Facial expression recognition application using CNNs

Peagno Eleonora
1237035

Peron Giovanni
1237783

Rossi Daniel
1211017

Abstract

The facial expression recognition is a challenging task in machine learning field, and there is an active research on this topic. Develope a machine able to understand human emotions is a fascinating goal. The purpose of this report is to exploit all the topics studied during the Vision and Cognitive Services course and others like Machine Learning and Deep Learning in order to implement a system able to predict human emotions. We will describe how we realized this system specifying all the steps performed, from the first CNN we tried to the final model we obtained. We will illustrate all the procedure we used trying to achieve better results. Our target was not very high since this is a challenging task as we said, however with our final model we arrived to a predition accuracy of ***71.52*** *that is better than human accuracy. Finally using this model we reached our goals implementing a nice application.*

1. Introduction

Facial expressions recognition (FER) is an interesting and challenging problem in machine learning field. It is also a task that can be applied in many important applications. Facial expressions have an important role in every human interaction so developing a machine able to recognize and understand human expressions automatically can be very useful in many existing and novel fields [5].
One of these fields is behaviomedics, where systems able to exploit automatical analysis of affective and social signals to aid diagnosis, monitoring and treating medical condition that alter behavior. Facial expression recognition can be also use in data analytics field, for example to understand emotions of people that are looking at ads or political debate and make statistics related to people's preferences. Another application field for Facial expression recognition is human-computer interaction. Understanding human emotions would make the attitude of systems like vocal assistants or robots much closer to the way that humans interact with each other. Recognizing expressions could also be useful to improve the identification of micro facial expressions which can be used in deceit detection applications. Due to all these possible applications, facial expressions recognition is widely studied also because recognizing human expressions in natural condition environment is a very challenging task. With this project we aim to build a facial expressions classifier able to reach and overtake human accuracy on this task. The main idea was focusing mostly on study different types of model in order to understand a good way to achieve valid outcomes. For this reason all preprocessing techniques that can be applied to the input data for improving classification results were not be consider.

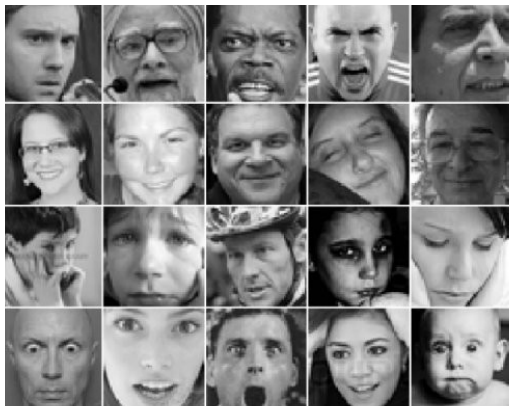


Figure 1. Example of images from FER2013 dataset

The chosen dataset is the FER2013 it was selected after some researches, above some examples of images of this dataset are reported. FER2013 was the desired dataset and the most suited for our purpose. Training a model with that dataset was the challenge with the right level of difficulty we was searching for. Moreover FER2013 provides a very large set of examples well differentiated in terms of subject age, face pose, illumination and other realistic conditions. So using this dataset our goal was to realize an ensemble model using many types of Convolutional Neural Networks (CNNs) able to achieve a test accuracy greater than 65.5%, that is the human accuracy measured on FER2013 dataset [1]. Our target was reached using the final model described in the next sections, which achieve test accuracy of 71.52% on FER2013.

2. Related Work

The first step for reaching our goals it has been a research of all the scientific papers related to the facial expression recognition. We found many different works related to the topic all reviewed in *"Deep Facial Expression Recognition: A Survey"* by S. Li and W. Deng [4], thank to this paper we could find a lot of correlated works about the same kind of classification we were looking for. We searched for all the works reported that were using the FER2013 dataset, trying to find a way to realize a model aligned with the best results achieved in the last years. We found six different papers related to the FER2013 dataset, they were classified by accuracy reached and type of neural networks used. After inspecting all these works we chose to follow the paper that achieved the best accuracy, moreover it used an ensemble method and we were interested in understanding better this approach. So we decided to follow the methods used in *"Facial Expression Recognition using Convolutional Neural Networks: State of the Art"* by C. Pramerdorfer and M. Kampel [6], that consists on an ensemble method made up with eight different CNNs of three different types, the accuracy reported for this method was 75.2%. This paper has been enticing for us, because in it are explained the power and the bottlenecks of the CNNs models, moreover the final part of that paper was destined to explain how is possible to overcome the problems of CNNs. We found also other papers that we used as support to realize many different types of CNNs:

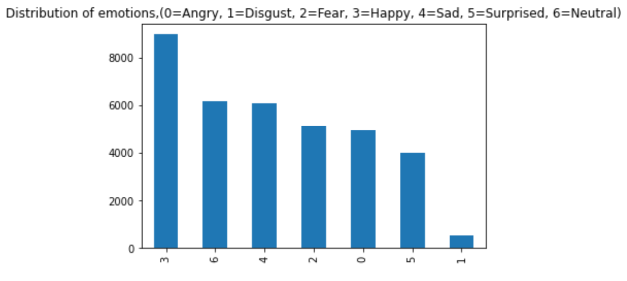
- "Very Deep Convolutional Networks for Large-Scale Image Recognition" by K. Simonyan and A. Zisserman [7];
- "Going Deeper with Convolutions" by C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich [8];
- "Deep Residual Learning for Image Recognition" by K. He, X. Zhang, S. Ren, and J. Sun [2].

Following we report a table that summarizes three of the works, executing facial expressions recognition task on the FER2013 dataset, we considered from [4].

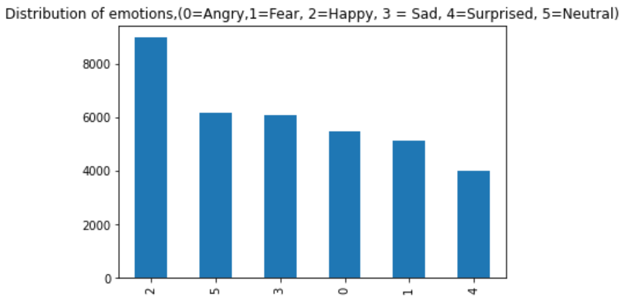
Papers	Network type	Accuracy reached
Zhang et al. [9]	CNN Multitask Network	Test: 75.10
Kim et al. [3]	CNN Network Ensemble	Test: 73.73
Pramerdorfer et al. [6]	CNN Network Ensemble	Test: 75.2

3. Dataset

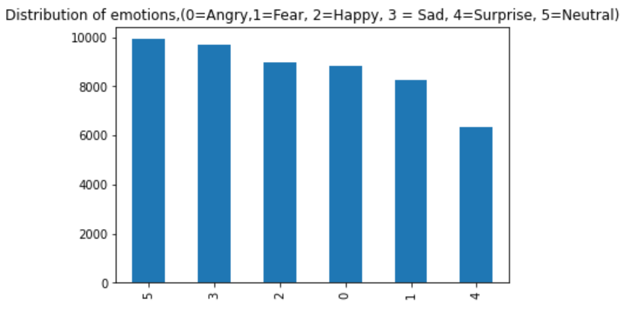
The FER2013 database was created by Pierre Luc Carrier and Aaron Courville. It was introduced during the ICML 2013 Challenges in Representation Learning and it is a large-scale and unconstrained database. This dataset was built collecting images in an automatic way using the Google image search API. In order to find useful faces images a research has been carried out combining a set of emotion related keywords with other words associated to gender, age or ethnicity. In this way about 600 strings were obtained and they were used to query the search API. Then all the images collected with this system have been cropped and resized to 48*48 pixels and they have been also converted to grayscale. We choose this dataset for many reasons, for example we find it is cited in many papers and we consider it well formed dataset for the reason that it contains images with different illumination, subjects with different age, pose and expression intensity, moreover some images have also occlusions. In general the images contained in the FER2013 dataset represent a good sampling under realistic conditions. The most important reason that pushed us to chose it is the huge number of images that it provides. Precisely it contains 28709 training images, 3589 validation images and 3589 test images, so in total 35887 images. For each image there is an associated class that represent seven different expression labels: Angry, Disgust, Fear, Happy, Sad, Surprised, Neutral. Below we report an histogram showing the distribution of all the FER2013's labels.



We tried to apply as few preprocessing techniques as possible because we want to focus on the model performance providing to it a dataset not too optimized. We do this choice in order to emphasize the power of the model built, in the sense we want a model able to reach interesting performance independently the goodness of the dataset provided to it. For this reasons we operate only some minimal preprocessing modification over the dataset, in order to prepare the data to be learned. First of all we notice that the class disgust is not so big with respect to the others, so we decide to merge it with the angry class. Following a graph showing the distribution with merged classes.



After this operation the classes are not so much leveled, in particular compared to happy class. In fact we have a lot of images for the happy class instead the ones for the surprised emotion are very few.To solve this issue we decide to replicate the 60% of the images of each class, excluding the happy class because as we said it already has enough images. In this way we make the images amount for each class more balanced and we reach the following distribution with a total amount of 52025 images.



Finally we performed a normalization operation over the dataset in order to provide to the model a numeric common scale. After all these needed preprocessing steps we obtain a training set formed by 44847 images and a validation set and test set composed both by 3589 images.
An evaluation of other dataset has been done, we considered also CK+ (CohnKanade) dataset that is also commonly used for evaluating FER systems. The first point that brought us to discard CK+ were the huge amount of data provided by FER2013, we decided that have more images as possible was the right choice in order to develop a model that could well generalize, moreover the aim of reaching an interesting accuracy with FER2013 was more challenging. We decided to prefer FER2013 also because it appeared to us more easier to work with, in that FER2013 was provided in a csv file. A more important reason that led us to prefer FER2013 is the fact this last provides specified training, validation and test sets, instead CK+ does not, so in order to compare our results with other works it was the most appropriate dataset.

4. Method

The final model we build is an ensemble composed by different models from VGG, inception and ResNet types, for each type of model we train different of it in order to obtain predictions well generalized, the best models we develeped is reported in table 4.3.

4.1. VGG

For realizing this kind of network we follow the structure VGG-B descripted in [7]. The VGG model is composed by several [CCPD] blocks and after flatting the data thery are followed by some Fully connected (F) layers. We developed 3 different version of this type of network, as shown below:
VGG_v0 is composed by 4[CCPD], followed by 3F layers with respectively 25,25,6 neurons.
VGG_v1 is composed by 4[CCBPD], followed by 4F layers with respectively 512,256,128,6 neurons.
VGG_v2 is composed by 4[CBCBPD], followed by 4F layers with respectively 512,256,128,6 neurons. VGG1 and VGG2 regularize the first C layer with L2 norm $\lambda = 0.01$.

4.2. Inception

Inception model is a type of network developed from Googlenet [8] with some modification according to paper [6], we remove the initial pooling layer because, how the paper report [6], images are not so big. The network consist of several Inception (I) blocks followed by a flatten and dropout layer and ending in a F layer with 6 neurons. For creatinig the I block we used a function that took a parameter n representing the number of neurons of each block. We start with 64 neurons and we increment it of 32 after each I block. Every I block use one C layer padding same and L2 norm $\lambda = 0.0002$, also we report kernel size (K) for the C layers. I block is composed as follow:

1. 1C relu, $3/4 \cdot n$, KernelSize (K) = (1,1)
2. 2C relu, $1/2 \cdot n$, K=[(1,1),(3,3)]
3. 2C relu, $1/8 \cdot n$, K=[(1,1), (5,5)]
4. 1C relu, $1/4 \cdot n$, K=(5,5)
5. PC relu, $1/4 \cdot n$, K=(1,1)

All these blocks are concatenated in a [1C,2C,2C,PC] layer and after a P layer it become the input of next I blocks.

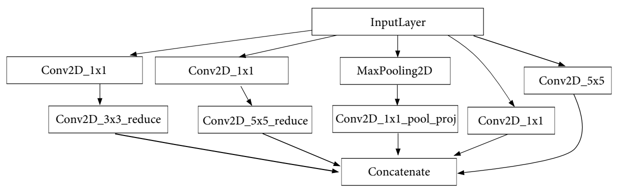


Figure 2. Representation of an I block

4.3. ResNet

Resnet model is a network that use Residuals (R) blocks. Each R consists of BatchNormalization (B) layer follow by CFBC layers with strides=1, in the end we find an add (A) layer where we concatenate the output of CFBC and the initial B. Each C layer has kernel size = (3,3) and use padding same, moreover the number of filter increase by a factor of 2 at each R block.

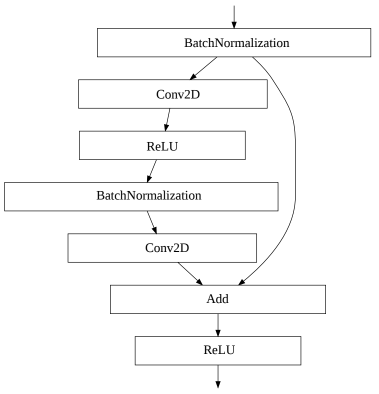


Figure 3. Representation of a residual block

At some fixed number of R blocks it is necessary to downsample the data and we substitute, into the A layer, the initial B with another C layer with strides=2, also the first C layer of CFBC layers has stride=2.

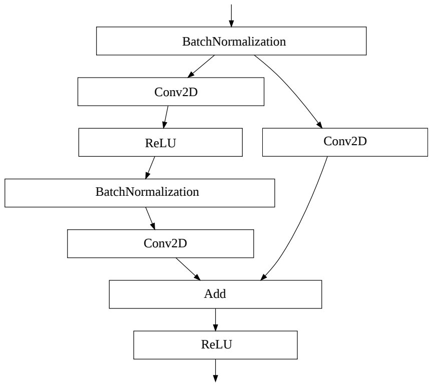


Figure 4. Representation of a residual block with downsample

4.4. Optimization

In this part we exploit how we train the models and what tools we use in order to get best model. Each model was training with 150 epochs using Adam optimizer. Train this models were very long so we decided to use early stopping with a patience of 10 epochs and in some models we use a regularizer to help the loss function when it find plateaus by reducing the learning rate if there is no improvement on the loss function of the validation set after 10 epochs.
We use checkpoints to save best weights during training.
After the training phase we obtain about 11 models and we operate a selection over them, that we explain in experiments section 5. We find out that the best combination of models consist of 9 models, we have 5 VGG, 1 inception and 3 ResNet models. We combine them and create two different ensemble, the first one make the average and the second the weighted average based on accuracy of the model to each class.

Name	Type	Architecture	Depth	Params
VGG0	VGG_v0	4[CCP] FF	10	4.8m
VGG1	VGG_v1	4[CCBP] FFF	11	5.9m
VGG2	VGG_v1	4[CCBP] FFF	11	5.9m
VGG3	VGG_v1	4[CCBP] FFF	11	5.9m
VGG4	VGG_v2	CCBP 3[CBCBP] FFF	11	5.9m
Inc0	Inception	CIPIIPIIPIIPF	16	4.8m
ResNet2	ResNet	C 2[R]5[R]5[R]2[R]PF	30	15.6m
ResNet3	ResNet	C 3[R]4[R]6[R]3[R]PF	34	21.3m
ResNet4	ResNet	C 3[R]4[R]6[R]3[R]PF	34	21.3m

Table 1. Summarize all the models realized. C, P, B, I, and F stands for convolutional, pooling, batch normalization, inception and fully connected layers respectively. 3[R] means group of three residual blocks. 4[CCP] means group of four CCP blocks. Final layer and dropout have been omitted.

4.5. Ensemble method

The main goal was to train different model and then combine them in order to obtain a better generalization and improve the performance of the Ensemble, as follow the Ensemble we tried:

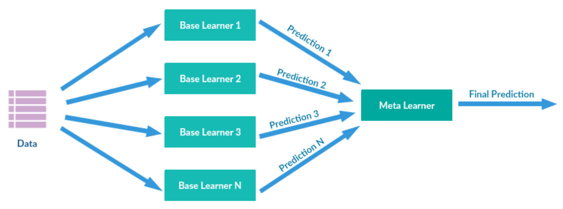


Figure 5. Example of general ensemble method, in the meta-learner we compute the weighted average using the accuracy over the class and global accuracy for each model.

Name	0 (%)	1 (%)	2 (%)	3 (%)	4 (%)	5 (%)	Test set (%)
ensemble9_1	66.53	42.33	84.69	61.40	83.37	70.01	71.41
ensemble9_2	66.53	42.13	84.69	61.56	83.37	70.51	71.41
ensemble9_3	66.15	41.93	84.46	61.40	83.37	70.34	71.52

Table 2. 0=Angry,1=Fear,2=Happy,3=Sad,4=Surprised,5=Neutral, accuracy over the classes for each of the ensemble method.

Ensemble9_1: we take the prediction from all models and then we calculate the simple average over the prediction of the same class, the best ensemble we obtain has accuracy 71.41% over the test set.
Ensemble9_2: a natural step was to try weighted the average based on the accuracy of each model, but the final accuracy of the model is the same as the previous, the accuracy is 71.41% over the test set.
Ensemble9_3: we saw that some model in specific class are not so good, so we decide to weighted the average with class accuracy instead global accuracy of each model, the accuracy is 71.52% over the test set.

Name	0 (%)	1 (%)	2 (%)	3 (%)	4 (%)	5 (%)	Test set (%)
VGG0	53.53	49.19	76.87	51.60	81.92	57.00	63.58
VGG1	55.64	39.31	80.00	59.26	82.65	62.60	64.45
VGG2	60.80	31.85	80.33	61.86	76.86	61.12	65.78
VGG3	65.77	36.08	82.01	42.72	81.20	76.93	66.06
VGG4	60.03	40.72	81.56	47.93	79.03	72.98	67.18
Inc0	58.69	41.73	78.88	56.20	75.90	55.51	63.61
ResNet2	64.62	34.47	79.88	39.05	77.83	69.85	61.80
ResNet3	56.78	41.53	80.89	49.92	82.89	61.94	63.17
ResNet4	63.28	39.91	83.35	51.30	75.90	50.08	63.53

Table 3. 0=Angry,1=Fear,2=Happy,3=Sad,4=Surprised,5=Neutral, accuracy over the classes for each of the 9 models selected for the ensemble method.

5. Experiments

In this section we will explain the process we made in order to realize the final ensemble model, illustrating all the steps we perform to build the needed networks and to improve the performance of them.

5.1. Starting point

Our experiments started from an unique convolutional neural network with a VGG like structure, which we called old_net. This first network was trained with Adam optimizer and it reached a test accuracy of 60.27%. We started from this first network and we searched for best methods in order to increase model's accuracy. Like we said in the previous sections the final model we obtained is an ensemble method composed by nine CNNs of three different types, that are VGG, Inception and ResNet. And in this nine networks used for the final ensemble method we had to discard some networks we realized during the experiments process pursued in order to achieve models with best performance. In 4 we show two models we discarded.

Name	0 (%)	1 (%)	2 (%)	3 (%)	4 (%)	5 (%)	Test set (%)
old_net	57.93	37.70	75.08	46.70	79.27	54.20	60.27
ResNet0	57.36	45.16	80.44	47.62	75.42	58.31	61.80

Table 4. 0=Angry,1=Fear,2=Happy,3=Sad,4=Surprised,5=Neutral, the models we rejected because get worse the ensemble.

5.2. Realizing three CNNs types

After some modification tests, which did not lead so much improvement to our first network old_net, we started following the model described in [6], implementing three CNNs architecture described in 4.1, 4.2, 4.3.
The initial modification has been the addition of B layers after every C layer. We also added some D layers, which have been useful to control the overfitting problem. Then we tried to use an SGD optimizer with initial learning rate = 0.1, L2 norm $\lambda = 0.0001$, momentum = 0.9, batch size = 128. Initially we trained every network for 300 epochs but then we decided to reduce the number of epochs for which we trained the models to 150, for reasons of time needed for the training process.

5.3. Modifications for performance improving

Following we describe al the modifications we apported to each model we realized. We added a learning rate regularizer in order to half the learning rate if the validation accuracy did not improve for 10 epochs and with this optimizer we did not reach very interesting results due to the fact that sometimes the validation accuracy remained stuck on too low value, despite the learning rate regularizer.
For this reason we switched to use Adam optimizer, which being an adaptive learning method solved our problem. Using the tensorflow callback ReduceLROnPlateau[1] we added another system to control the learning rate.
With this method we help the loss function to get rid when the gradient descent is stuck in plateaus regions. What is done is reducing the learning rate parameter multiplying it for a defined factor if there is no improvement on the validation loss after a certain amount of epochs specified with the patience parameter. In order to save time and to optimize the accuracy of our models we decided to use the early stopping technique, trying different patience setting, in this way we were stopping the training when the model were not able to improve his validation loss for 10 epochs. We also use a checkpointer callback[2] in order to save the best model reached during the training process. In addition to the dropout we tried to use also an L2 regularizer in the VGG type networks. Combining all this settings we managed to get improved results.
Practically trying many different combination of modifications including the change of filters used in convolution layers and the addition of some more dense layer before the output layer. Like we reported on the table3 we achieved a 67.18% of test accuracy with a VGG neural network, that is our best result for a single network model.

[1]`https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/ReduceLROnPlateau`
[2]`https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/ModelCheckpointin`

So, after many experiments we obtained eleven different models and we selected nine of them shown in table 3.

5.4. Ensemble models

Initially we used all the eleven models to build the ensemble model, then like is illustrated in 4.5
The ensemble model was built using a bagging technique, so retrieving the predictions made from all the single models and combining in a unique final prediction.

Name	0 (%)	1 (%)	2 (%)	3 (%)	4 (%)	5 (%)	Test set (%)
ensemble11_1	66.73	43.34	84.58	59.72	82.40	70.51	70.95
ensemble11_2	66.53	43.75	84.69	59.57	82.16	70.67	70.39
ensemble11_3	66.92	43.75	84.35	60.03	82.40	71.33	70.95

Table 5. 0=Angry,1=Fear,2=Happy,3=Sad,4=Surprised,5=Neutral, in this ensemble we use all the 11 models developed.

ensemble11_1: the predictions are combined making the average over all the predictions of the same class of each eleven models. In this way, that is putting together all the models we obtained a test accuracy of 70.95%.
ensemble11_2: At this point we tried to improve out performance weighting the results of each models based on accuracy of each model over the validation set, and we reach a test accuracy of 70.39%, it is worst compared to the previous.
ensemble11_3: we tried to weighting the predictions of each models with the accuracy over the classes for each model and we reach an accuracy of 70.95%.

In order to compute this average we calculated a vector of weights to apply on the predictions of every model. To compute the weights we calculated the recall on the validation set for every prediction value returned by every model and we normalize it dividing by the sum of the weights computed. The computation of the weights can be represented as following

$$\sum_{m=1,\ldots,n} \frac{r_m}{\sum_{i=1,\ldots,6} r_m[i]}$$

where r_m is a vector containing the recalls value for the model m Having the weights normalized means that we have a level of the importance for every predicted class of every models. The weighted average has been calculated as follows

$$\sum_{m=0,\ldots,n} y_m \cdot w_m$$

where in y_m we have the predictions of the model m and in w_m we have the computed weights for the model m.

5.5. Models selection

After create the first version of the ensemble we decided to try all different combination of the models from a minimum subset of 6 models to the maximum set of 11 models, we calculated for each combination the accuracy of the ensemble3 of the selected models, we chose to use ensemble3 because it seemed the better one.
So the final model we obtained is the ensemble9_3 model described in 4.5, with which we obtain 71.52% test accuracy.

5.6. Application

Finally we produced a python application for trying to use our model in order to predict emotions using the webcam of our personal computer. We realized a simply application that retrieve images from webcam, it crops them and it convert them to grayscale in order to make a little bit similar to the FER2013 images. The application starts the webcam and use our final model to predict an emotion and it shows the prediction below the webcam video that is being captured. Below we reported some application screenshot example.

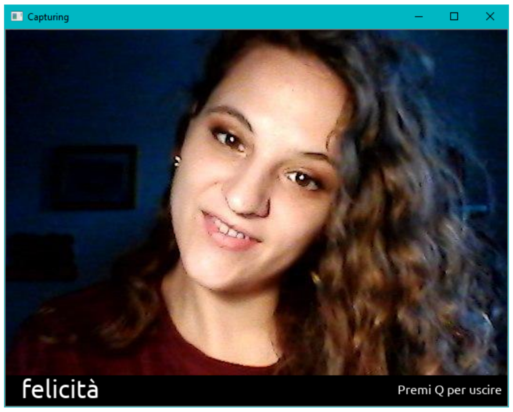


Figure 6. Example of happyness prediction

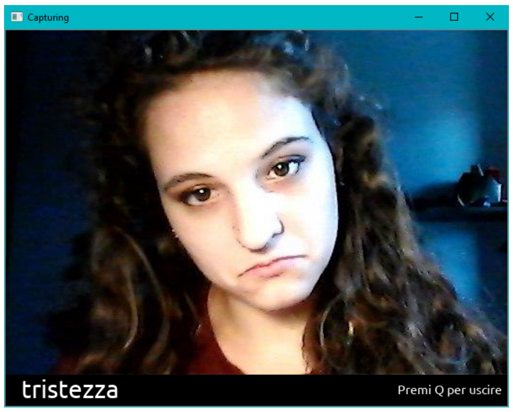


Figure 7. Example of sadness prediction

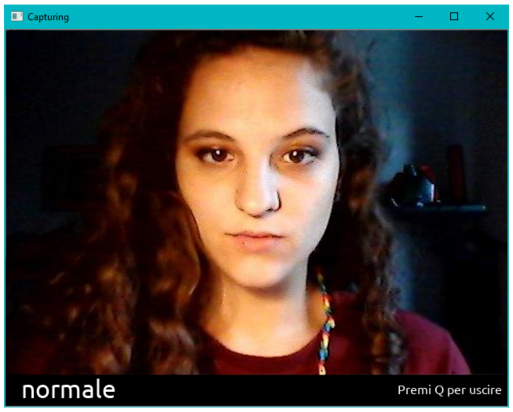


Figure 8. Example of neutral prediction

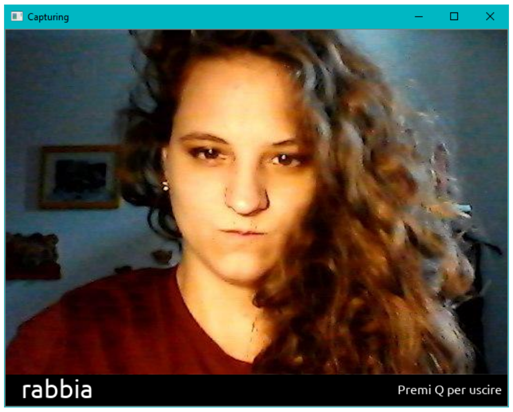


Figure 9. Example of angry prediction

6. Conclusion

We can say we have achieved our aims which were to overtake the human accuracy estimated on the FER2013 dataset, which is 65.5%. We wanted to reach a better performance than that and we our final model we registered a test accuracy of 71.52%, for this result we are satisfied. Futhermore we overtakes the accuracy reached from three papers, cited in [4], in which were described models that perform the same task with the same dataset used by us, we are fullfilled for this result too. We can say it was very difficult to develop all the different models and it tooks a lot of times to train them, but we achieved a very good performance for each model. Moreover the ensemble obtained was a challenge in general, because combining all different models were not too easy. One important point we could absorb from this project is that we learnt how to make appropriate modification in the models, in order to obtain improvements of performance. We also understood how it is possible to use many networks in order to produce an ensemble method and use it directly into an application. Surely we have explored only a part of the process to improve the performance on a model that predicts facial expressions, in fact the accuracy results obtained nowadays are greater than our. So there would be other techniques to apply in order to improve our model starting from data-augmentation to modifications of the ensemble model adding more types of CNNs and also improving the outcomes obtained from the ones we already use.

References

[1] Ian J. Goodfellow, Dumitru Erhan, Pierre Luc Carrier, Aaron Courville, Mehdi Mirza, Ben Hamner, Will Cukierski, Yichuan Tang, David Thaler, Dong-Hyun Lee, Yingbo Zhou, Chetan Ramaiah, Fangxiang Feng, Ruifan Li, Xiaojie Wang, Dimitris Athanasakis, John Shawe-Taylor, Maxim Milakov, John Park, Radu Ionescu, Marius Popescu, Cristian Grozea, James Bergstra, Jingjing Xie, Lukasz Romaszko, Bing Xu, Zhang Chuang, and Yoshua Bengio. Challenges in representation learning: A report on three machine learning contests. 2013.

[2] K. He, X. Zhang, haoqing Ren, , and J. Sun. Deep residual learning for image recognition. 2015.

[3] B.-K. Kim, S.-Y. Dong, J. Roh G. Kim, and S.-Y. Lee. Fusing aligned and non-aligned face information for automatic affect recognition in the wild: A deep learning approach. 2016.

[4] S. Li and W. Deng. Deep facial expression recognition: A survey. *IEEE Transactions on Affective Computing*, 2020.

[5] M. V. B. Martinez. Advances, challenges, and opportunities in automatic facial expression recognition. *Advances in Face Detection and Facial Image Analysis*, 2016.

[6] C. Pramerdorfer and M. Kampel. Facial expression recognition using convolutional neural networks: State of the art. 2016.

[7] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. 2014.

[8] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, , and A. Rabinovich. Going deeper with convolutions. 2015.

[9] C. C. Loy Z. Zhang, P. Luo and X. Tang. Learning social relation traits from face images. 2015.