ESIEE
PARIS
École Supérieure d'ingénieurs en
Électrotechnique et Électronique

Machine Learning II
Final Project : Modern face recognition with deep learning

Auteur :
Vítor Fonseca Loures

Teacher :
Giovanni Chierchia

7 novembre 2019

1 Introduction

This report describes the final report of the course : Machine Learning II that aims at develop a deep neural network for recognizing people's face. The goal of the network is identify faces in some picture and identify whose this face is. This kind of application is already used in some social networks or photos application for automatic labeling the people's present in some photo.

We develop the project in two steps : first we develop the neural networks using a data set with images from Jurassic Park movies to fast test it and in the second step we build a personal dataset to test the project.

The system recognition system uses the following pipeline :

1. **Face detection** : Look at a picture and find all the faces in it.
2. **Pose estimation** : Understand where the face is turned and correct its pose.
3. **Face encoding :** Pick up unique features from a face that can be used to distinguish it from others.
4. **Face recognition** : Compare the unique features of a face to those of all the people in a database.

2 Face detection

This is the first part of the system pipeline. The face detection aims at detecting the area of the picture that contains a face, in order to output it to the next step of the pipeline.

He we disposed of a small dataset of pictures containing exactly one face in each picture. We extracted the faces and the label associated and saved in a pickle file, where data is stored in a serialized object structure. Then we normalized the cropped faces, dividing the pixel values by 255 to let it between 0 and 1 for each pixel. After, we splited the data in train set, containing 70% of the dataset and test set, containing 30%. The other important preprocessing step used was transforming the outputs in test and train set to one-hot enconding formating, as we are going to do a multi-class classification. At this point the data was preprocessed and we should focus on designing a good neural network to perform the task.

The approach done was to choose simple convolutional neural networks (CNN) architectures and fine the model according to the results. The first architecture chose was a CNN network containing two convolutional layers, which both had kernel with size 3 and pooling with size (2,2), the first had 8 filters and the second 16. After the flattening step, one fully connected layer was used in hidden layers, disposing of 32 nodes and connected to the output layer, that used a softmax activation function. The optimizer used was ADAM, that is the most common optimizer used in CNN architectures and we chose to follow this traditional approach. The loss function was the categorial cross entropy and it was chose by the same reason that took us to choosde the activation function in the output layer : the goal of the model is to perform a multiclass classification.

During the model development, the training and validation loss and accuraccy were ploted in order to evaluate the model quality in terms of learning ability, accuraccy and generalization capacity. The first architecture tested are shown in the Figure 1.

FIGURE 1 – Architecture containing 2 CNN layers and no dropout.

We can note that around the 10th epoch among the 20 used, the training loss keep descending and its accuracy keeps increasing while the validation measures stabilizes. It means that the model generalization capacity was not good due to the overfitting phenomen. To tackle this problem, we used the dropout technique in the convolutional layers in the second approch, showed in the Figure 2.

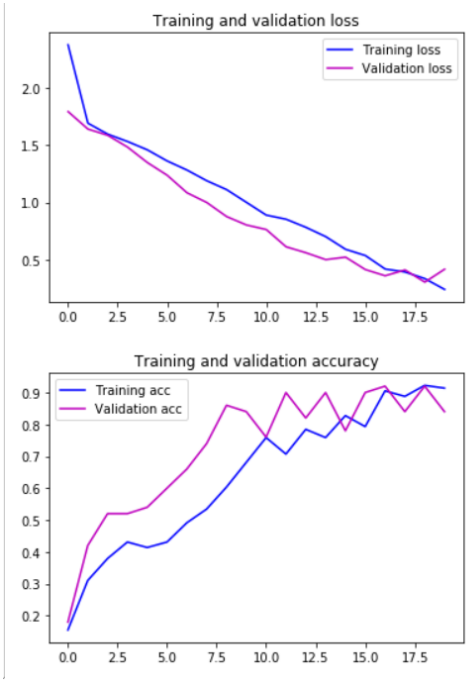

FIGURE 2 – Architecture containing 2 CNN layers and dropout.

After using the dropout, the validation and training curves showed the same trend either in loss or in accuracy graph and the training loss and accuraccy were worse than before using

dropout, as expected. It means that the overfittint problem was correctly solved. Then, we used a deeper network architecture to try to get a better accuraccy result. We added a new convolutional layer and increased the number of features in each layer. It was used 128 features in the first convolutional layer, 64 in the second, 32 in the third and 64 nodes in the flat and fully connected layer. The pooling and kernel keeped the same. The result of these modification are shown in the Figure 3.

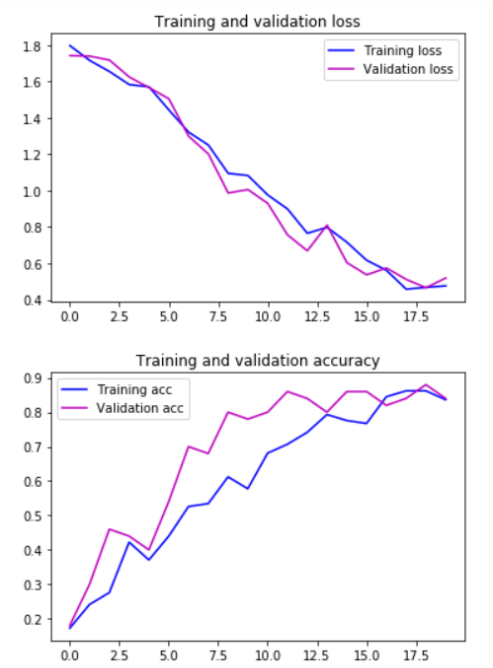

FIGURE 3 – Architecture containing 3 CNN layers, more features and dropout.

Even after building a deeper model, the validation accuraccy didn't increase significantly and, then, we stopped to change the architecture, choosing the last one tested to use in the project. The model summary are shown in the Figure 4. Its test accuraccy was equal to 84%.

Layer (type)	Output Shape	Param #
conv2d_34 (Conv2D)	(None, 126, 126, 128)	3584
max_pooling2d_31 (MaxPooling	(None, 63, 63, 128)	0
dropout_27 (Dropout)	(None, 63, 63, 128)	0
conv2d_35 (Conv2D)	(None, 61, 61, 64)	73792
max_pooling2d_32 (MaxPooling	(None, 30, 30, 64)	0
dropout_28 (Dropout)	(None, 30, 30, 64)	0
conv2d_36 (Conv2D)	(None, 28, 28, 32)	18464
max_pooling2d_33 (MaxPooling	(None, 14, 14, 32)	0
dropout_29 (Dropout)	(None, 14, 14, 32)	0
flatten_11 (Flatten)	(None, 6272)	0
dense_15 (Dense)	(None, 64)	401472
dense_16 (Dense)	(None, 6)	390

Total params: 497,702
Trainable params: 497,702
Non-trainable params: 0

FIGURE 4 – Face detection model summary.

3 Pose estimation

The pose estimation was the second step of the system pipeline. In this step, we used a face landmark estimation algorithm to locate specific point in the face detected to make the face recognition more accurate and deal with the problem that the face can be turned in different directions. Then, we alligned the faces in the same direction according to the landmarks obtained before. After, we saved the pickle file, normalized the image and split in the train and test sets as before.

For choosing a good network architecture we performed the same steps as before : a 2 conv net architecture without dropout was chose, the overfitting problem appeared, we added a dropout and then we added a new layer. We didn't add many features in this part of the pipeline and before because the accuraccy test accuraccy was already good in this part with 87.99% obtained. The Figure 5 shows the history training and validating curves and the 6 shows the model summary.

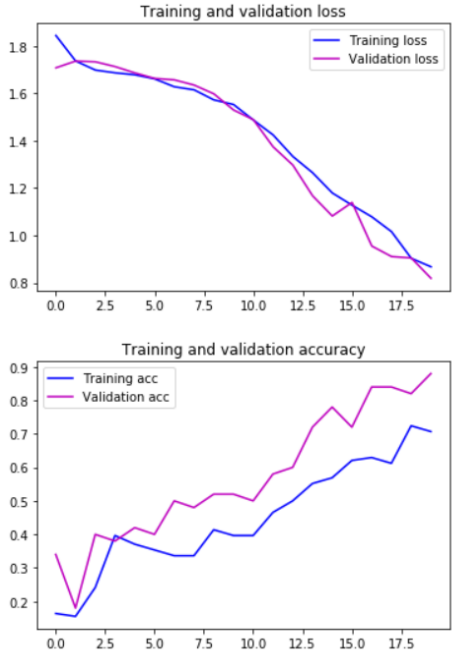

FIGURE 5 – Pose estimation model training history.

Layer (type)	Output Shape	Param #
conv2d_55 (Conv2D)	(None, 126, 126, 8)	224
max_pooling2d_46 (MaxPooling	(None, 63, 63, 8)	0
dropout_30 (Dropout)	(None, 63, 63, 8)	0
conv2d_56 (Conv2D)	(None, 61, 61, 16)	1168
max_pooling2d_47 (MaxPooling	(None, 30, 30, 16)	0
dropout_31 (Dropout)	(None, 30, 30, 16)	0
conv2d_57 (Conv2D)	(None, 28, 28, 16)	2320
flatten_18 (Flatten)	(None, 12544)	0
dense_29 (Dense)	(None, 32)	401440
dense_30 (Dense)	(None, 6)	198

Total params: 405,350
Trainable params: 405,350
Non-trainable params: 0

FIGURE 6 – Pose estimation model summary.

4 Face encoding

The idea of the face encoding is to optimize the process to find whose the face is. Instead of working to images we are going to work with a vector of 128 face measurements that are much more simpler and computationally more efficient to work than images. To know which measure we will get from each face, we imported a pre-trained model from OpenFace open source project, that is based on the CVPR 2015 paper FaceNet : A Unified Embedding for Face Recognition and Clustering by Florian Schroff, Dmitry Kalenichenko, and James Philbin at Google.

Due to the fact we working with vectors instead of images, we use a fully connected layers in all the layers of the model. We choose 3 fully connected layers with 128 nodes each and we obtained 100% test accuraccy in the end, that is very better than around 85% obtained in the two pipeline previous steps. The model training curve are shown in the Figure 7 and the model summary in the Figure 8

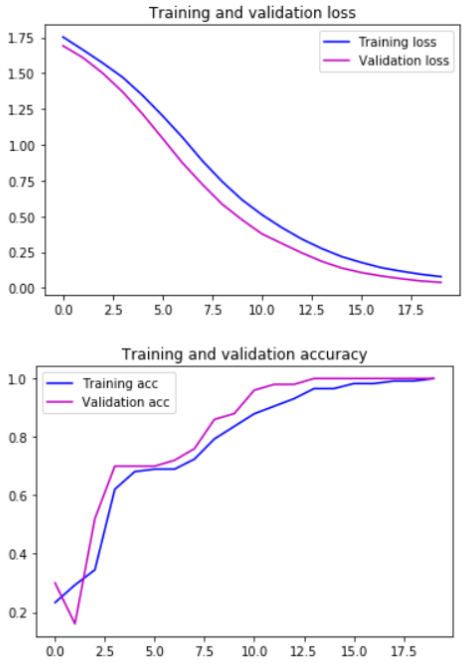

FIGURE 7 – Face encoding model training history.

Layer (type)	Output Shape	Param #
dense_51 (Dense)	(None, 128)	16512
dense_52 (Dense)	(None, 128)	16512
dense_53 (Dense)	(None, 128)	16512
dense_54 (Dense)	(None, 6)	774

Total params: 50,310
Trainable params: 50,310
Non-trainable params: 0

FIGURE 8 – Face encoding model summary.

5 Face recognition

This last step of the system pipeline is the simplest one. Here, we use a classifier to find the person in our database of known people who has the closest measurements to some test image. We trained 3 different classifiers and all the three obtained a 100% accuraccy. They were : suport vector machines (SVM), k-nearest neighbors (kNN) and the neural network Multi-layer Perceptron classifier (MLP).

We tested all the classifiers in a test image containing 3 people from our databases. The SVM made a mistake in classifing one person. The other two made the correct prevision and this result is presented in the Figure 9 above :

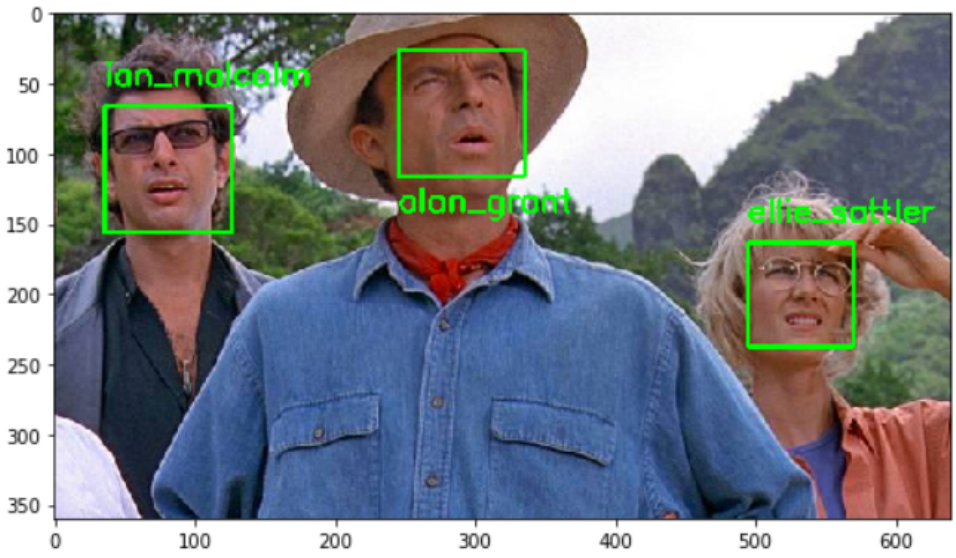

FIGURE 9 – Face recognition test using MLP classifier.

6 Builded dataset

In the second part of the project, after having created a model to perform a face recognition in the teacher's provided dataset, we builded a dataset, in which we used the builded system to perform facial recognition. The chosen dataset was one containg 12 politians, actual or former presidents or prime ministers from different countries. We called this dataset as 'politics_dataset' in this project. The 12 chosen personalities were : Shinzo Abe, prime minister of Japan, Jair Bolsonaro, president of Brazil, Dilma Roussef, former president of Brazil, Xi Jinping, president of China, Boris Johnson, prime minister of United Kingdom, Emmanuel Macron, president of France, Angela Merkel, Chancellor of Germany, Barack Obama, former president of United States, Donald Trump, president of United States, Vladimir Putin, president of Russia, Margareth Thatcher, former prime minister of United Kingdom and Justin Trudeau, president of Canada. For creating this dataset, it was used a video frame extractor and YouTube videos and after it was performed a manual step to eliminate all the images that contained more than one face or the wrong face, due to the fact that it was used video interviews and, generally, this kind of video not focalises just the person interviewed for all the time.

The system used in this project had the same pipeline, however we performed small modifications in network architecture, as adding early stopping. This dataset has 1889 images and 12 labels, while the original one has 218 images and 6 labels. As this dataset has more data, it learns better the features and the network provides better accuraccy in each step.

Due to the fact that this model is more likelly to learn using this dataset than the previous one, since the face detection, it could be noted that the model fast learned the features and stabilized near 0 loss and 100% accuraccy. Then, it was not necessary to perform 20 epochs, as we did in the previous system and we used a simple early stopping here, not to avoid an overfitting, as the usual application of this technique, but just to learning and academic purposes. The figures 10 and 11 shows the modifications brought by the using of early stopping

techniques. It worths say that the accuraccy didn't change, it keeped near 100%. The early stopping was used just to demonstrate the technique, not to increase the performance. The use of this technique obtained the same results, but using less epochs to train, which could be very useful for large datasets, that would take much time at each epoch.

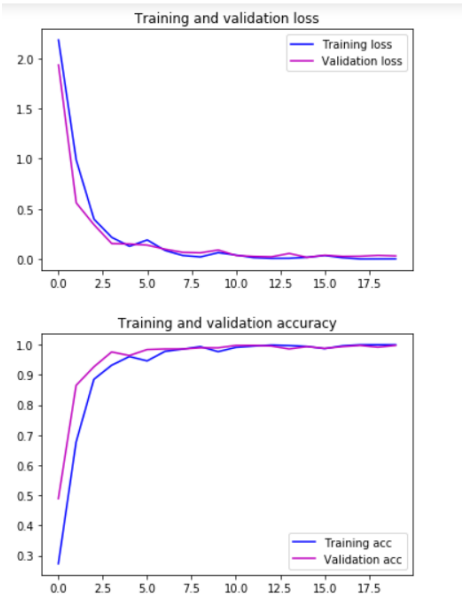

FIGURE 10 – Face detection training before early stopping.

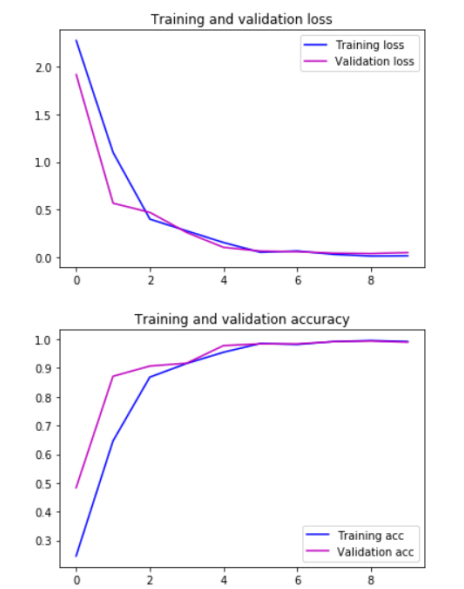

FIGURE 11 – Face detection training after early stopping.

7 Results

Some interesting obtained tests results are presented below. In the Figure 12 we can note that the network identified correctly the faces and who they belongs to.

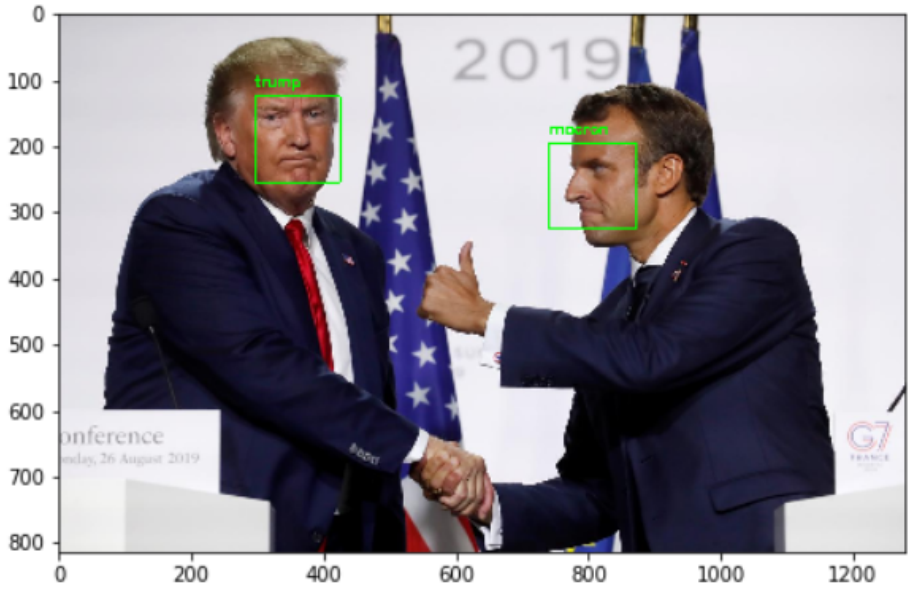

FIGURE 12 – Donald Trump and Emmanuel Macron correctly identified.

The Figure 13 has 3 politians that are in the dataset, two of them had theirs faces and owners correctly identified. On the other hand, Boris Johnson face wasn't either identified, probably because the network is weaker to identify perfil face's and the ilumination conditions were not favorable to him.

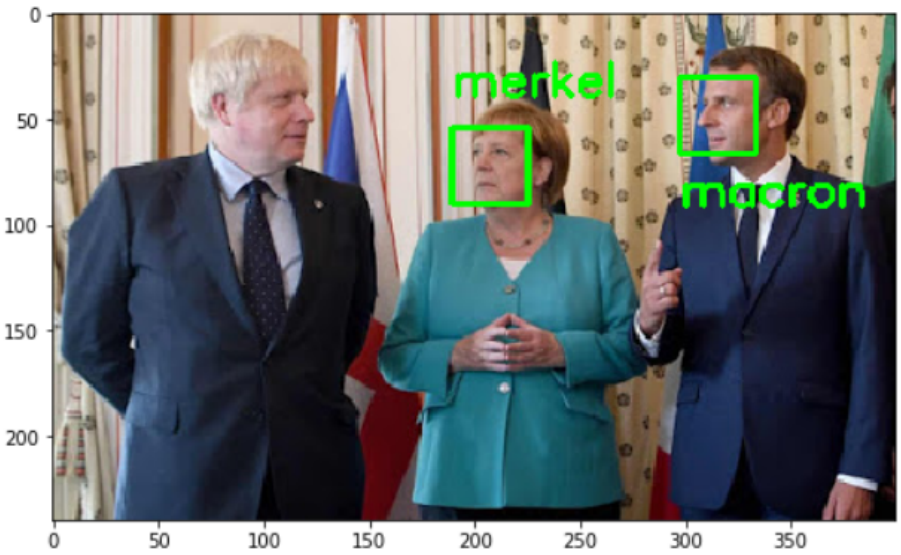

FIGURE 13 – 2 faces corrected recognized and one unrecognized face.

The Figure 14 also brings an interesting result. There is 9 people in the G7 meeting picture, which 5 are in the dataset and 4 are not, and theirs faces are more distant from the camera than in the training dataset. The model was able to correctly the all 5 politians they were trained to identify.

FIGURE 14 – Leaders existing in the database were correctly identified.

The Figure 15 shows an image of a protest, where the demonstrators used some politians puppers mask. The model identified some faces, but it recognizes 4 from 5 faces wrong. As these kind of masks distorces the measurements taken by the neural network, the network aren't able to recognize the correct owner, because it doesn't use the same features to recognizes someone as humans do.

FIGURE 15 – Bad results in puppet mask face identification.

On the other hand, the Figure 16 could correctly identify as the face, as recognizes who they belong for every person that is also in the dataset. The model can recognize well the

owner of the face in this case probably because the printed mask doesn't distorces too much the measurements taken to the neural network.

FIGURE 16 – Good results in printed mask face identification.

8 Conclusion

In this project we could have a better idea of how to use neural networks to perform some tasks, we could understand the idea of using pipelines in data science projects, we could see the problems appearing during the development and tackles them with the techniques seen in this course.