CAP 6655 – Project 3

Facial Emotion Recognition model

Matteo Bassani

Introduction

Facial emotion recognition is the task of classifying the expressions on face images into various categories such as anger, fear, surprise, sadness, happiness and so on. Typically, emotion detection is implemented as a three-stage process:

1. Locating faces in the scene (face detection).
2. Extracting facial features from the detected face region (facial feature extraction).
3. Analyzing the motion of facial features and/or the changes in the appearance of facial features and classifying this information into some emotion categories like happiness or anger, fear, disgust, surprise etc. (facial expression interpretation).

Face expression recognition is one of the most powerful and challenging tasks in social communication. Generally, face expressions are natural and direct means for human beings to communicate their emotions and intentions. This task can find some applications in business for example, since can provide valuable information about the sentiment of a target audience towards a marketing message, product or brand.

Description

In this final project, my goal is to build a facial emotion recognition algorithm using images of people's faces (taken from public datasets found online). In particular, the algorithm should surround each face with a bounding box and label it with the age (or a range of ages) predicted and with the name of the emotion detected (i.e. happy, sad, angry etc). Also, it should work both for statical images and videos.

Faces are detected using the *face detection model of the DNN module of OpenCV*. In the first part of the project, we demonstrated that it works better than Haar cascade and is significantly less resource consuming than MTCNN. Age ranges are estimated using a CNN model developed from scratch in the second part of my project.

Once obtained the bounding box coordinates of the faces, they are extracted ignoring the rest of the image/frame. Doing so allows the facial emotion detector to focus solely on the person's face and not any other irrelevant "noise" in the image. The face ROI is then passed through the model, yielding the actual emotion prediction.

In particular in this project, we're going to compare the performance of a pre-trained model available in the Python libraries to the one of a model build and trained from scratch.

Models

- **Pretrained model**

Initially we will use a pre-trained Python model for Facial Emotion Recognition. If the performance is acceptable, we will test this model trying to analyze its pros and cons. Otherwise, we will try to train a new model from scratch, hoping to get better result than the pretrained one.

The model is called *Facial Expression Recognition Library* and is developed by Justin Shenk. This Library requires OpenCV>=3.2 and Tensorflow>=1.7.0 dependencies installed in the system. Faces are detected using OpenCV's Haar Cascade classifier or the more accurate MTCNN (that we discussed in Project 1). The CNN model used to recognize facial expression is presented in the paper *Real-Time Convolutional Neural Networks for Emotions and Gender Classification* (https://arxiv.org/pdf/1710.07557.pdf) written by O. Arriaga, P. G. Ploger and M. Valdenegro.

Dataset:

This model is trained on the popular FER-2013 emotion dataset. This dataset consists of 48x48 pixel grayscale images of faces labeled with a single emotion. The faces have been automatically registered so that the face is more or less centered and occupies about the same amount of space in each image. The total number of categories are seven (0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral). The training set consists of 28,709 examples and the public test set consists of 3,589 examples. The training set is divided in the following way:

- 7215 happy images
- 4830 sad images
- 4097 fear images
- 3171 surprise images
- 4965 neutral images
- 3995 angry images
- 436 disgust images

As we can see, the dataset is quite unbalanced, having too many happy images and too few disgust images. The other classes are quite balanced, with a little overpopulation of sad images and underpopulation of surprise one. We'll remember this and try to see if the performance of this model reflects the unbalancing.

Architecture:

The architecture of this model is a fully-convolutional neural network that contains 4 residual depth-wise separable convolutions where each convolution is followed by a batch normalization layer and a ReLU activation function. The last layer applies a global average pooling and a soft-max activation function to produce a prediction. This architecture has approximately 60000 parameters.

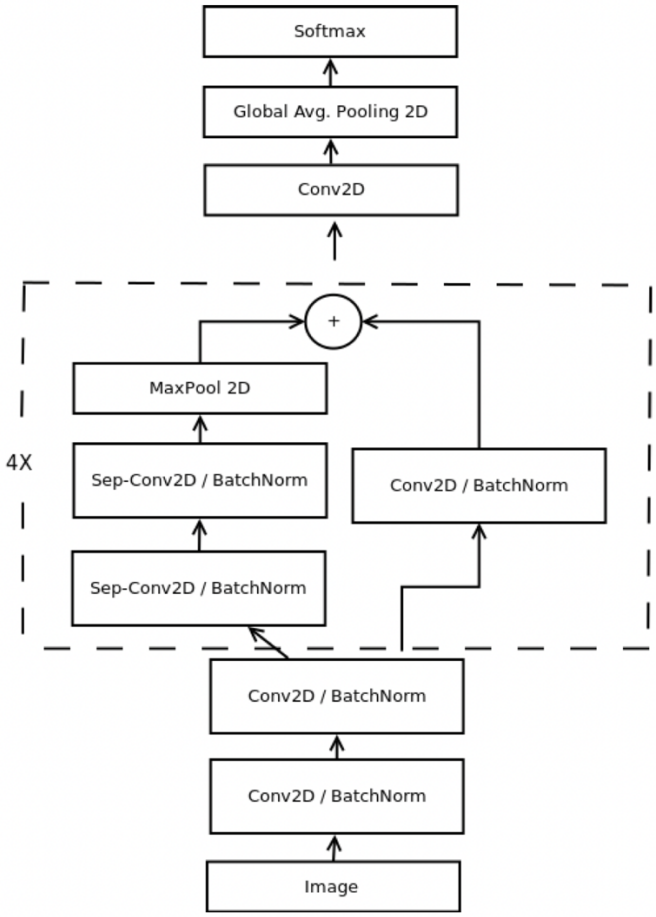


Figure 1: The architecture of this pre-trained model.

Results:

To test this model, we will adopt a basic video of me trying to mimic some basic facial expression. The reason of this choice is to avoid any possible image right issue. Here are some frames taken from this sample video:

Figure 2: Here are some frames taken from the sample video, with the emotion in the left-upper corner.

Here are some results of the pretrained model:

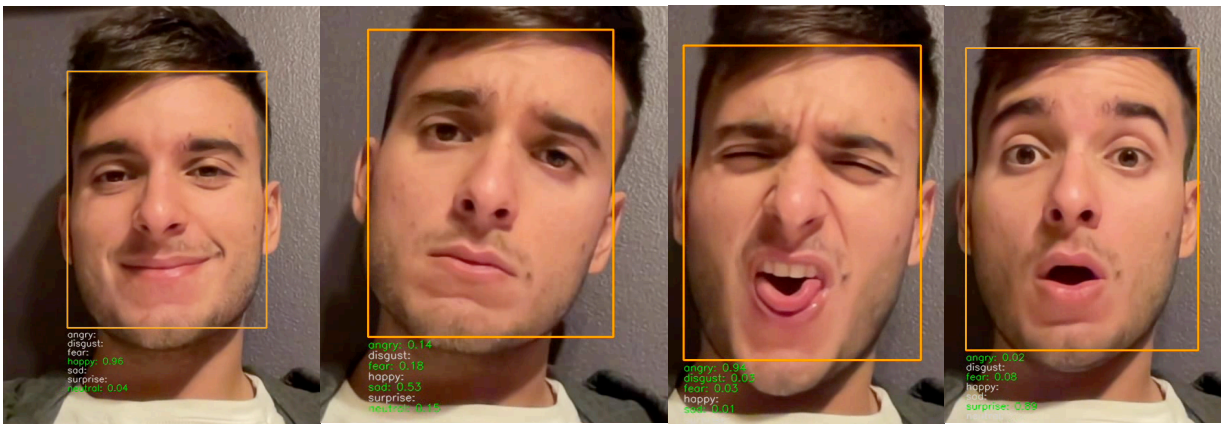
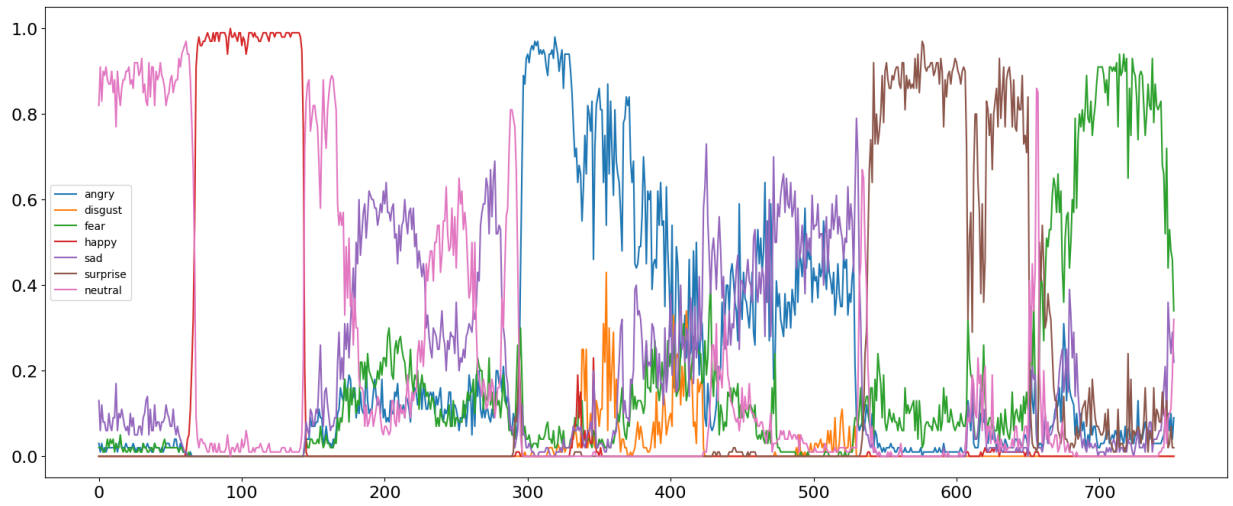


Figure 3: Some results of the pretrained model.

To be more precise in reporting the performance of this model, we can plot the confidence values of the different emotions during all the video. (The ground truth sequence is: neutral, happy, sad, disgust, angry, surprise, fear).



As we can see, the performance of the pretrained model is acceptable/good for: neutral face, happy face, surprise face and fear face.

However, the model has some problems recognizing sad faces, and angry ones. Angry, sad and neutral are often mixed or confused. But the major problems came with disgust. In this particular face expression, the performance is pretty bad, with confidence values that still remains very low even when the face is clearly showing disgust.

We will deal later with these results traying to find out how to interpret them.

This performance was good enough, but we still tried to build and train our own model from scratch to see if we could do even better.

- **New model**

Dataset:

This model is trained on the same exact dataset of the pre-trained model: *FER-2013 emotion dataset*.

The dataset is split in the following way:

- 89% training set (28,709 examples)
- 11% test/validation set (3,589 examples)

This is the distribution of the test set:

- 958 angry images
- 111 disgust images
- 1024 fear images
- 1774 happy images

- 1233 neutral images
- 1247 sad images
- 831 surprise images

As we can see the distribution of the test set reflects the one of the training set. We'll use the test set to validate the performance of the method after each epoch and then save the weights if the model improved its performance.

Loss function and Metrics:

Since we're dealing with a multi-class classification problem, we're using *categorical cross-entropy* as loss function. *Accuracy* is used as a metric.

Architecture:

The network contains three convolutional layers, each followed by a batch normalization layer (which should help the network converge faster), a rectified linear operation and a max pooling layer. The first Convolutional Layer contains 64 filters of 3×3 pixels, the second Convolutional Layer contains 128 filters of 5×5 pixels, the third layer contains 256 filters of 3×3 pixels. Finally, two fully connected layers are added: one of 256 neurons, and the last (output layer) with 7 only neurons (the total number of classes).

The number of epochs is set to 30, and the batch size to 64 (this will help the algorithm converging faster). The optimizer is Adam with a learning rate of 0.0005.

Results:

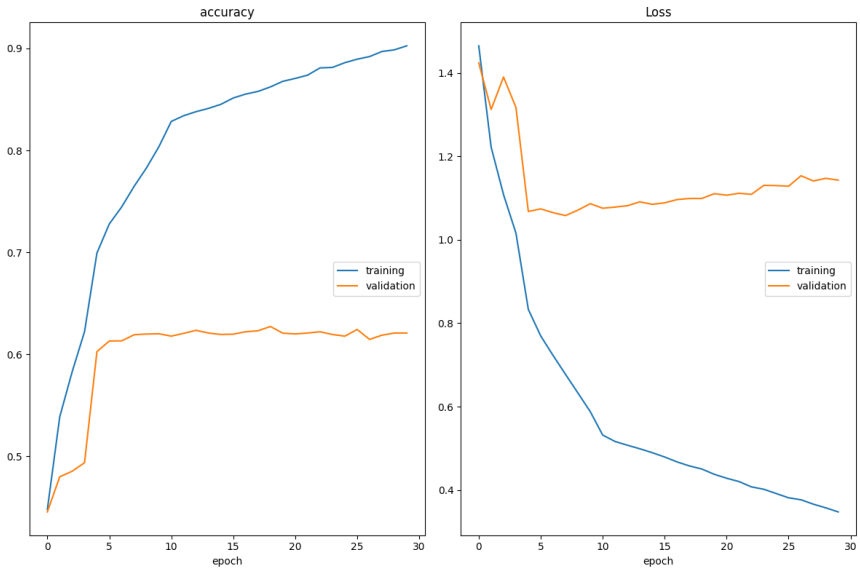


Figure 4: Loss and accuracy for training and test.

As we can see, the network obtained 0.621 as validation accuracy after 30 epochs. The score is not bad, but we can try improving this result by modifying our architecture a little. In fact, as we can clearly see from our plots, the training loss keeps decreasing while the validation one starts a little grow. This is a clear symptom of *overfitting*: the model has too many degrees of freedom and starts learning more and more accurately the distribution of the training set, but it is not able to generalize. We can try building a bigger model but add a lot of regularization to it to help the model generalize better.

Here are some result of this first model:

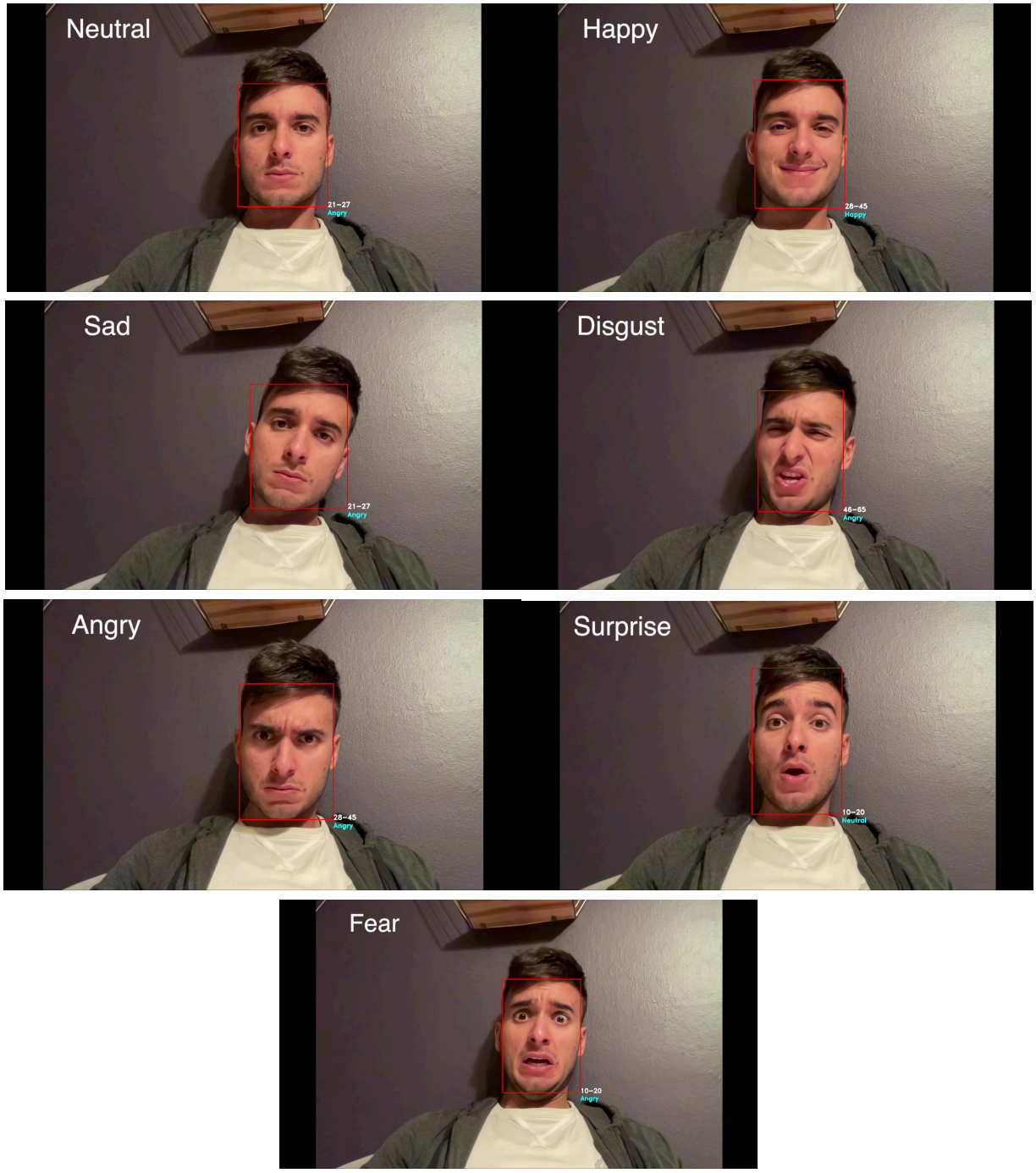


Figure 5: Some results of the model.

These results are significantly worse than our pretrained model. As we can see, the network seems able to distinguish only 3 expressions: happy, angry and neutral. This is not enough for our project, so we tried to build a better model.

Improvements:

The new network contains four convolutional layers, each followed by a batch normalization layer, a rectified linear operation, a max pooling layer and, most importantly, a Dropout layer. The dropout is a regularization method that approximates training a large number of neural networks with different architectures in parallel. During training, some number of layer outputs are randomly ignored or "dropped out." This has the effect of making the layer look-like and be treated-like a layer with a different number of nodes and connectivity to the prior layer. Theoretically, this should solve the main overfitting issue that we saw on our fist model. The first Convolutional Layer contains 64 filters of 3×3 pixels, the second Convolutional Layer contains 128 filters of 5×5 pixels, the third and the fourth layer both contains 512 filters of 3×3 pixels. Finally, three fully connected layers are added: one of 256 neurons, one of 512, and the last (output layer) with 7 only neurons (the total number of classes).

The model has now 4,474,759 trainable params, against the 2,863,495 of the first network. This bigger architecture and, most importantly, the dropout should help the model to generalize better.

The number of epochs is still set to 30, and the batch size to 64. The optimizer is Adam. But let's see the results:

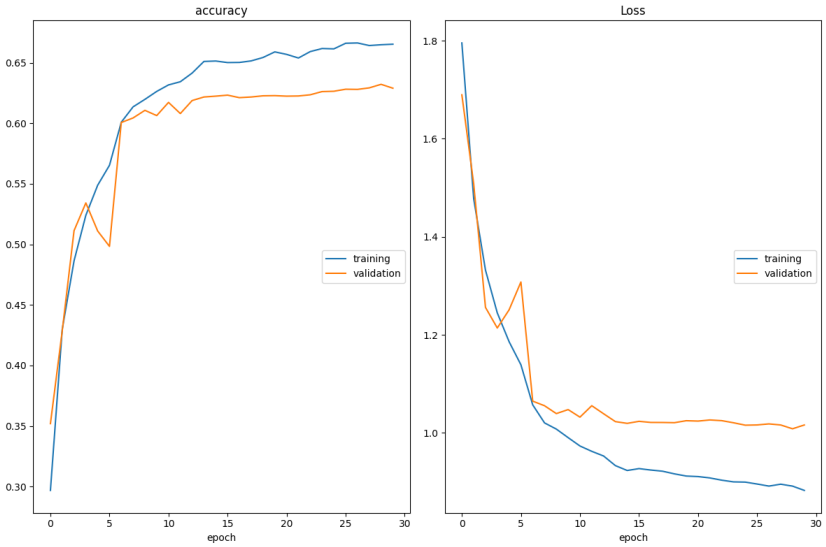


Figure 6: Loss and accuracy for training and test for the bigger model.

Now the curves are different from before. Not only the training loss and accuracy are lower (because of the regularization), but also the *generalization gap* (the gap between the training and

the test losses) is reduced. This probably means that the overfitting is reduced, even if it is still present. Our best test accuracy is now 0.632, which is a little improvement from before.

We can see the real improvement testing the model on our video:

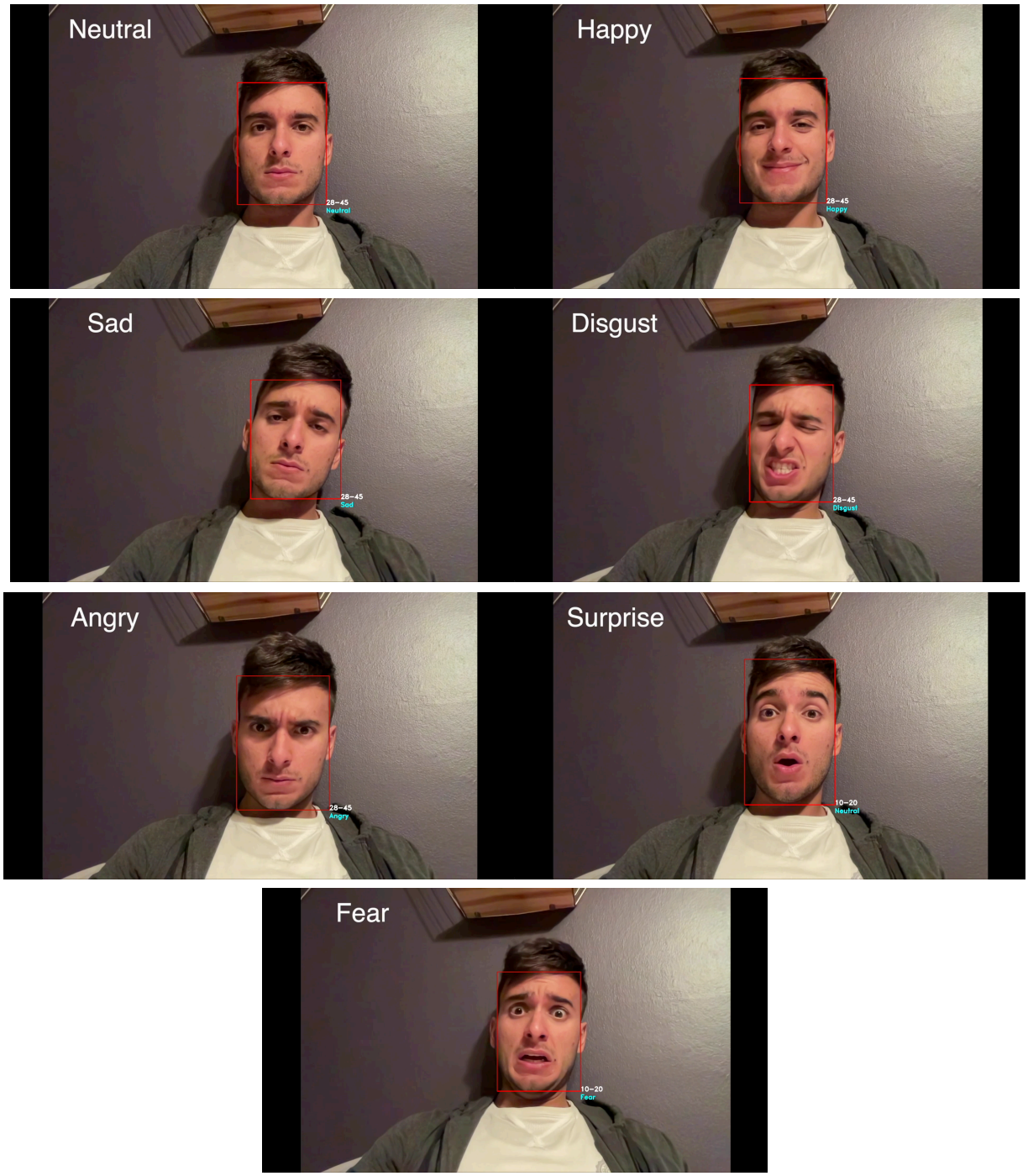


Figure 7: Some results of the bigger model.

Now the performance of the model seems improved a lot, there are still some minor errors, but we can be satisfied by this detection.

Analysis

Facial Emotion Recognition is a quite complex task. In this type of classification problem, CNNs and deep learning are the most suitable techniques, which can outperform almost every classical machine learning or computer vision approach. In fact, due to the complexity and variability of human facial expression emotion features, traditional facial expression emotion recognition technology has the disadvantages of insufficient feature extraction and susceptibility to external environmental influences. The feature extracted using convolutional neural network instead ignores the problem of subtle changes in facial expressions.

However, these models are still far from being perfect, and our analysis demonstrates it. One major issue could be caused by the dataset used to train the models. FER-2013 dataset is very popular and widely used but employing it without any data augmentation technique could mean a major issue in recognizing disgusted faces, for example. This is clearly demonstrated by the pre-trained model, which shows good/acceptable performance for every kind of face except for disgusted one. To be noted is that this first pre-trained model used very few weights if compared to our models build from scratch. The efficiency of the network is still impressive, however our last model outperforms it in almost every situation.

Unfortunately, the model is trained with the same FER-2013 dataset, and while it seems to solve the “disgust” issue, it shows bad performance for surprise emotions. The surprise class is the second class less represented in both the training and test set, and this could mean that it is still difficult to detect.

Another important limit of deep learning techniques is that they rely a lot on the architecture of the model and on the hardware employed. As we could see, training a bigger model and adding regularization helped a lot the network to reduce overfitting and help it to generalize. But training bigger and bigger networks could be too much for certain commodity hardware. So, these methods typically require finding a balance between performance and complexity/expensiveness of the training.

To conclude, in my projects I wanted to start from a classical computer vision research (face detection) and show how important is the mixture between classical Computer Vision techniques and Deep Learning ones. Especially for complex detection/recognition problems, it is almost impossible nowadays to achieve good results without relying on a Deep Neural Network.

Flow Chart and code

To access the code of this project:
https://github.com/Matteo299/FacialEmotionRecognitionModel.git

- emotion_detector_small.ipynb: first CNN trained
- emotion_detector.ipynb: final CNN model
- pretrained_emotion_detector.ipynb: pretrained model to detect expressions in video
- video_person_detector.py: python script to detect ages and expressions in videos
- webcam_person_detector.ipynb: notebook to detect ages and expression using the webcam