of a latent vector and a model (among three selected) that assigns a class to the latent vector with respect to the target labels. This means that the first part of the training is unsupervised, because the autoencoder tries to learn the hidden features of the raw signal on the latent vector, and then tries to reconstruct its input. The actual classification takes place later, after discarding the decoder part of the model, so to have only the convolutional layers of the encoder and its final flat latent vector, that encodes the dataset examples to map the input to a lower dimensionality space. This output vector will later be the input to a standard classifier that tries to discern the right class.

We hope that the latent vector learn hidden features that would otherwise need to be extracted manually

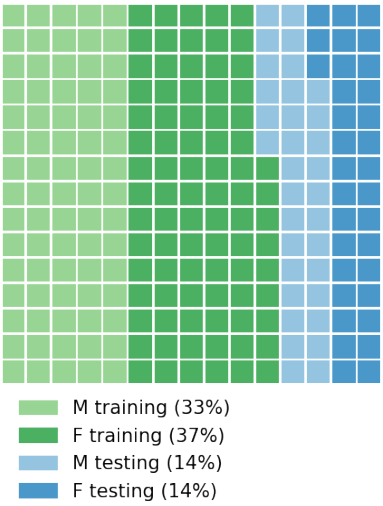


Figure 3: Gender composition.
Total of 1486 (1050 training, 436 testing). One square stands for 99 samples.

based on the knowledge of field experts. Discarding the need of feature extraction, we hope that the generalization needed for a multilingual emotion recognition emerges from the hidden features learnt.

The three classifiers selected for the classification of the latent vector are a Support Vector Machine, a Multi-Layer Perceptron and a Convolutional Neural Network.

The implemented models can be summarized as follows:

- DNN on extracted features (1 model/language)
- CNN on extracted features (1 model/language and 1 model for all 3 languages)
- Convolutional Autoencoder (1 model/language) and:
  - Support Vector Machine on encoded signal (1 model/language) (Radial Basis Function as kernel)
  - MultiLayer Perceptron on encoded signal (1 model/language) (6,6 hidden nodes with ReLU)
  - CNN on encoded signal and extracted features (1 model/language) + (1 model for all 3 languages)

In the end, we hope to get similar or better results than the traditional method of learning on extracted features.

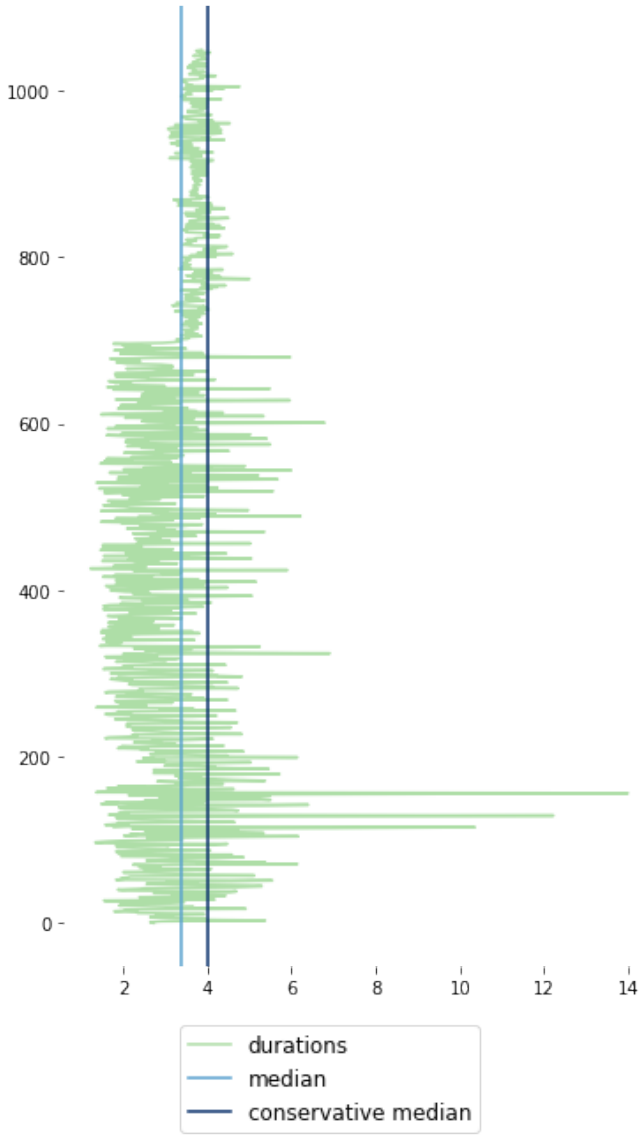


Figure 4: Duration of the samples and their median and conservative (ceil function) median values. The first batch is EMOVO, the middle is EMODB, the last is RAVDESS.

4.1 Preprocessing

Several preprocessing techniques have been applied to the data. The first one is that of downsampling to lower the dimensionality. To avoid any aliasing, the function *scipy.signal.decimate()* has been used, ending with signals with sampling rate equal to 8kHz. To have signals of the same length in order to feed them to the learning models, each audio data point has been cut into different subsignals of 8000 samples, so 1 second worth of signal. Subsignals that didn't reach the length of 8000 samples have been discarded. At the end of this process, we have 2827 data points with 8000 features. The success of the downsampling technique can also be manually verified by listening to the endproduct of the process, which is still comprehensible.

The raw signals have been scaled via *sklearn.preprocessing.MinMaxScaler()* to help the model learn from normalized values. For the Convolutional Neural Network on extracted features, we extracted the feature via Python's library *Librosa*.

The labels have been encoded to a categorical numerical representation as required by *Keras* (which we used to build the networks), mapping each of the six emotions to a integer from 0 to 5.

4.2 Feature extraction

The features extracted, using the Python's library *Librosa* are:

- Log-Mel Spectrogram, with number of Mels = 16.
  We get a spectrogram by computing the Fast Fourier Transform on overlapping segments of a signal to get a representation of the signal loudness. The horizontal axis represents time, the vertical axis represents frequencies, and colors are present to show the intensity of the amplitude. Then scale it by a Mel scale, which is a perceptual scale of pitches based on empirical data. [Img 5]

- MFCCs, with number of MFCCS = 12.
  Discrete Cosine Transform applied to the Log-Mel Spectrogram of a signal. It aims at representing the envelope of the short time power spectrum, which depicts the shape of the vocal tract of a person that filters the sounds they can make.

- Chroma.
  A 12-feature vector which represents the level of energy of each pitch class in a standard chromatic scale.

- Centroid.
  It locates the center of mass of the spectrum, which is related to the impression of brightness of a sound

Concatenated, the extracted features form a matrix (41x16) from the original 8000 features of one 1-second long signal.

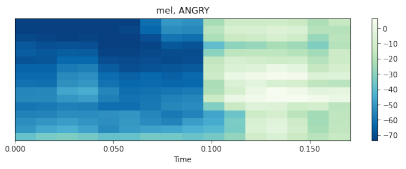


Figure 5: Log-Mel Spectrogram. (Italian, "Angry")

4.3 Evaluation metrics

The evaluation metric on which the models were trained is the *Cohen's Kappa Coefficient* [2]. For this task, accuracy would not be a good metric because during the development of this project, it was very clear that the models were learning a random label, often the most frequent in the set; even if the dataset was mostly balanced, a difference of few samples was the reason for that model to predict always the same label. On the contrary, Cohen's Kappa Coefficient (κ) takes into consideration the possibility of the model agreeing with the true label only by chance.

$$\kappa \equiv \frac{p_o - p_e}{1 - p_e} = 1 - \frac{1 - p_o}{1 - p_e}$$

Cohen's kappa takes two imaginary raters that give their opinion about N samples of C (mutually exclusive) classes. We call p_o the relative observed agreement among raters, while p_e if the hypothetical probability of chance agreement, calculated as

$$p_e = \frac{1}{N^2} \sum_k n_{k1} n_{k2}$$

for k classes, N examples of which we predict the respective labels, and n_{ki} the number of times rater i predicted category k.

For the final evaluations, we also look at the confusion matrices of each model, to get insights about which emotions are similar or more relevant when discussing the a certain language.

At the beginning of this project, a different approach was taken: we started by training all the models on a dataset which included all the 3 different languages. This proved to be a bad approach, because none of the model were able to reach a $\kappa >> 0$ on the validation data. Even worse, some models had a negative κ meaning that the models were not up to the task. While still experimenting with other classification metrics, the *categorical accuracy* of the models were scoring ≈ 0.19.

This clearly means that the models were just "tossing a coin" to decide which of the 6 classes classified that data point, with a probability of $\frac{1}{6} = 0.16$ in the hypotesis of a perfectly balanced dataset. Because this was not the case, in the end each model decided for the most present label for each instance, which usually resulted in a confusion matrix with only one column "light up" [Img 18, 19].

5 Models description

The model architectures are included in each of the following subsections, and so are the parameters and functions used. For each language we trained a model, so three for each model. The first two supervised models where trained on features extracted from the signals after processing them (see Section 4.1). The last unsupervised model was trained on the raw signals (after preprocessing) and the classification was handled by 3 different classifiers: first only on the latent vector "translated" by each encoder, then on the latent vector attached to features extracted from the (preprocessed) signals. In general, the parameters of the models are not many due to the low number of features of the data points, which also saves training time and prevents overfitting.

To encourge the model to avoid making random choices, every model has been trained with adjusted weights for each example to balance the dataset, which have been estimated via *sklearn.utils.class_weight.compute_class_weight* .

The models architectures can be consulted in Section 8.

Plots of the training history are presented to show how much the model was able to learn, by also representing the learning score on the validation data (which was shuffled through learning to get data from every actor randomly). The models metric of evaluation during training is that of Cohen's Kappa (see Section 7), and that is the metric plotted. Results close to 1 means a perfect accuracy independent from random guessing, while those close or even below 0 mean a very bad performance. Most models were trained for 100 epochs with batch size of 16 examples.

5.1 Dense Neural Network

A simple Dense Neural Network trained with *Adam* optimizer and *Sparse Categorical Crossentropy* loss (which is used throughout all the other models), with layers of *Batch Normalization* and *Dropout* at 0.2. It is clear from [Img 6] that the training was very inconsistent among the 3 languages, and in general with pretty bad results both in training and in the validation testing phase. This probably is due to the disruption of the time-series sequence information.

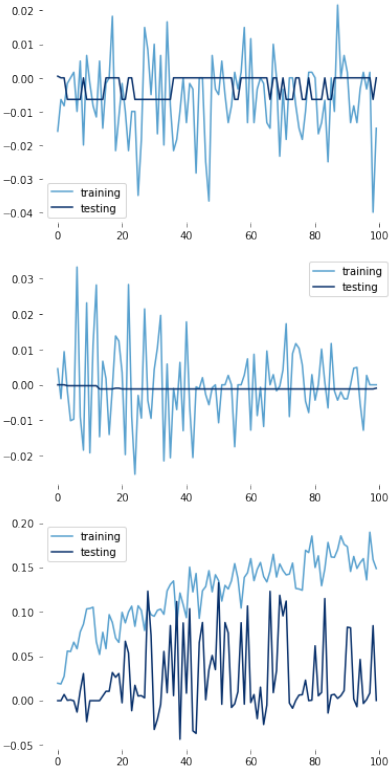


Figure 6: Training history of DNN on features from EMOVO, EMODB, RAVDESS, respectively.

5.2 Convolutional Neural Network

A simple Dense Neural Network trained with *Adam* optimizer and *Sparse Categorical Crossentropy* loss, with *Dropout* at 0.5. It is clear from [Img 7] that the training was very difficult for EMOVO, but more manageable by EMODB and RAVDESS. With more than 100 epochs, no further improvements were noticed, and validation κ remained stable and low.

We can see that on the same architecture, different datasets (or languages) perform differently not only in terms of validation loss, which is very high during all training for Italian (overfitting), but also in terms of accuracy, which was very low for English (underfitting), while German managed to get overall very good results, with respect to the other two datasets.

It is worth noticing that this architecture is the basic proposed solution to the task of Speech Emotion Recognition, so it has been trained on features extracted from each dataset separately, but also on the examples group all together, to have a baseline of comparison with the proposed model. We can see that the training history of

said model [Img 8] follows a trend that could very well be a combination of the previous separated models, but in the worst way: it overfits the already poor results.

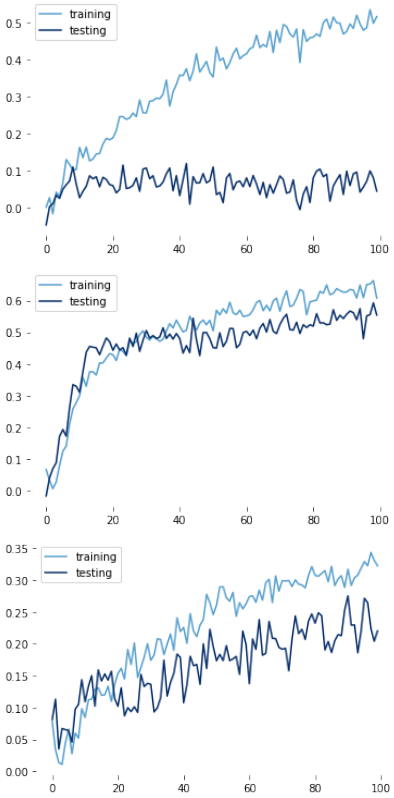


Figure 7: Training history of CNN on features from EMOVO, EMODB, RAVDESS, respectively.

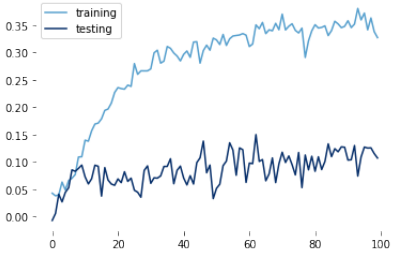


Figure 8: Training history of CNN on features of the full dataset.

5.3 Autoencoder for Feature Extraction

Autoencoders are an example of unsupervised learning. In this context, it has been used to extract features from a 8000 features long signal by learning a 2000 features long latent vector which is positioned as a flat layer between one *encoder* and one *decoder*.

Because the architecture has more than one layer per each part, we call our model a Stacked Autoencoder, or Deep Autoencoder. We compute the loss from the original signal to the reconstructed signal via *Mean Square Error* loss function, and the learning rate optimizer is *adadelta*. [8] The network architecture and parameters can be summarized by Table [3] and Img 17c. For each language, a different SAE with the same architecture has been trained for 100 epochs, batch size of 16 and validation split of 0.2 (with shuffle=True).

At the end of training, we saved only the encoder part of the trained SAE to use it later to encode training and testing data, onto which we conducted standard supervised classification. Like with previous models (DNN and CNN on extracted features), this model also shows the difference in training [Img 9] among different languages while reconstructing the original signal for each training example. Overall, overfitting doesn't occur and the loss is low for every dataset.

Layer	Shape	Size	Stride/Rate	Activation
Conv1D	20/30/40	3	2	ReLU
MaxPooling1D	2			
Dropout			0.2	
Dense	16			
Flatten				
Reshape				
Conv1D.T	40/30/20	3	2	ReLU
UpSampling1D	2			
Conv1D.T	1	3		Sigmoid

Table 3: Parameters for SAE architecture for feature extraction. The first part is the encoder (stacked 3 times), the second the latent vector part, the last is the decoder (stacked 3 times), with a final output layer.

6 Training on encoded signals

Once we obtain an Encoder for each language, we can feed to it our examples and use them for our classification task. To see what work best, we used a Support Vector Machine, a MultiLayer Perceptron, and a Convolutional Neural Network. We evaluate their performance on Cohen's Kappa Coefficient and examine their Confusion Matrix to get more information about each emotion and each language.

6.1 SVM and MLP

These two model are very used and efficient when it comes to Speech Emotion Recognition with selected features. They usually work very well despite their low complexity and small number of parameters. This is also what we want, because to avoid overfitting on a

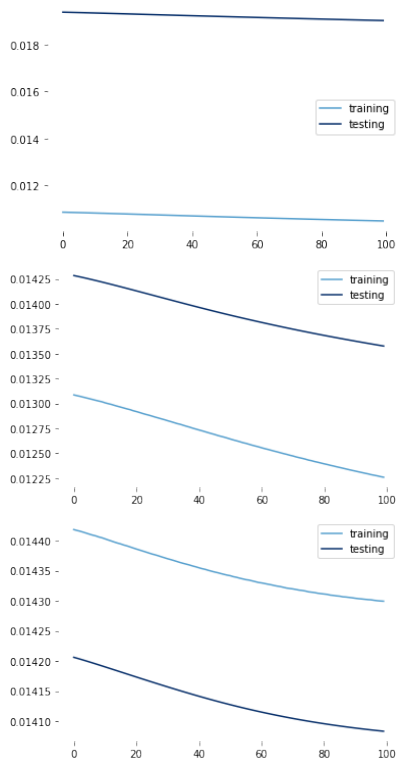


Figure 9: Training history of SAE loss on EMOVO, EMODB, RAVDESS, respectively.

low dimensional space we want to avoid using deeper networks. In our experiments, we tried to classify 6 emotions from a 1 second long signal, which once encoded was made by only 2000 features, with respect to the original 48000 (or 16000, depending on the dataset).

The Support Vector Machine tries to find a hyperplane that separate each example from each other, where *sklearn.svm.SVC* is implemented to test each class with a one-vs-one classification scheme. To map our non-linearly separable data points to a linear space, we used a Radial Basis Function kernel, which seemed to perform better with respect to the other available kernel functions.

The MultiLayer Perceptron from *sklearn MLPClassifier* with activation function *ReLU* for two hidden layers of 6 neurons each, optimized weights with *adam*. If after 200 iterations the model doesn't converge, it stops training.

6.2 Convolutional Neural Network

Because on extracted features the CNN model worked much better with respect to the DNN [Img 18], we relied on one also for classifying the encoded signals. The architecture is fairly simple:

- Conv1D layer with 16 filters, activation ReLU, kernel size of 100 and 10 strides.
- Dropout layer of 0.5
- MaxPooling1D layer with pool size of 3
- Flatten layer
- Dense layer with 6 neurons and *sigmoid* activation function for the classification of 6 emotions

It has been trained for 100 epochs on each language, with weights optimized by *adam*.

6.2.1 On encoded signal per language

The performance of these small classifier, despite the premises, was very poor. From what we can see in Img [19], they are still very language-dependent and tend to classify one emotion for each example, despite the balanced dataset (and class weights). This emotion is usually the *Angry* one. The best performance was on the German language and with CNN. In particular, we can also see from the training history of CNN on each language [Img 10], that the line trends resembles those from CNN on tailored features [Img 7], with even the same EMOVO extreme overfitting.

6.2.2 On encoded signal and features per language

To further explore the relationship between the tailored features and the encoded signal, we train the same CNN used for encoded signals, but attaching to the vector also the extracted features like MFCCs and so on. If this approach improves the performance, we could get better results by extracting features in two ways, before feeding the final vector to the model.

Our training dataset is now made of 2827 examples, with dimensions 166x16 (125 rows from the 2000-features long encoded signal, and the rest from the tailored features). The architecture is the same of the previous CNNs, except for the first layer which is now a Conv2D layer with 16 filters, kernel size of 3x3, with stride equal to 1.

This happens to be the case for most languages [Img 11], and we can notice that EMODB even reaches a very high κ in the last epochs, with a slight overfit on validation data. This overfit is increased on EMOVO and RAVDESS, but the network clearly learns more with respect to both CNN on tailored features [Img 7] and the CNN on the encoded signal [Img 10].

This result encourages us to finally try to classify all the languages in a single dataset.

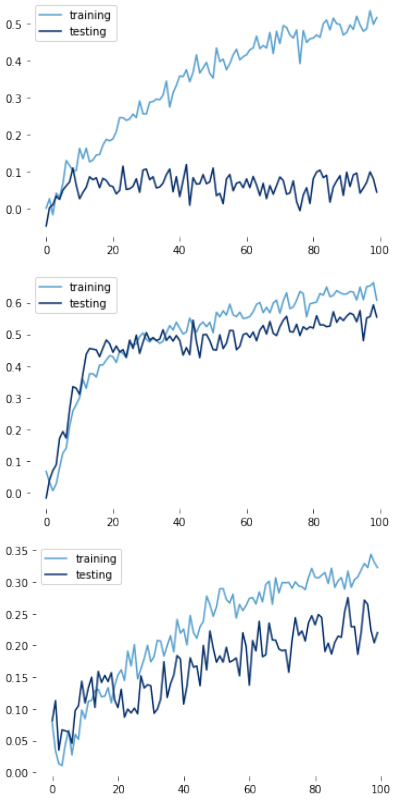


Figure 10: Training history of SAE encoded signal classified by a CNN loss on EMOVO, EMODB, RAVDESS, respectively.

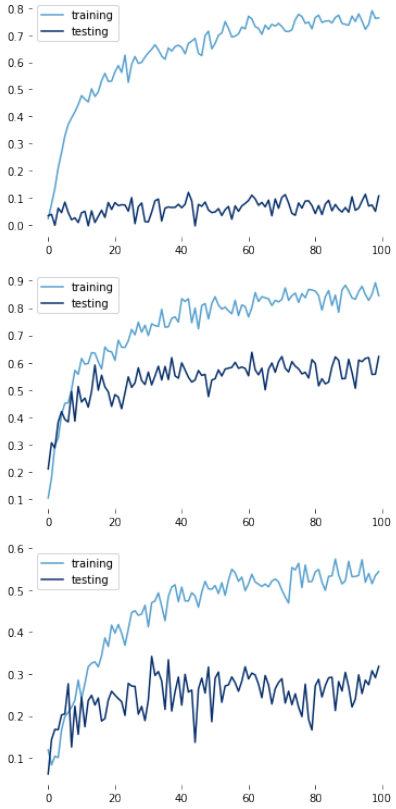


Figure 11: Training history of SAE encoded signal and extracted features classified by a CNN loss on EMOVO, EMODB, RAVDESS, respectively.

6.2.3 On encoded signal and features, all languages

Because we got the best results from using a vector composed of the encoded signal (encoded by the encoder part of a Stacked Autoencoder trained on one language) concatenated with the features extracted from that signal (Mel-Spectogram, MFFCs, Chroma, Centroid), fed as a input to a simple CNN, we try to do the same but using a dataset made by all the languages together. What we want to achieve are results which are better than those of the standard CNN-on-tailored-features approach.

The architecture and its training parameters are the same of the previous model, but we now train for 300 epochs.

As we can see from Img 12, the performance is not as good as one would hope, but its still better than the CNN trained of tailored features with the full dataset [Img 8]. In general, the overfitting is high as before, but the model seems to learn more after the same number of epochs.

7 Results

First we look at the Cohen's Kappa Coefficients κ for each model in Table 4 and Img 13. We can see that very few of the trained models reach decent results on the testing data, and that it is due mostly to the considered dataset. German seems to be the easier language to learn to classify by every other model, but the best result was reach on English by the language-dependent CNN on encoded signals and tailored features.

Because, as was the case for accuracy, κ alone doesn't gives us enough information about the best model, or why it is the best model. We take a look at the confusion matrices of the models [Img 18, 19, 14, 15, 16]. They were all normalized with respect to the predictions and their range of values scaled to be comparable among models.

We can see that most models that didn't use encoded signal + features for classification struggled in almost any case, often misclassifying most emotions as one. The only model that seems to work better than the others (when working only with encoded signals) is the German CNN, which carries a smaller number of test-

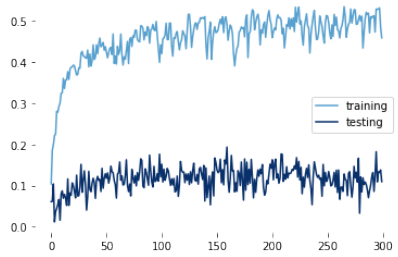


Figure 12: Training history of SAE encoded signal and extracted features classified by a CNN loss on the full dataset.

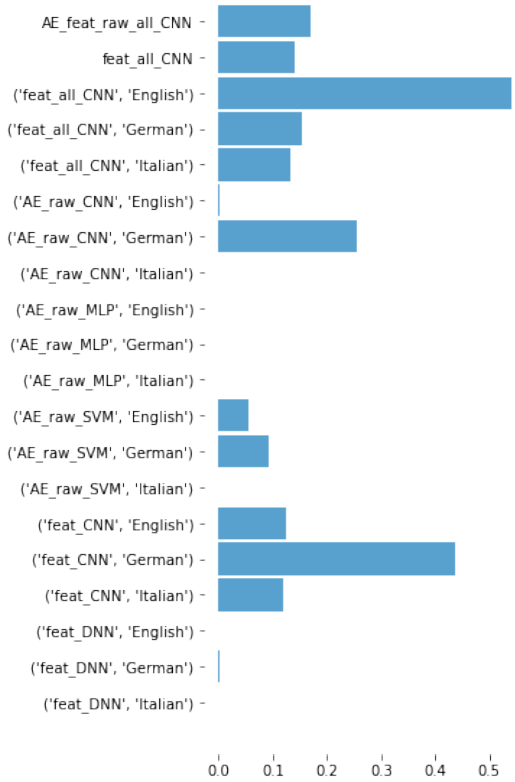


Figure 13: Cohen's Kappa Coefficients for all the models on testing data.

ing examples (almost the half of the other sets). This carries on also when taking into account manually extracted features: its confusion matrix is the best one for all the considered models, with a almost perfect score on "Sad" and "Angry", and lower errors outside the diagonal line. While we cannot say the same for the other languages tested on each respective model, we can see that they perform similarly or better with respect to the extracted-features-only CNN.

Finally Img 15 and Img 16 answer to the initial answer of this study: can be reach better accuracy by learning a latent vector representation of different languages via a Stacked CNN Autoencoder? The answer seems to be yes: the diagonal in the proposed model looks slightly more defined, while making less classification errors. The same can be said for the slight increase in κ. This however is not enough, because we are testing on a really low number of samples, and the statistical relation between the two models should be further investigated.

7.1 Comparison among languages

To check if the final model learns features differently with respect to the data language, we can look at how the proposed model performed on the separated datasets with respect to the full dataset. During the training of the various models, it was clear that even if the data was preprocessed and scaled in the same way, it didn't *teach* the models in the same way. From overfitting to underfitting, the same model performed very differently with respect to the data it was fed. We can suppose it is because of something more that needs to be done in the preprocessing phase to normalize the data, or maybe it is due to the natural characteristics of the presented languages.

This fact alone demonstrate that the challenges of multi-language speech emotion recognition are not easy to overcome. By looking at the confusion matrices, even the bad ones, we can see that each language has a "preferred emotion", onto which each models takes its better guess. Most models perform really well on the "Anger" emotion, and the next favourite is "Sad". This is especially true for German, which was the easiest to recognize: this could be because German speakers are naturally more inclined to give a noticeable change in their speech tone, or because the model learn something entirely different that isn't noticeable, like some bias in the data. It is also interesting to notice in Img 14 that a lot of German "Anger" was misclassified as "Happy", and that "Neutral" was also easier to classify.

When looking at English and Italian we notice the same trend: "Angry" and "Sad" are correct 40% of the time, but an honorable mention is "Fear", which reaches the best result of 43/45%. The error rate is not so high but it is very uniformely sparse, so that shows a lot of indecision by the model. Italian seems to perform slightly better than English, despite its huge overfitting during training in all the models, while English was performing a lot better!

Finally, we can see that the trend of getting right "Angry" and "Sad" follows up in the models trained on the full dataset, but "Sad" reaches a very high accuracy with respect to the other emotions. Something else that

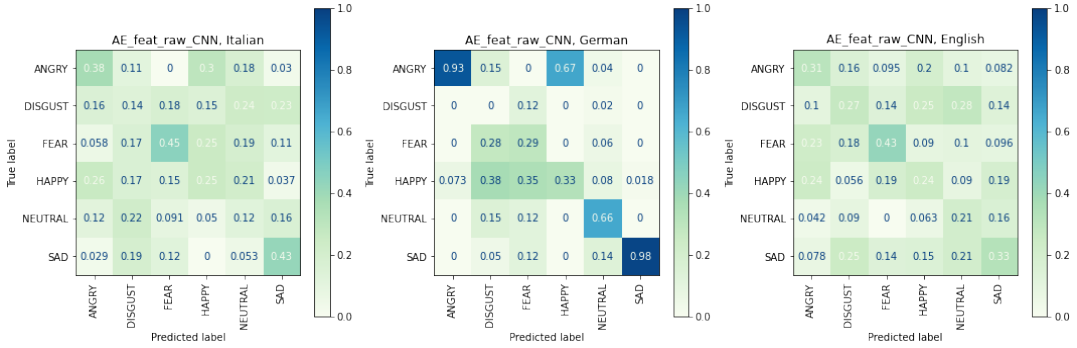


Figure 14: Confusion matrix of CNN on encoded signal + tailored features for each language.

Model	EMOVO	EMODB	RAVDESS
DNN (features)	.0	.002	.0
CNN (features)	.12	.436	.124
CNN (features)		**.141**	
AE + SVM (encoded signal)	.0	.092	.056
AE + MLP (encoded signal)	.0	.0	.0
AE + CNN (encoded signal)	.0	.254	.003
AE + CNN (encoded signal + features)	.133	.153	.542
AE + CNN (encoded signal + features)		**.171**	

Table 4: Cohen's kappa for each model/language on testing data. In bold models trained on all available data.

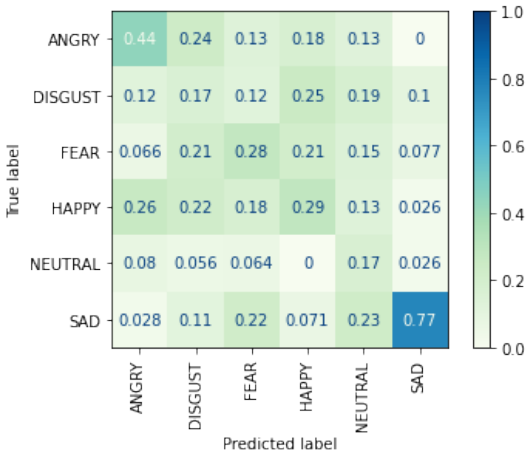


Figure 15: Confusion matrix of CNN on tailored features (all languages).

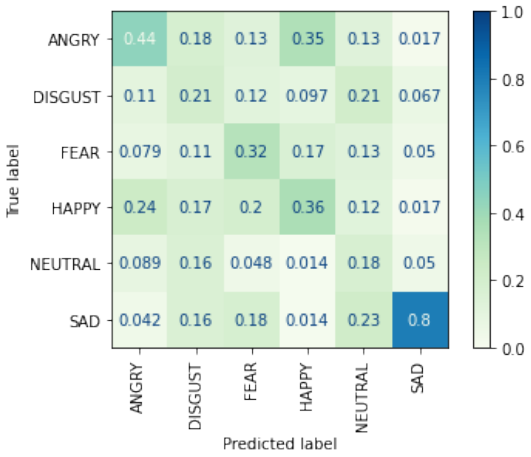


Figure 16: Confusion matrix of CNN on encoded signal + tailored features (all languages).

it is interesting to notice is that in Img 16, the proposed model catches up the German error or misclassifying "Angry" with "Happy", while the same doesn't occur in the standard model confusion matrix [Img 15].

8 Conclusions

The challenge of Speech Emotion Recognition with a variety of languages remains undefeated, but the work done suggest that some form of unsupervised learning is needed in order to improve the performance of models based on features extracted based on knowledge of field experts. For the different way in which models learned,

it is also clear that we must look at a way of normalizing datasets that takes into consideration the nature of the language they are based on, because every language has its own difficulty while training. This however doesn't mean that a difficult training will result in a bad performance, because this task, even with balancing methods, highly depends on the expressivity of the language.

It is also worth noticing that languages that could be considered more clear to understand (emotionally-wise) don't always look the same to a machine: emotions of German were easier to classify, with respect to the Italian ones. Although this could be due to some bias in the data, it would also be interesting to investigate the reasons, maybe with linguistic experts.

While unsupervised learning for extracting features could require a lot a time, it seems to go a little step further in getting models unstuck (that converge to some training accuracy without learning after some number of epochs). This means that tuning of the unsupervised model and in general the preprocessing phase are much more relevant to the classification task than the task itself.

Further research on the topic could involve the study of finding why languages learn so differently on the same model. Another important question arises when looking for the best way to limit the feature space of a language-dependent signal: could different languages need different ways to be mapped onto a smaller feature space? Lastly, why are some emotions easier to recognize across languages, and is there a way to help the model learn the more difficult ones?

An interesting research could be conducted by grouping languages based on their linguistic family (Roman, Germanic, Celtic, etc.), and compare the learned latent vectors, to see if languages from the same family branch would be encoded similarly. However, as is often the case, it would require data acquisition (specific to certain languages) and coherence between datasets, something which has not been done at the time of writing this study.

References

[1] Felix Burkhardt et al. "A database of German emotional speech". In: *9th European Conference on Speech Communication and Technology* 5 (Sept. 2005), pp. 1517–1520. DOI: 10.21437/Interspeech.2005-446.

[2] Jacob Cohen. "A Coefficient of Agreement for Nominal Scales". In: *Educational and Psychological Measurement* 20.1 (1960), pp. 37–46. DOI: 10.1177/001316446002000104. eprint: https://doi.org/10.1177/001316446002000104. URL: https://doi.org/10.1177/001316446002000104.

[3] Giovanni Costantini et al. "EMOVO Corpus: an Italian Emotional Speech Database". In: (May 2014), pp. 3501–3504. URL: http://www.lrec-conf.org/proceedings/lrec2014/pdf/591_Paper.pdf.

[4] Irina Higgins et al. *Early Visual Concept Learning with Unsupervised Deep Learning*. 2016. arXiv: 1606.05579 [stat.ML].

[5] Siddique Latif et al. "Direct Modelling of Speech Emotion from Raw Speech". In: *ArXiv* abs/1904.03833 (2019).

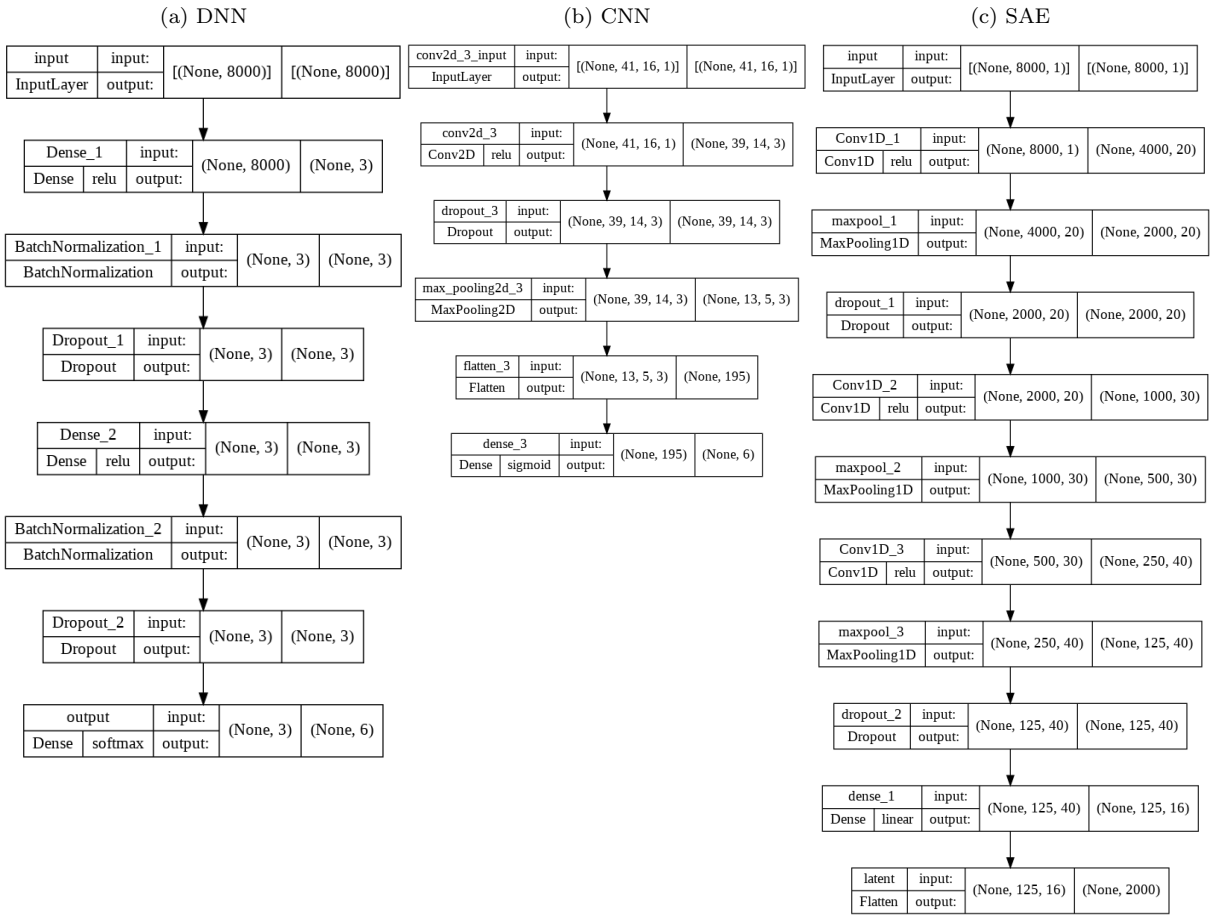
[6] Steven R. Livingstone and Frank A. Russo. "The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS): A dynamic, multimodal set of facial and vocal expressions in North American English". In: *PLOS ONE* 13.5 (May 2018), pp. 1–35. DOI: 10.1371/journal.pone.0196391. URL: https://doi.org/10.1371/journal.pone.0196391.

[7] Panagiotis Tzirakis, Jiehao Zhang, and Björn Schuller. "End-to-End Speech Emotion Recognition Using Deep Neural Networks". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2018), pp. 5089–5093.

[8] Matthew D. Zeiler. *ADADELTA: An Adaptive Learning Rate Method*. 2012. DOI: 10.48550/ARXIV.1212.5701. URL: https://arxiv.org/abs/1212.5701.

Appendix

Figure 17: The architectures for:
(a) Dense architecture trained on features extracted from the signals.
(b) CNN architecture trained on mel-spectrograms extracted from the signals.
(c) Stacked Autoencoder architecture trained raw signals to learn hidden features. In the image, only the encoder part is shown. Refer to Table 3 for the full architecture.



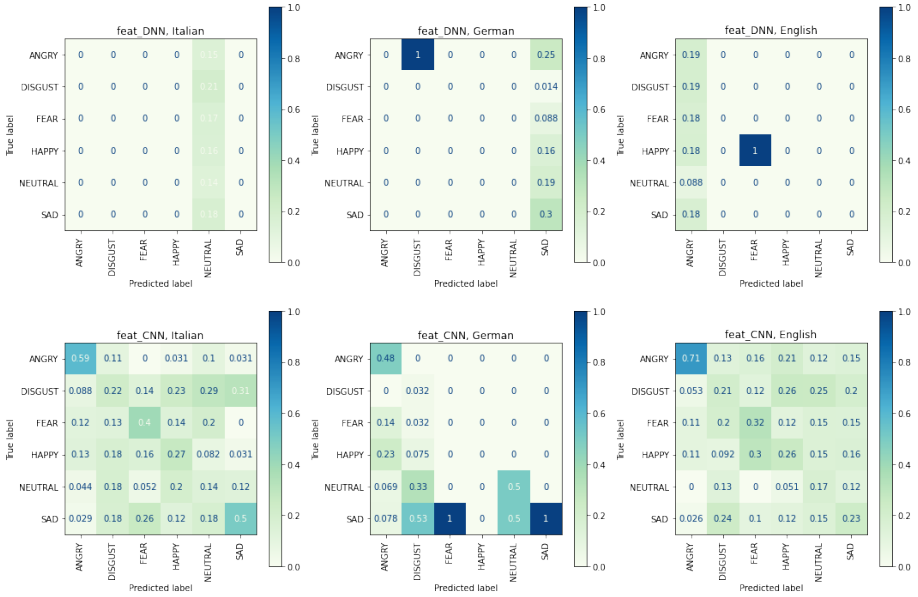


Figure 18: Confusion matrix for DNN [17a] and CNN [17b], trained on each language extracted features, normalized with respect to the predictions vector.

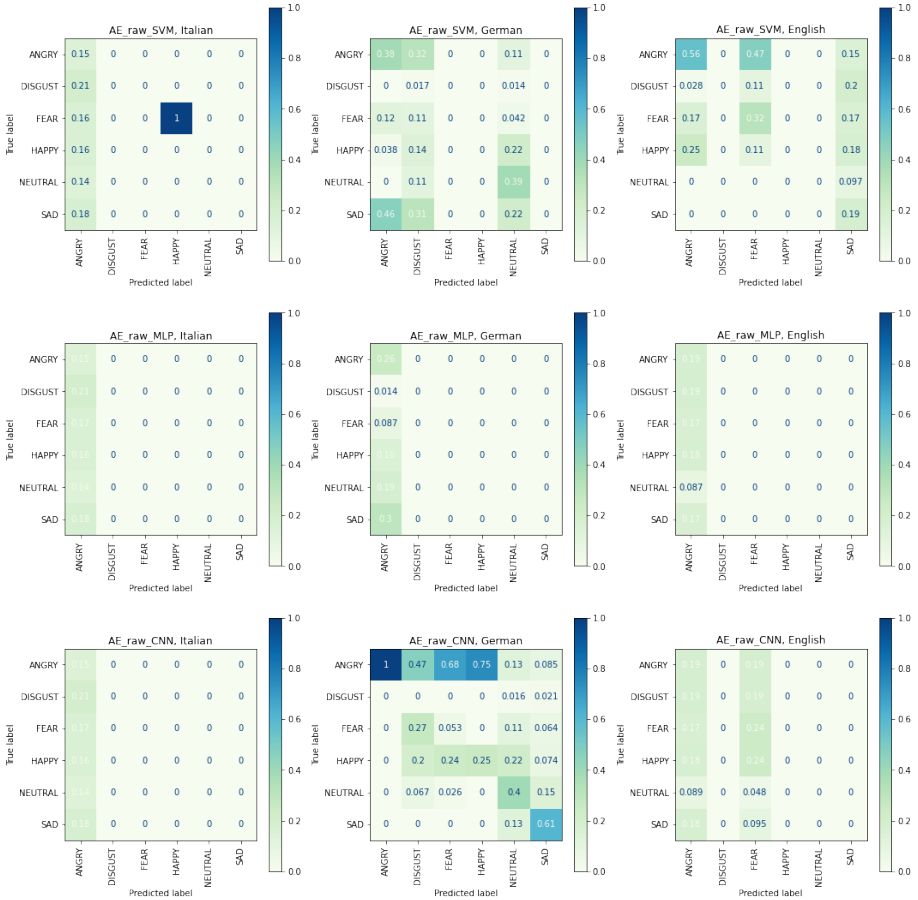


Figure 19: Confusion matrix for SAE [17c] and 3 classifiers (SVM,MLP,CNN), trained on each language encoded signal, normalized with respect to the predictions vector.