UNIVERSITÀ DI PISA

MSc in Computer Engineering

Intelligent Systems Project

EMOTION RECOGNITION ANALYSIS

TEAM MEMBERS:

GIOVANNI BARBIERI

FEDERICO CAVEDONI

ALESSIO DI RICCO

Anno Accademico 2022-2023

Summary

1. Introduction……3

2. Related Works……..4

3. Methods and Experiments……5

4. Multi-Layer Perceptron……6

5. Convolutional Neural Network……10

6. Recurrent Neural Network……15

7. FUZZY Network……..20

1. Introduction

In recent years it has become increasingly important to recognize and analyze the emotions that users feel during human-machine interaction through BRI (Brain-Computer Interface).
By Brain-Computer Interface we mean interfaces capable of converting neuronal activity into electrical signals. BCI systems are widely used in rehabilitation as an intention recognition tool.

In our reference system we have taken as input signals values from an electroencephalography (EEG), carried out using two sensors placed on a helmet, and from a bracelet capable of measuring the electrodermal activity of the body, the heart rate, the temperature body, and blood pulsation.
The data provided by these sensors were then used for the recognition of emotions, through combinations of different values which establish the sensation experienced in each moment. A specific emotion can be seen as a combination of the values of valence (identified by the degree, as a percentage, of pleasure experienced) and arousal (identified by the degree, as a percentage, of intensity of pleasure).

During the analysis carried out, we focused on the study of different types of networks to verify which machine learning model was the most suitable, comparing the degree of accuracy of each algorithm and analyzing the degree of accuracy that each system can produce.

The types of networks that we have dealt with in this project are the following:

- MLP
- CNN
- RNN

The analysis carried out led to the construction of a fuzzy network with rules created ad hoc to develop a system capable of imitating human reasoning and cognition for the recognition of emotions.

2. Related Work

AI emotion recognition is a very active current field of computer vision research that involves facial emotion detection and the automatic assessment of sentiment from visual data. Human-machine interaction is an important area of research where artificially intelligent systems with visual perception aim to gain an understanding of human interaction.

In particular, the research is very active in the analysis and recognition of emotions using images, some examples of how the recognition of emotions through the use of images and dialogues has achieved good results are reported below:

Deep learning has been applied in cognitive wireless framework to the audio-visual emotion recognition system that could automatically identify patients' emotions in the Internet medical care framework. researchers evaluated the system through experiments and proved that the system was beneficial to the development of Internet medical care. Validated performance of medical image fusion model based on deep learning algorithm. They found that the deep learning model could automatically extract the most effective features from the data, and it could enhance the efficiency and accuracy of image processing when used for image fusion. At the same time, increasing the scale of training data could further improve the training accuracy.

Another interesting topic is the analysis of the recent works of automatic facial emotion recognition through deep learning. They found that related scholars focused on exploiting technologies to explain and encode facial expressions and extract these features to accomplish excellent forecasts by computers. Their research results showed the effectiveness of deep learning algorithms.

Twin neural network are also a field of study for speech-based emotion recognition through learning analogy, researchers modeled this relationship on the combined log-Mel and time-modulated spectrum space. The research results showed that the proposed framework could run under non-stationary conditions, and the model prediction could be explained by layer-by-layer investigation of the activation graph. Some related research of emotion recognition or similarity measure methods of recognition have also gained some achievements.

In summary, the deep learning algorithm has been studied in many aspects, such as speech-based image recognition, and it has great application value in speech and image recognition.

3. Methods and Experiments

3.1 Datasets

The K-EmoCon dataset covered in this project was developed with the aim of studying the recognition of emotions during social interactions.
The dataset contains multimodal measurements including audiovisual recordings, EEG, physiological signals acquired through different devices in 16 different discussion sessions between two people lasting 10 minutes each.
Candidates were required to note every 5 seconds the emotions experienced during the conversation in terms of valence, arousal and other 18 additional categories of emotions. At the same time, an external psychologist was assisting the debate, who in turn noted the changes in mood of the two participants.
The dataset aims to obtain a multi-perspective analysis of emotions with the following objectives:
- Extend research on how having multiple points of view on emotional expressions can improve their automatic recognition
- Provide a new opportunity to investigate how emotions can be perceived differently from multiple perspectives, especially in the context of social interaction

K-EmoCon is in fact the first dataset developed that allows to have the point of view of three different subjects: the user, the partner and the external observer.

Figure x: In the image on the left we can see the two participants during the debate. Two smartphones in the middle of the table record the subjects' facial expressions and body movements. On the right, an image taken from the shooting

For the analysis of the system, not each single emotion was examined, but mainly the value of the external arousal.
By measuring the degree of momentary pleasure and its strength, we eliminate as much as possible the subjectivity linked to an emotion: not everyone feels anger, happiness or joy in the same way, thus making our model as generic as possible.

4. Multi-Layer Perceptron

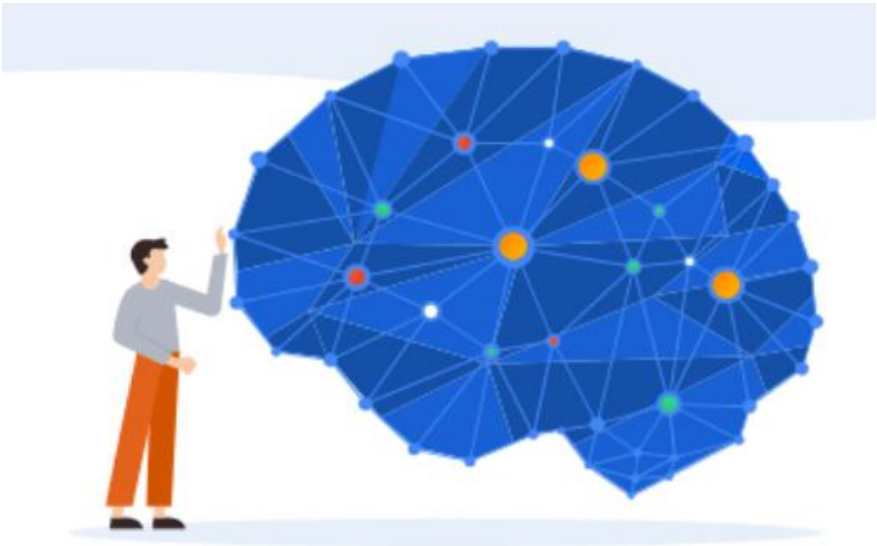
Using an MLP (Multi-Layer Perceptron) network for emotion classification offers several significant advantages. The complex and subjective nature of emotions requires a sophisticated approach to analyze and accurately understand sensory data. An MLP network can learn from a wide range of features extracted from sensors, enabling a rich and detailed representation of emotional information.

The multi-layer structure of an MLP allows for the creation of one or more hidden layers, enabling the learning algorithm to identify complex patterns and relationships between the input features and the emotion classification labels.

Moreover, using an MLP network for emotion classification provides flexibility in dealing with complex and nuanced emotions. Since emotions are classified as integer values ranging from 0 to 4, the MLP can be trained to recognize the nuances within each emotion class. This is particularly advantageous as human emotions can be ambiguous and exhibit different shades that require accurate classification.

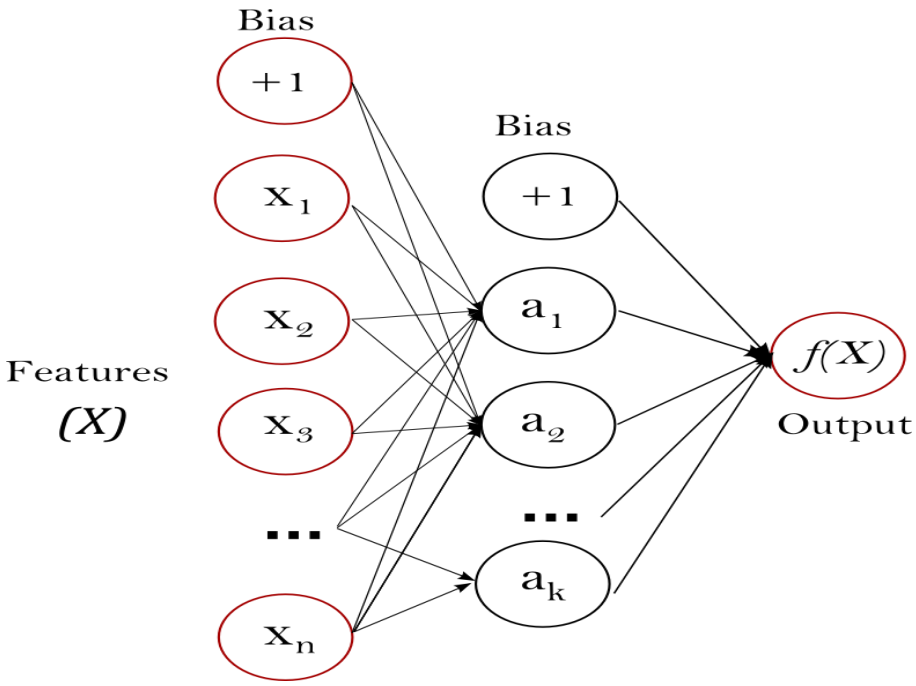
Furthermore, the MLP allows for nonlinear data fitting, enabling the capturing of intricate correlations between input features and emotions. This makes it possible to detect non-intuitive or hidden patterns that could be overlooked by simpler models.

In summary, employing an MLP network for emotion classification offers the ability to learn from a wide range of sensorial features, flexibility in handling complex and nuanced emotions, and the capability to capture nonlinear correlations between input features and emotion classification labels. These features make the MLP a powerful and effective option for emotion analysis.



Architecture

Each hidden layer of the MLP acts as a feature extractor, gradually transforming the input data into higher-level abstractions. The nodes in the hidden layers use activation functions to introduce non-linearity, allowing the network to model and understand complex emotional patterns that may elude linear models. This capacity for non-linear transformation is particularly important in accurately representing the diverse and nuanced nature of human emotions.



Early stopping

To prevent overfitting, we have used the Early Stopping technique.

We compared a first case without Early stopping and a second case with Early stopping, setting the parameter 'Early _stopping = True'. We got the following results:

```
Iteration 109, loss = 13.32579650
Iteration 110, loss = 13.87263295
Training loss did not improve more than tol=0.000100
MPLClassifier score: 0.49411997885835096
```

The first case without Early stopping performs 110 iterations and has a final score of 0.4941.

```
Iteration 31, loss = 17.07245163
Validation score: 0.608456
Validation score did not improve more than tol=0.000100
MPLClassifier score: 0.6043208245243129
```

In this second case, in which we use the early stopping technique, the algorithm uses 31 iterations and has a final score of 0.6043.

We can observe that the algorithm converges with much fewer iterations and also has a greater generalization capacity. We can therefore say that this technique had its benefits and we actually managed to avoid overfitting.

Tuning using GridSearchCV

We then proceeded to optimize our network by tuning the hyperparameters. For the MLP, we used GridSearchCV, testing various parameter combinations as shown in the following figure.

```
param_grid = {
    'hidden_layer_sizes': [200, 500, 1000],
    'activation': ['relu', 'tanh'],
    'solver': ['sgd', 'adam'],
    'learning_rate_init': [0.001, 0.01, 0.1],
    'max_iter': [100,200,500],
}
```

Grid search tests all possible combinations and returns the one that yielded the best performance. In our case, the optimal combination was found to be as follows:

- Activation: relu
- Hidden_layer_sizes: 1000
- Learning_rate_init: 0.001
- Max_iter: 500
- Solver: adam

```
Migliore accuratezza: 0.6356589147286821
```

Using MLP we therefore obtained an excellent accuracy score.

Tuning was very useful in achieving these results, which led us to a score of 0.6356.

Convolutional Neural Network

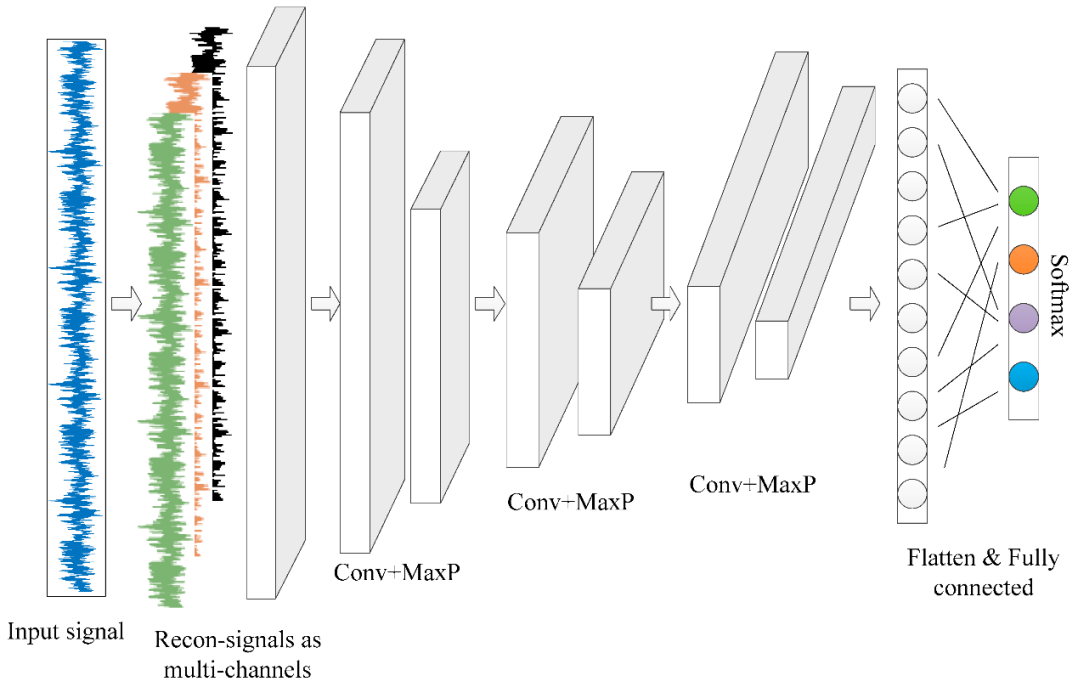
Using a CNN (Convolutional Neural Network) for emotion classification could be an excellent choice for various reasons. Although CNNs were initially developed for image processing, their application can be effectively extended to the classification of sensor data, such as signals acquired from devices like electroencephalograms (EEG) or accelerometers.
CNNs have the ability to automatically learn salient features within sensor data that can be indicative of different emotions. By applying convolutional filters, a CNN can detect relevant patterns, such as frequency changes or distinctive temporal patterns present in the sensor signals.

Architecture

Our CNN network has been structured as follows:

1. Conv1D: a convolutional layer that uses 32 filters of size 3 with ReLU activation.
2. MaxPooling1D: a MaxPooling layer with a window size of 2.
3. Conv1D: another convolutional layer similar to the first one, with 32 filters of size and ReLU activation.
4. MaxPooling1D: a second MaxPooling layer with a window size of 2.
5. Flatten: a layer that transforms the input data into a one-dimensional vector.
6. Dense layer with 100 units
7. Dense: a densely connected layer that performs the final emotion classification.

The parameters of each layer has been subsequently tuned.



These layers constitute the architecture of our model and are designed to extract, refine, and combine information from the input data to achieve accurate emotion classification.

The inclusion of a network architecture that involves a pattern of convolutional layers followed by pooling layers, such as convolution-pooling-convolution-pooling, followed by a flatten layer and a densely connected layer, is a good pattern for emotion classification using a CNN (Convolutional Neural Network).

The first two convolutional layers play a crucial role in extracting salient features from the input data. By using convolutional filters, these layers identify relevant patterns and features within the sensor data that can be indicative of different emotions. The convolution operation allows the network to learn the temporal distribution of features and their interactions.

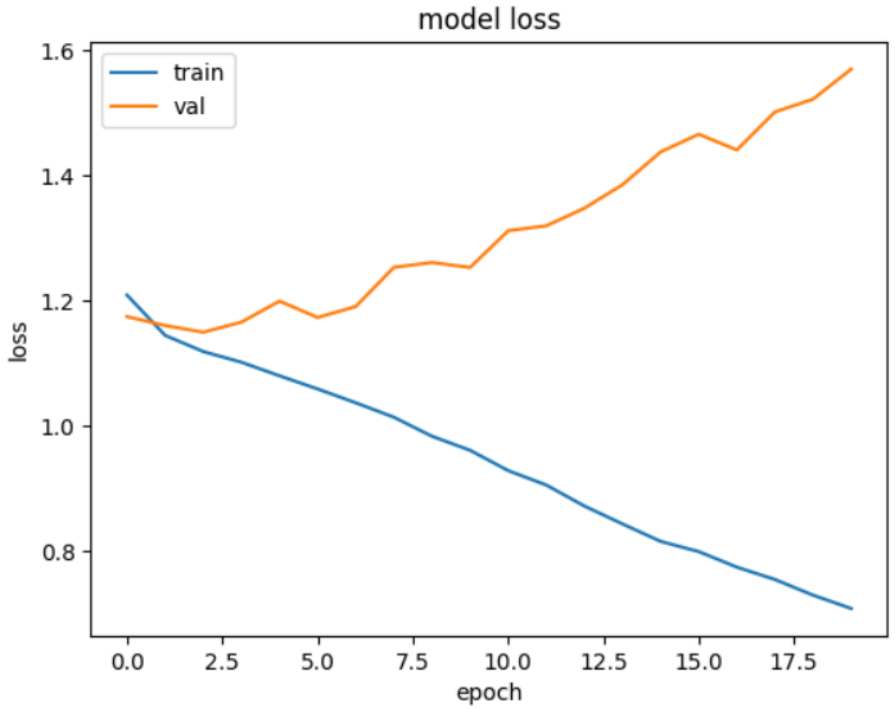
Pooling layers subsequently reduce the dimensionality of the extracted features, enabling better generalization and mitigating overfitting. Pooling aggregates the most significant feature information and reduces the spatial dimension, creating a more compact and abstract representation of the data. This allows the network to focus on key features and diminish the influence of noise or irrelevant details.

Finally, the flatten and densely connected layers serve as the final classifier. The flatten layer transforms the extracted features into a one-dimensional vector, while the densely connected layer combines the feature information to perform the final emotion classification. This combination of layers enables the network to adapt to the specific task of emotion classification and produce accurate predictions.

Overall, the architecture that involves convolutional layers followed by pooling layers, followed by a flatten layer and a densely connected layer, is an appropriate approach for emotion classification using a CNN. This configuration enables the network to progressively extract meaningful information, learn complex representations, and perform accurate emotion classification based on sensor data.

Early stopping and dropout

When we went to train the network we put a data split for the validation (10%) in order to detect the possible presence of overfitting, plotting the loss of the training and validation we get the following result.



As we can see, the validation loss initially decreases and then begins to increase over time, while the training loss continues to decrease, this is a clear case of overfitting.

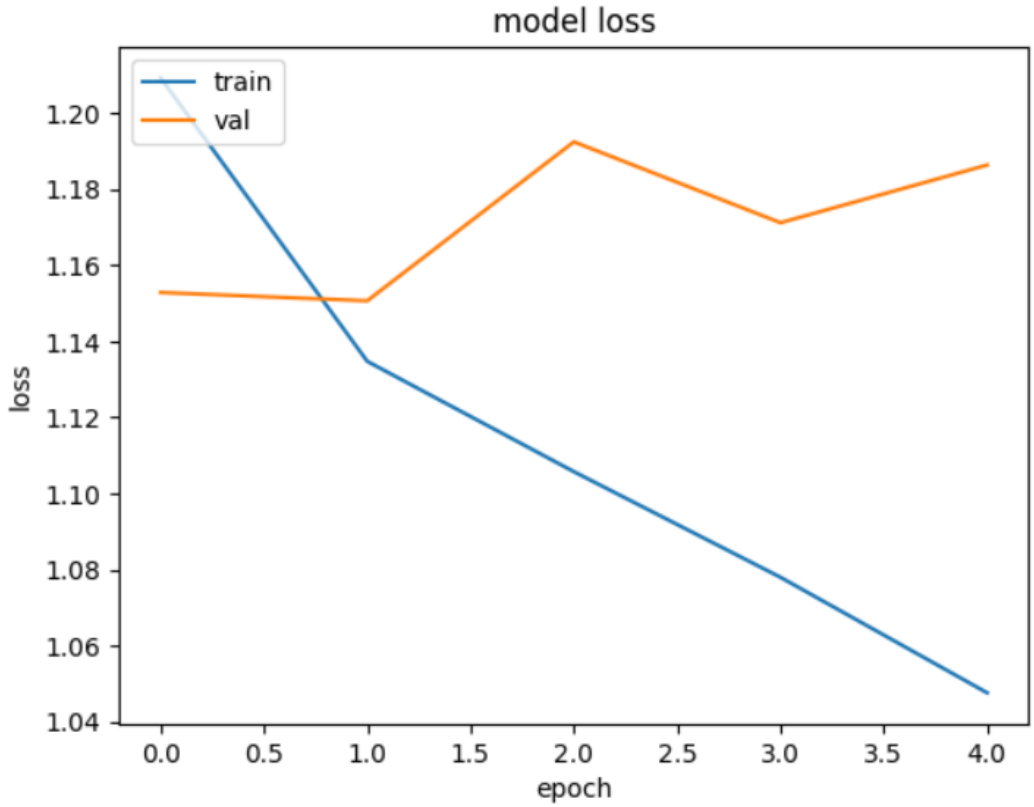
To handle overfitting we used the early stopping method in this way:

```
early_stopping = EarlyStopping(monitor='val_loss', patience=3, restore_best_weights=True)

history = model.fit(X_train_scaled, y_train, validation_split=0.1, batch_size=10, epochs=20, verbose=1, callbacks=[early_stopping], shuffle = True)
```

Training is stopped when the validation loss does not decrease for three consecutive epochs.

Using this technique and making the graph again in which we compare training and validation loss over time, we get the result we wanted, i.e. the training is stopped when the validation loss starts to have an increasing trend.



Since the network takes very few epochs to reach the best generalization capacity, after just 5 epochs we have to stop otherwise we would already be overfitting, so we thought that our network has a model that is too complex for our dataset and risks learning by heart the training data.
We therefore decided to further simplify the architecture of our network by removing the second layer of Conv and Pooling layers. Furthermore, always with the intention of avoiding overfitting, we have also added a dropout layer.

Our CNN network has been structured as follows:

1. Conv1D: a convolutional layer that uses 32 filters of size 3 with ReLU activation.
2. MaxPooling1D: a MaxPooling layer with a window size of 2.
3. Dropout: a Dropout layer with a rate of 0.01.
4. Dense layer with 100 units
5. Dense: a densely connected layer that performs the final emotion classification.

Parameters of each layer has been tuned subsequently.

By repeating the experiments with the new architecture we obtain that the score with the testing data remains the same, having a more complex model without having improvements in performance is not sensible therefore we have chosen to keep this last simpler model as definitive model.

The score obtained with the testing data with the final model is: 40.58%

Tuning

The tuning of the hyperparameters was performed using Hyperband by Keras, the parameters and values on which the tests were performed are shown as follows:

- Num filters convolutional layer: min_value=5, max_value=50, step=5
- Kernel size convolution layer: min_value=1, max_value=10, step=2
- Activation function convolution layer: ['softmax', 'relu', 'sigmoid']
- Pool size pooling layer: min_value=1, max_value=5, step=2
- Dropout rate layer dropout: [0.1, 0.01, 0.0]
- Num units first layers dense: [100, 500, 1000]
- Activation function first layer dense: ['softmax', 'relu', 'sigmoid']

The values of the hyperparameters that have led us to have the best performance are the following:

- Num filters convolutional layer: 20
- Kernel size convolution layer: 1
- Activation function convolution layer: sigmoid
- Pool size pooling layer: 5
- Dropout rate layer dropout: 0.01
- Num units first layers dense: 100
- Activation function first layer dense: sigmoid

We did various tests also varying the time_steps value heuristically, finding that the best value is 10.

Score obtained after the tuning is: 42.33%.

Recurrent Neural Network

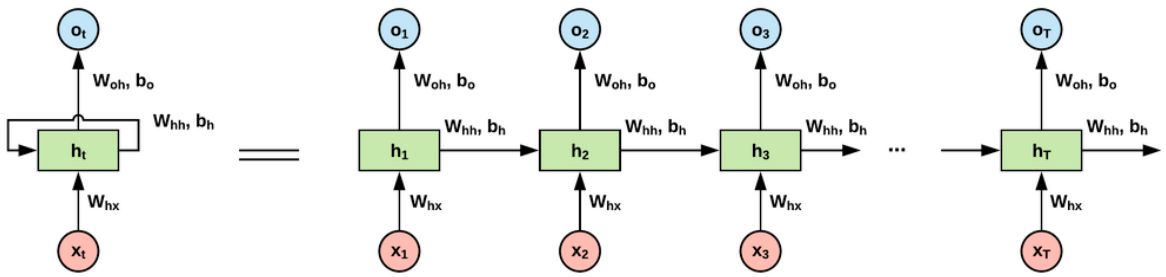
A recurrent neural network (RNN) is a type of artificial neural network that is designed to process sequential data or data with temporal dependencies. Unlike traditional feedforward neural networks, which process inputs in a fixed sequence, RNNs have loops within their network architecture, allowing information to persist and be passed from one step to the next.

This is particularly important in the case of emotions, as emotions are often influenced by temporal context and the relationships between past and present events, for this reason it's suitable for the task of our project.

The key feature of an RNN is its ability to capture and utilize the sequential information present in the input data.

At each step of the sequence, an RNN takes an input and produces an output, while also maintaining an internal hidden state. The hidden state serves as a memory that stores information about the past inputs seen by the network. This hidden state is updated at each step using the current input and the previous hidden state, creating a feedback loop that allows the network to capture long-term dependencies.

The basic building block of an RNN is called a recurrent neuron or a memory cell. The most common type of RNN cell is the long short-term memory (LSTM) cell, which includes mechanisms to selectively remember or forget information over time.



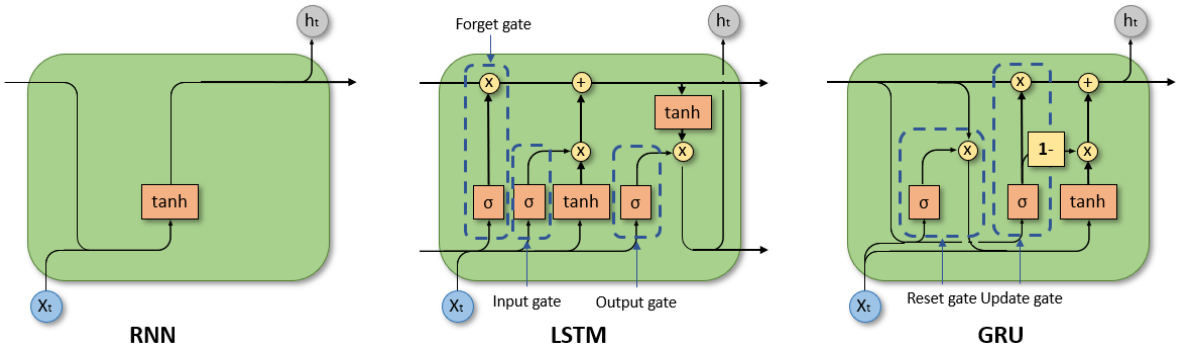
Architecture of LSTM

The architecture of a LSTM recurrent neural network (RNN) consists of three main components: an input layer, recurrent layers, and an output layer.

1. Input Layer: The input layer of an RNN receives the sequential data as input. Each element of the sequence is fed into the network at a specific time step.

2. Recurrent Layers: The recurrent layers are responsible for processing the sequential information and capturing temporal dependencies. These layers have recurrent connections, where the output of a neuron is fed back as an input to the same neuron or other neurons in the layer at the next time step. This feedback loop allows the network to maintain a memory of the past inputs and make decisions based on them.
   The Long Short-Term Memory (LSTM) RNN, which is the most common type of recurrent layer, includes memory cells and gating mechanisms. The memory cells can store and update information over time, while the gating mechanisms control the flow of information into and out of the cells. LSTMs are effective in capturing long-term dependencies in sequences.

3. Output Layer: The output layer of the RNN receives the final hidden state or output from the recurrent layers and produces the desired output. The specific design of the output layer depends on the task at hand.



Network Architecture

Our RNN network was structured as follows:

1. LSTM Layer with 64 units.
2. LSTM Layer with 64 units.
3. LSTM Layer with 64 units.
6. Dense layer with 100 units
7. Dense: a densely connected layer that performs the final emotion classification.

Early stopping

To prevent overfitting, we have used the Early Stopping technique.

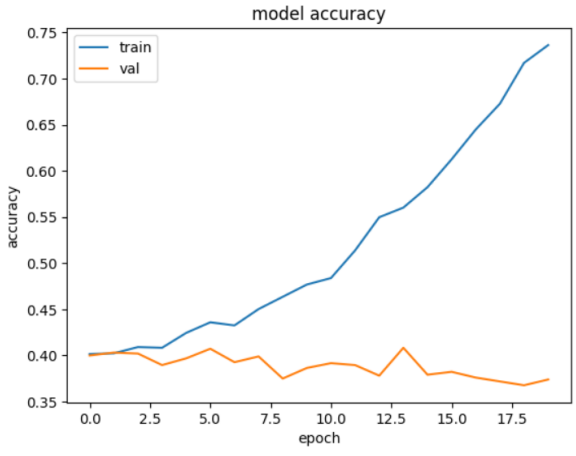
We compared a first case without Early stopping and a second case with Early stopping, using the EarlyStopping function from 'tensorflow.keras.callbacks' to achieve this object. We got the following results:

```
Epoch 19/20
135/135 [==============================] - 3s 21ms/step - loss: 0.0918 - accuracy: 0.6674 - val_loss: 0.1575 - val_accuracy: 0.4146
Epoch 20/20
135/135 [==============================] - 3s 21ms/step - loss: 0.0841 - accuracy: 0.6961 - val_loss: 0.1643 - val_accuracy: 0.4271
38/38 [==============================] - 2s 9ms/step - loss: 0.1720 - accuracy: 0.3892

[test loss, test accuracy]: [0.17202308773994446, 0.3891666531562805]
```

We can observe that the network obtains a very high accuracy during the learning phase, but in the testing phase the score drops rapidly.

This is definitely due to Overfitting, and we can demonstrate it with the graphs shown.

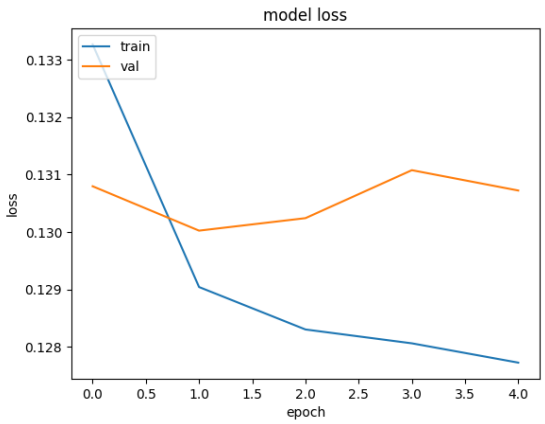


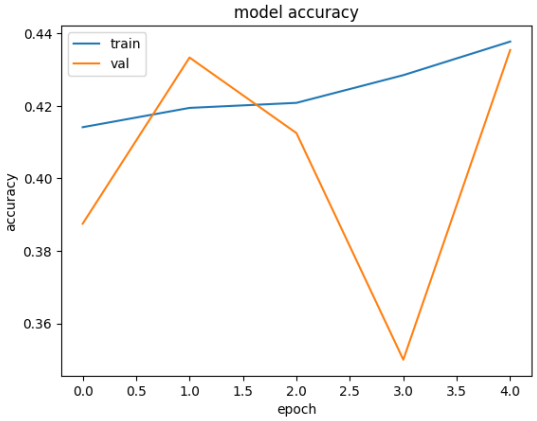
Therefore, we used the early stopping technique to prevent overfitting and we got the following results:

```
Epoch 4/20
135/135 [==============================] - 2s 17ms/step - loss: 0.1283 - accuracy: 0.4273
38/38 [==============================] - 1s 7ms/step - loss: 0.1325 - accuracy: 0.3967

[test loss, test accuracy]: [0.1324583888053894, 0.39666667580604553]
```

We observe that during the training phase the accuracy remains lower, but in the final test the score is better. This is due to the fact that the network does not go into overfitting and is therefore able to make correct predictions.





To better manage overfitting, an LSTM layer was also removed since the network was too large for the proposed dataset and therefore, removing a layer result in an improvement in the final score.

Then Our RNN network was structured as follows:

1. LSTM Layer with 64 units.
2. LSTM Layer with 64 units.
3. Dense layer with 100 units
4. Dense: a densely connected layer that performs the final emotion classification.

Tuning using HyperBand

We then performed hyperparameter tuning using the Keras HyperBand model. We decided to tune on the following hyperparameters and with the following values:

```
def model_builder(hp):
    model = tf.keras.Sequential()
    model.add(tf.keras.layers.LSTM(units=hp.Int('units_L1_LSTM', min_value=32, max_value=512, step=32),
                                   return_sequences=True, input_shape=(time_steps, input_dim)))
    model.add(tf.keras.layers.LSTM(units=hp.Int('units_L2_LSTM', min_value=32, max_value=512, step=32)))
    model.add(layers.Dropout(hp.Choice(name='Dropout_Rate', values=[0.0, 0.1, 0.01, 0.001])))
    model.add(tf.keras.layers.Dense(units=hp.Int('units_Dense', min_value=50, max_value=500, step=50)))
    model.add(tf.keras.layers.Dense(output_dim))

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

    return model
```

We did various tests also varying the time_steps value heuristically, finding that the best value is 10.

After searching for the best hyperparameters we obtained the following values:

```
Migliori iperparametri:
units_L1_LSTM: 160
units_L2_LSTM: 320
Dropout_Rate: 0.1
units_Dense: 400
```

We then used hyperparameters to train the model one last time and check the results obtained:

```
Epoch 6/20
135/135 [==============================] - 14s 106ms/step - loss: 0.1286 - accuracy: 0.4238
38/38 [==============================] - 3s 41ms/step - loss: 0.1313 - accuracy: 0.4075

[test loss, test accuracy]: [0.1312514841556549, 0.4074999988079071]
```

We have therefore achieved a better result, albeit slightly, with the optimization of the hyperparameters, with a final score of around 0.4.

FUZZY NETWORK

INTRODUCTION

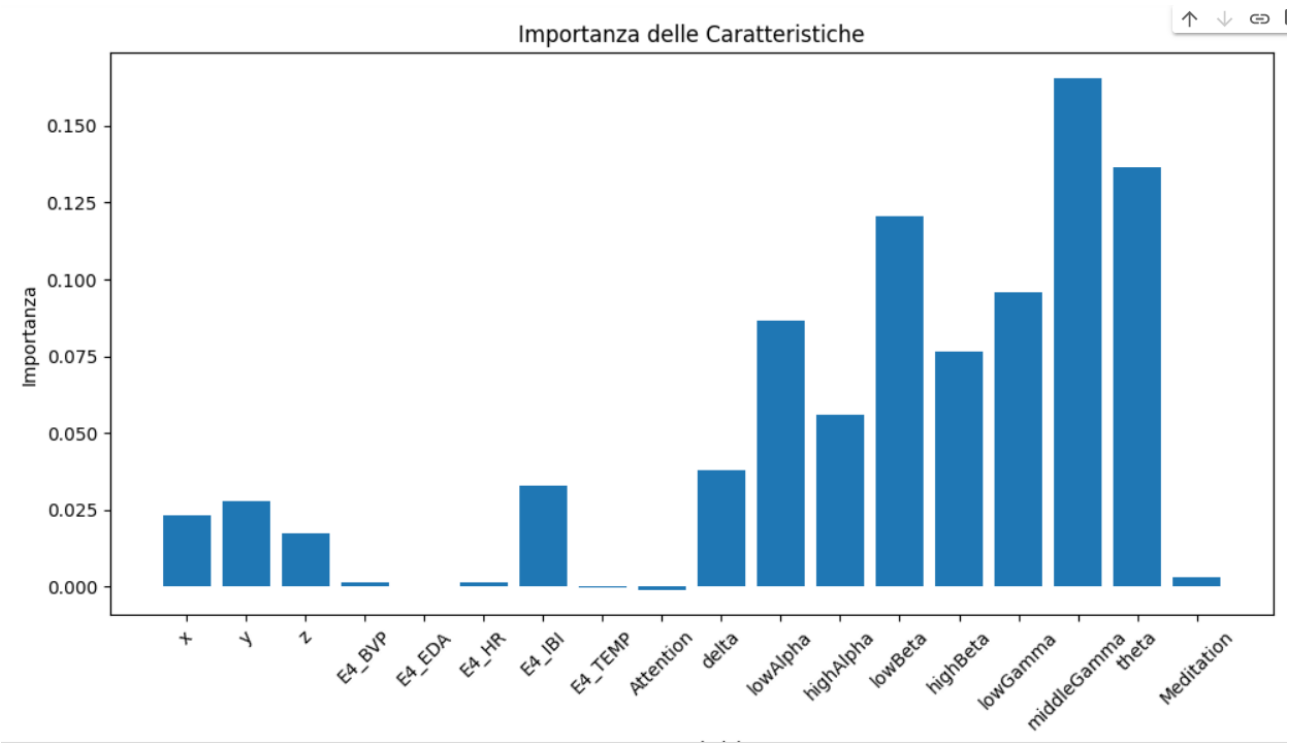
Emotions are complex and multi-dimensional phenomena that are challenging to represent using traditional crisp logic models. Fuzzy logic, with its ability to capture the uncertainty and fuzziness inherent in emotional states, offers a more flexible and natural approach to emotion recognition. By allowing emotions to be represented as fuzzy sets, we can account for the inherent ambiguity and subjectivity associated with emotional states.

Linguistic Representation of Emotions Fuzzy logic provides a convenient framework for representing emotions linguistically. Emotions can be described using linguistic variables such as "happy," "sad," "angry," etc., and associated membership functions can be defined to capture the degree of each emotion. This linguistic representation facilitates the mapping of raw input data, such as facial expressions or speech patterns, to emotional states.

Emotion recognition is inherently subject to uncertainty and ambiguity due to variations in individuals' expression of emotions. Fuzzy logic enables us to model and handle this uncertainty by assigning degrees of membership to multiple emotional states simultaneously. This allows for a more nuanced understanding of emotional responses and avoids the binary classification limitations of traditional approaches.

FEATURE IMPORTANCES

In order to generate fuzzy rules we decided to select only 4 parameters within our dataset. Using MLP we then analyzed the most important features in order to generate the fuzzy rules



Looking at the graphs, we used the 4 most impactful features within our system:

- highAlpha
- theta
- lowGamma
- delta

To implement the fuzzy network, we decided to implement the wang-mendel algorithm.
One significant advantage of the Wang-Mendel method over Mamdani is its ability to handle uncertainty more effectively; in fact it allows for modeling of higher levels of uncertainty and vagueness. We can also capture and represent uncertainties in

linguistic variables more accurately, leading to improved decision-making in complex and uncertain environments.

The rule explosion problem is a common issue in fuzzy logic systems, particularly with complex rule sets. The Mamdani method often requires many rules to represent the system's behavior accurately. In contrast, the Wang-Mendel method provides a more compact representation of rules by utilizing fuzzy clustering algorithms to generate a reduced rule base. This reduces the computational complexity of the system and enhances its efficiency.

```
45408 primitive rules
5774 non conflict and non duplicate rule
```

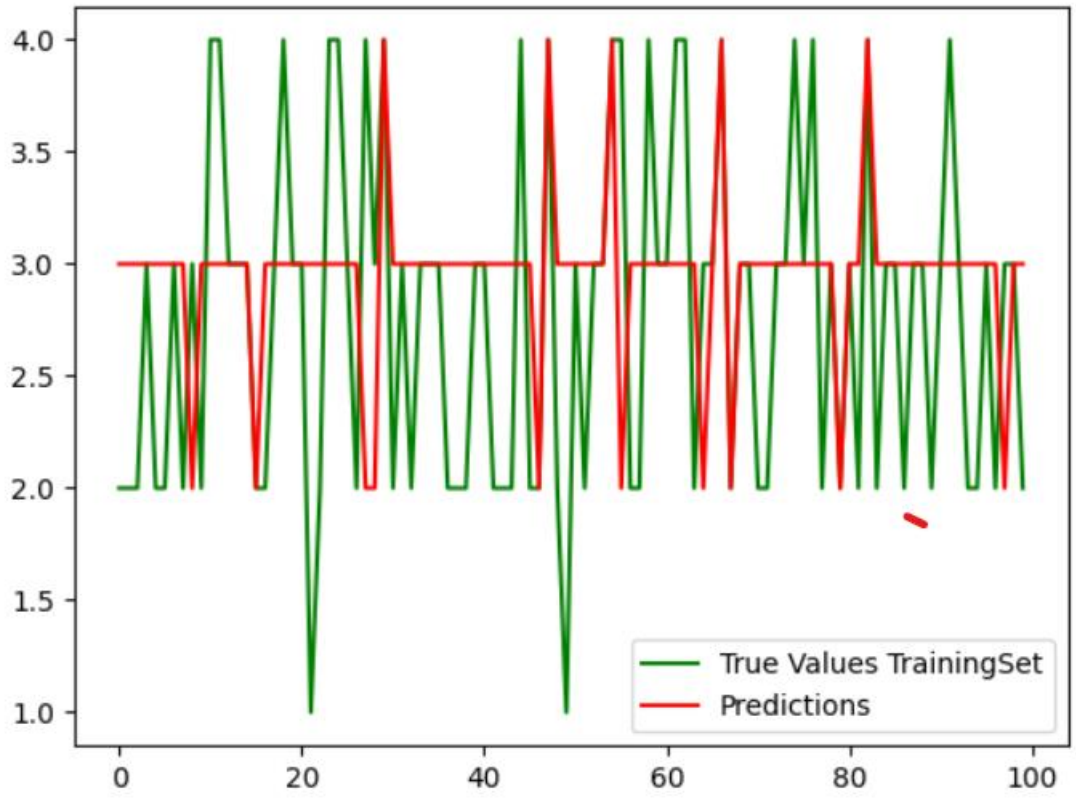
With our features Wang-Mendel algorithm generates 750 rules and only 362 of them are not in conflict or duplicate.

RESULTS

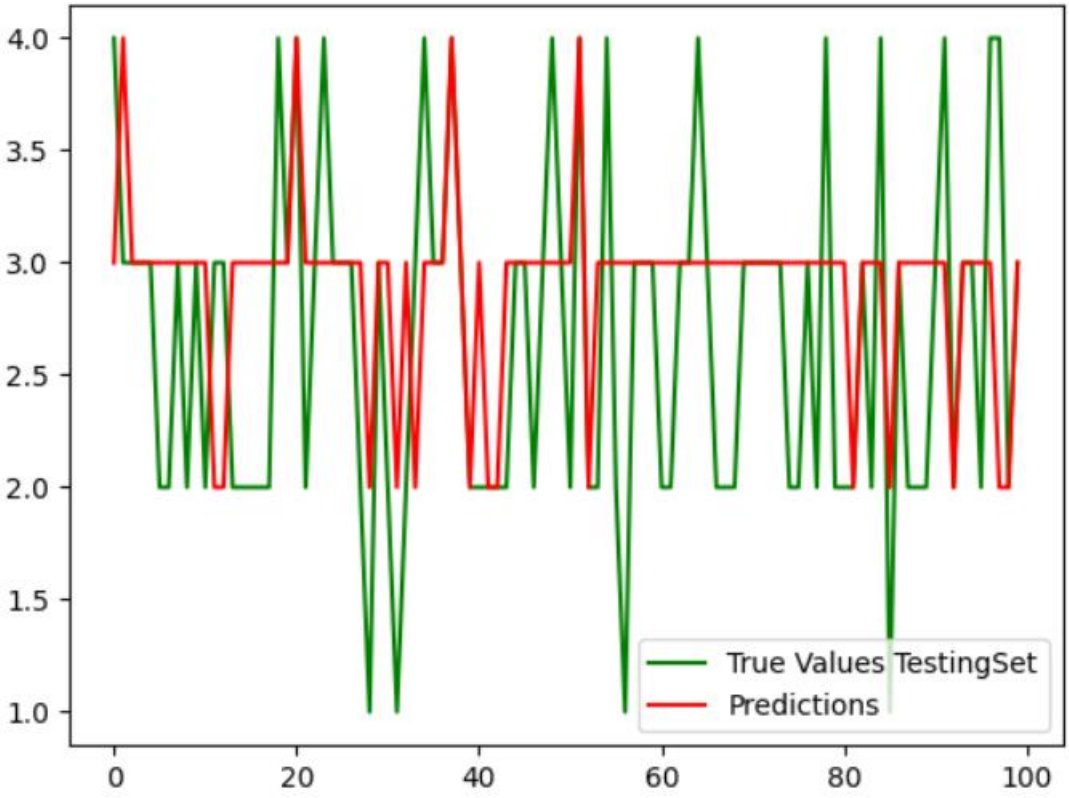
To verify the accuracy of the network, we plotted the values produced by the prediction with respect to the real values of the dataset

We analyzed two scenarios, dividing the dataset into training data and testing data:

- the first graph represents the prediction using the training data, so in this case both the rules and the prediction were made with the same data



- In the second graph we always used the training data for the generation of the rules while for the prediction testing data, thus analyzing the behavior of the rules created with different data



Accuracy results for both sets are the following:

```
Accuracy training test:
0.45
Accuracy testing test:
0.46
```