Report of the final project — Lab7

Antoine AUBERT
IA et Deep Learning
ESIEE – 2023
France
antoine.aubert@edu.esiee.fr

Pauline GOUILLART
IA et Deep Learning
ESIEE – 2023
France
pauline.gouillart@edu.esiee.fr

Abstract

Developing a face recognition system involves several challenges, including accurately identifying and distinguishing between different faces, especially those with similar features or orientations. To address these challenges, a pipeline of four steps can be used: face detection, pose estimation, face encoding, and face recognition. Pose estimation corrects for variations in face orientation, and deep learning identifies the most important features for face encoding.

Building a custom dataset for face recognition is another challenge, which can be achieved by gathering examples of faces to recognize via a webcam. Overcoming these challenges requires the use of state-of-the-art computer vision and machine learning techniques, including deep learning and classification algorithms like neural networks, logistic regression, SVM, and nearest neighbours.

The system will need to be carefully evaluated and validated to ensure accurate and reliable performance. These steps require significant computational resources and expertise in computer vision and machine learning.

Index Terms

Deep Learning, Face Recognition, Keras, Classification, Convolutional Network

I. INTRODUCTION

The goal of lab7 is to perform face recognition on a small database using a convolutional neural network (CNN) and compare its performance with other classifiers such as KNN, SVM or logistic regression. Three approaches are used for face recognition.

The first approach involves directly using the TensorFlow API Keras to build a CNN. In this method, raw images are used as input data, which is not highly efficient because the data is not normalized. Therefore, in the second step, data normalization is performed to improve the efficiency of the model. Finally, face encoding is performed using vectors, allowing us to use a dense network directly. This approach is used for many reasons, including the ability to compare results with other classification methods that use vectors.

II. RELATED WORKS

There are numerous online articles and open-source codes available on face recognition. Platforms like Medium or GitHub are a treasure trove of information.

We came across Krasserm's page (1) which has a companion notebook. He employs libraries such as OpenCV, open Face, Dlib, Keras, and evaluates efficiency with various classifiers using the LFW (2) dataset. His approach is similar to ours, and he also computes a threshold distance for each image to use the f1 score instead of precision. His results are impressive, achieving 98%. Additionally, we appreciated the 2D space visualization of the dataset to identify clusters.

We also found the paper titled "Sparse Representation with Principal Component Analysis in Face Recognition" (3). It presents a method for face recognition that combines sparse representation and principal component analysis (PCA). The authors used the ORL dataset to test their method and achieved high accuracy. However, their method is limited to frontal face images, and they did not consider variations in poses and lighting.

In the lab's credit section, there is a link to Adam Geitgey's post (4) titled "Machine Learning is Fun! Part 4: Modern Face Recognition with Deep Learning." This article provides guidelines and tips to start with face recognition.

III. PROPOSITION

In the following section, we will provide a detailed explanation of each part of our work and how we approached it. Before delving into the details, let us give an overview of the dataset. It comprises 218 pictures of well-known actors from Jurassic Park, and the images are not pre-processed, i.e., they are not cropped, normalized, or aligned. Our initial step involves extracting faces using Dlib and storing them as NumPy arrays along with their respective labels. We chose to use a vector structure of shape (207, 2) to store this data.

A. Face detection

Face detection and face recognition are two different processes. As stated in the Sciforce paper (5), "Face detection is the critical first step of face recognition, determining the

number of faces in a picture or video without remembering or storing details."

For face detection, the received data is composed of 218 images of Jurassic Park actors: 35 for Owen Grady, 53 for Claire Dearing, 41 for Ian Malcolm, 36 for John Hammond, 31 for Ellie Sattler and 22 for Alan Grant. Each picture does not have necessarily the same size at the beginning: for instance, it can be 416x500, 1920x1026 or 1440x653. It is necessary to do face detection for this data to be able to tell faces apart after having located them in the picture.

To pre-process the data, we used multiple techniques. First, we extracted the faces and the labels of the provided small dataset of pictures to attribute later a face to a person's name. While doing this, it allows the pictures to be cropped to the same size for an easier processing. Then, we created a hot one encoder for each label that helps to normalize the data. Finally, we split the data by using the train_test_split function from Scikit-learn. The random state parameter is predefined. This ensures that the dataset is split randomly, but consistently across multiple runs. Thanks to these different steps, batches could be prepared.

For the small convnet, we decided to use diverse types of layers: SeparableConv2D for reducing the number of parameters, MaxPooling2D for taking the maximum value of the sub-matrix, Conv2D for extracting features from the input data by applying filters, Flatten for flattening the last matrix, Dense for learning complex patterns with linear and non-linear functions. The last layer has the purpose of giving the probability that the picture belongs to a specific category.

When talking about our small convnet, the performance on the train set reaches 0.85 with 0.45 of loss while the test set reaches 0.75 with 0.76 of loss.

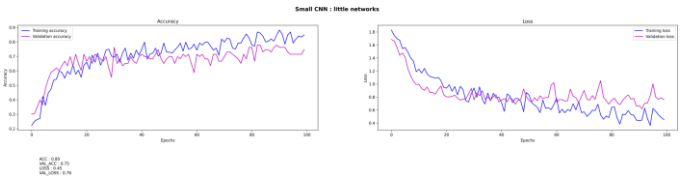


Fig.1 - Small CNN: little networks

When talking about our CNN, the performance on the train set reaches 0.88 with 0.39 of loss while the test set reaches 0.84 with 0.45 of loss.

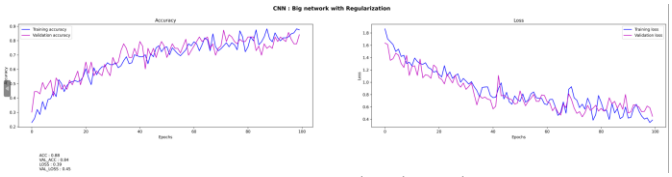


Fig.2 – CNN: Big network with Regularization

We can deduce that our convnet with a better performance is the big network with regularization. Indeed, the result can improve or decrease when changes are made on the number of layers, number of hidden units or learning rate initial value.

B. Pose estimation

"Pose Estimation is a computer vision task where the goal is to detect the position and orientation of a person or an object. Usually, this is done by predicting the location of specific key points", as stated in Papers With Code (6).

The data for pose estimation comes from the previous step of face detection. Now that the faces are isolated and normalized, the pictures to work with have the same dimensions: 128x128.

The pose correction of pose estimation is necessary to have the same face orientation in the picture. Indeed, to center the faces that might be from the side helps to get more identical pictures that are easier to recognize. This can be done thanks to face landmarks detected by key points that adapt their position to transform the face while respecting its characteristics.

The technique that we used to correct the face pose is aligning all the key points of a face in a centred way. When it was done, the faces now cropped, aligned, and normalized were divided into training and test thanks to the function split_data.

Fig.3 – Faces with key points

Fig.4 – Aligned faces

When talking about our pose estimation convnet, the performance on the train set reaches 0.8 with 0.53 of loss while the test set reaches 0.76 with 0.54 of loss.

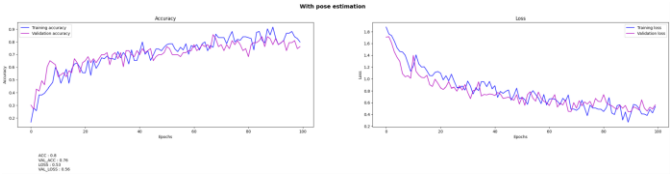


Fig.5 – With pose estimation

We can deduce that the results are quite similar with the previous convnets. With a higher number of epochs, we can believe that the results would be higher. It is also possible that are first convnets were already really optimized for having good accuracies thanks to the work that we have done on the training and test datasets. Finally, the last explanation would be to parameter in another way our model to gain accuracy.

C. Face encoding

As stated in Ai ML Analytics (7), “Encoding is a technique of converting categorical variables into numerical values so that it could be easily fitted to a machine learning model.” Concerning face encoding, facial features from the pictures need to be encoded to get their characteristics transformed from pixels to measurements.

The advantage of face encoding is that it allows for efficient and accurate comparison of faces for identification or verification purposes. By encoding faces into vectors, it is easier to compare faces using mathematical operations thanks to a set of unique features rather than on the entire image.

The data used during face encoding is the previous data from pose estimation that has been encoded with the function face_encoder. After applying this function, the data is now under the form of a list of generated measurements. For this new data, we apply the split into training and test before using a new type of network relying only on Dense layers. Indeed, now that the data is no longer pixels from the pictures but measurement, the network need to be adapted.

When talking about our face encoding convnet, the performance on the train set reaches 1.0 with 0.0 of loss while the test set reaches 0.94 with 1.25 of loss.

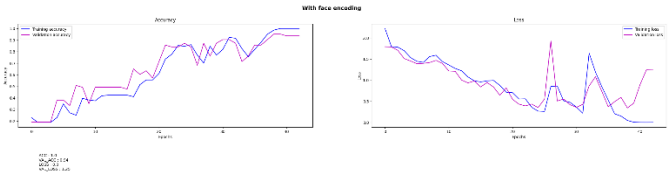


Fig.6 – With face encoding

We can conclude that, according to the previous convnets, it is the best one yet. Indeed, its results are quite outstanding. It is thanks to the fact that we used measurements instead of the whole picture. This more mathematical way to proceed has proved its efficiency.

D. Face recognition

While face detection is the process of locating human faces in an image or video, face recognition is the process of identifying or verifying a person's identity based on their facial features. As said by J. Brownlee (8), “Face recognition is the problem of identifying and verifying people in a photograph by their face.”

For this task, we chose to compare the KNN, SVM and Logistic Regression classifiers. Each one has its own characteristics. Firstly, the KNN (K-Nearest Neighbors) is a non-parametric algorithm that can handle complex decision boundaries and can be used for both classification and regression tasks. For the SVM (Support Vector Machines), it finds the optimal hyperplane that maximizes the margin between two classes, and it can handle high-dimensional data and non-linear decision boundaries through kernel trick. Finally, Logistic Regression is a simple and interpretable linear model that provides probabilistic outputs, and it can be easily updated with new data in an online learning setting.

When talking about the KNN classifier, the performance score is 98.61% for both “7” and “3” as parameters but the execution time is better for “3” (3.969ms VS 0.941ms). For the SVM classifier, the performance score is 100.00% for both “linear” and “rbf” as parameters but the execution time is better for “linear” (0.077ms VS 0.134ms). As for the Logistic Regression classifier, the performance score is 100.00% for an execution time of 0.028ms.

```
classifier_KNN : (7)
Execution time : 3.969 ms
Score          : 98.61 %
-----------------------------------

classifier_KNN : (3)
Execution time : 0.941 ms
Score          : 98.61 %
-----------------------------------

classifier_SVM : ('linear')
Execution time : 0.077 ms
Score          : 100.00 %
-----------------------------------

classifier_SVM : ('rbf')
Execution time : 0.134 ms
Score          : 100.00 %
-----------------------------------

classifier_LR :
Execution time : 0.028 ms
Score          : 100.00 %
-----------------------------------
```

Fig.7 – Classifiers

The best classifier when taking into consideration the performance and the execution time is the Logistic Regression, then followed by the SVM and finally the KNN.

E. Personal dataset

To create our dataset, we took between 50 and 65 pictures of 14 people. To get diverse images in a short time, we tried to have different kinds of background in a same room. We asked people to do diverse poses by changing their facial expressions and by using their hands. The pictures were taken from multiple angles to get various data.

Unfortunately, we did not have the time to use our personal dataset thus no possibility to compare our previous dataset with a new one. Our personal database was supposed to be host to GitHub but we could not upload it because of its size even as a .zip.

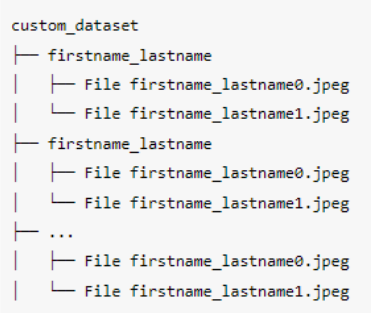


Fig.8 – Custom dataset

F. Extra - Bias analysis

Bias refers to the systematic and consistent errors or inaccuracies in the data, algorithms, or decision-making processes that favour certain groups or outcomes over others. It is a problem in machine learning because it can lead to unfair or discriminatory outcomes. If the data used to train a machine learning model is biased, the model can learn and perpetuate those biases in its predictions and decisions. This can result in unequal treatment of different groups of people, reinforcing societal prejudices and inequalities.

For example, if a facial recognition system is trained on a dataset that mostly includes white males, the system may not perform as well for individuals from other racial or gender groups. This is because the system has not been trained on a diverse dataset and therefore may not accurately recognize individuals from underrepresented groups. In this situation, bias can lead to misidentification or discrimination, which can have serious consequences.

When we created our personal dataset, we took the easy way by asking our friends who were available in the moment. In the friend group that was taken in picture, different ethnicities and different genders were represented. The issue that we could encounter with our dataset is the fact that we only used pictures of students and no younger or older people. Indeed, the age range in the pictures is between 18- and 23-year-old. We could ask ourselves if the model could be bias because of this.

CONCLUSION

This project on face recognition involved four main components: face detection, pose estimation, face encoding and face recognition. We started with a dataset of 218 pictures of actors from Jurassic Park where the images were not pre-processed. To adjust the data as we needed, we used Dlib to extract faces, pre-processed the data, and split it into training and test datasets. Thanks to the new data, we were able to train correctly several convolutional neural networks (CNNs). The CNN with the best performance was the big network with regularization for face detection, which achieved an accuracy of 0.84 on the test set. The performance of the pose estimation and face encoding CNNs was slightly lower but still promising. What was demonstrated is the fact that classifiers could do the sale work as CNNs with sometimes even a better performance.

In conclusion, we have learned how to use CNNs for face recognition tasks but mostly the highlighted point of the project was the importance of pre-processing and normalization techniques in improving model performance. This aspect was indeed shown thanks to the use of classifiers.

PROJECT

Our project is available on GitHub:
https://github.com/Aubert-Antoine/EL_3003-IA_et_Deep_Learning/tree/main/LAB%207%20FaceRecognition

REFERENCES

(1) “Deep face recognition with Keras, Dlib and opencv”, *Deep face recognition with Keras, Dlib and OpenCV - Martin Krasser's Blog*.
[Online]. Available: https://krasserm.github.io/2018/02/07/deep-face-recognition/
[Accessed: 21-Apr-2023].

(2) “Labeled faces in the wild home,” *LFW Face Database: Main*.
[Online]. Available: http://vis-www.cs.umass.edu/lfw/
[Accessed: 21-Apr-2023].

(3) M. C. Yo, S. C. Chong, K. K. Wee, and L. Y. Chong, “Sparse representation with principal component analysis in face recognition”, *MMU Institutional Repository*, 01-Jan-1970.
[Online]. Available: http://shdl.mmu.edu.my/10882/
[Accessed: 25-Apr-2023].

(4) A. Geitgey, “Machine learning is fun! part 4: Modern face recognition with deep learning,” *Medium*, 24-Sep-2020. [Online]. Available: https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78 [Accessed: 21-Apr-2023].

(5) https://medium.com/sciforce/face-detection-explained-state-of-the-art-methods-and-best-tools-f730fca16294

(6) “Pose estimation”, *Papers With Code*. [Online]. Available: https://paperswithcode.com/task/pose-estimation [Accessed: 25-Apr-2023].

(7) Tkhan.kiit@gmail.com, “Different types of encoding”, *AI ML Analytics*, 2022. [Online]. Available: https://ai-ml-analytics.com/encoding/ [Accessed: 25-Apr-2023].

(8) J. Brownlee, “A gentle introduction to deep learning for face recognition”, *MachineLearningMastery.com*, 05-Jul-2019. [Online]. Available: https://machinelearningmastery.com/introduction-to-deep-learning-for-face-recognition/ [Accessed: 25-Apr-2023].

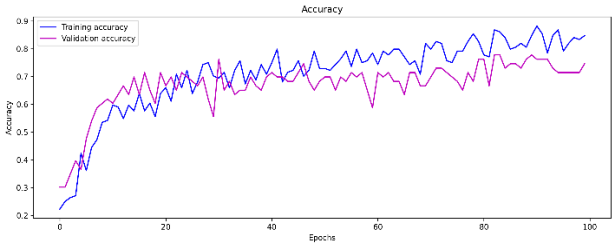
“Keras imagedatagenerator with flow(),” *Study Machine Learning*. [Online]. Available: https://studymachinelearning.com/keras-imagedatagenerator-with-flow/ [Accessed: 21-Apr-2023].

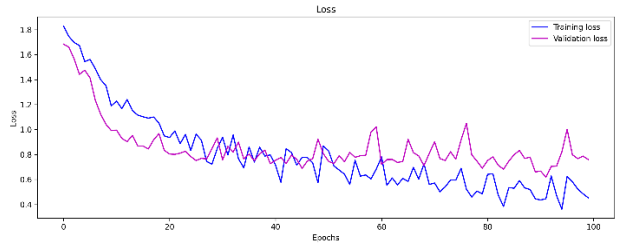
https://github.com/Aubert-Antoine/MIT-introtodeeplearning#lecture-videos

APPENDICES

To have a better perception of the previous figures, it is possible to find them in a bigger size.

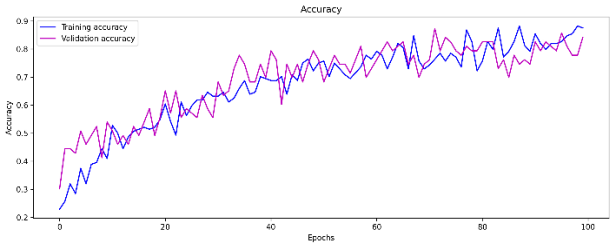
Small CNN : little networks

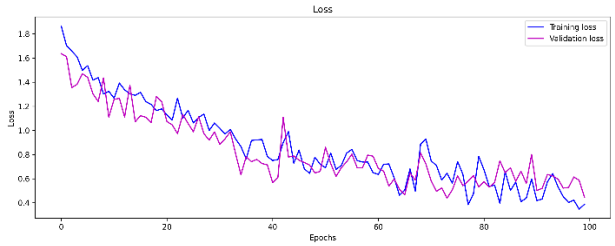




ACC : 0.85
VAL_ACC : 0.75
LOSS : 0.45
VAL_LOSS : 0.76

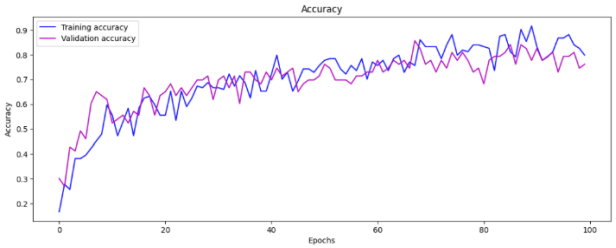
CNN : Big network with Regularization

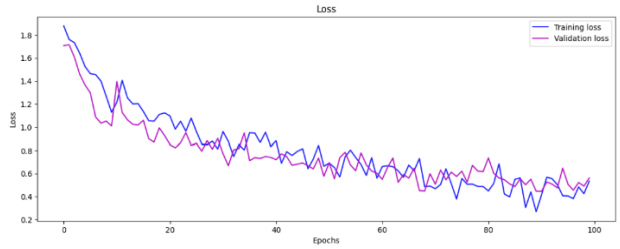




ACC : 0.88
VAL_ACC : 0.84
LOSS : 0.39
VAL_LOSS : 0.45

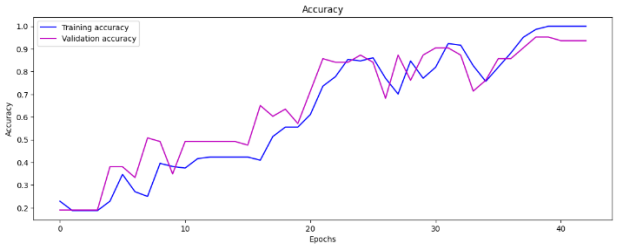
With pose estimation

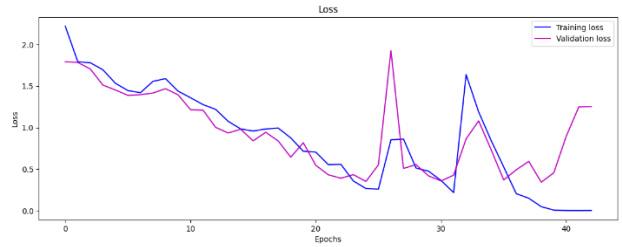




ACC : 0.8
VAL_ACC : 0.76
LOSS : 0.53
VAL_LOSS : 0.56

With face encoding





ACC : 1.0
VAL_ACC : 0.94
LOSS : 0.0
VAL_LOSS : 1.25