Université Nice Côte d'Azur
Enrico FORMENTI

Année 2021/22
M1 Info
NNL

Project Exam
Part 2/2

Facial recognition: the mask/no mask case

Contents

1 Before starting

Also this sencod part of the project needs to work in an adapted environment. Therefore, you need to install more libraries if it is not already done:

- `NumPy` : normally you should already have this library installed, just check to have the most recent version.

- `TensorFlow` : this is a library specialised in deep learning. Instructions for the installation can be found here. We are most interested by the Keras sub-library which is specifically conceived for the neural networks models that we are going to use. Usually, Keras is directly bundled with TensorFlow.
- It would also be interesting for you to follow the following excellent tutorial which recalls the CNN (Convolutional Neural Networks) model seen in our course and that you are invited to use for developing this project. Recall that they are multi-layer networks in which each layer has a specific purpose. Moreover, the output of each layer is obtained by the convolution of a suitable operator over the input. This tutorial recalls the global structure of a CNN and its functioning.
- Undoubtedly, it would be an asset to have access to the book of Chollet [1] and carefully reading the Sections 2.2, 2.3, 2.4, 3.1, 3.2 and the 5th chapter.

TensorFlow has not been ported to Python version 3.10 yet. Therefore it is recommended to implement your project in Python version 3.8 or 3.9.

We are going to briefly recall the most common layers used in imaging processing by saying a few words for each of them and by referring to specialised tutorials or to the Keras' API for more on parameters meaning and tuning. Here is a list of the layers that we think useful in the sequel (alphabetical order):

conv2D: this a layer which applies a convolutional operator on a 2D structure (image) and outputs a structure on which other layers will apply. Here you can find a good application example and here you can find a tutorial which explain how to choose the parameters.

dense: it is a layer in which the graph of the connections is complete. Better to read the Keras API's specification here.

dropout: this layer is used for regularisation purposes and to avoid overfitting. You can find an interesting tutorial here.

maxpooling2D: this layer is used to reduce the dimensionality of the input data by taking the maximum over the window of the convolutional operator passed as a parameter. Here you can find the Keras API.

2 The final project

The project aims at realizing a predictor for images containing people faces. The predictor should recognize the bounding box of a face and classify it in the categories *mask/no mask* if the face in the bounding box wears a mask or not. The rest of the section will provide the necessary details.

2.1 Building the data set

You are supposed to produce your own data set. To this purpose, just take your mobile phone and take photos of people around you, in the streets or in a store. A *hundred* of photos should be enough, at least in an initial stage.

First of all, use your Annotator tool to annotate the images. Be as careful as possible with the bounding boxes and avoid to classify clear *outliers*. You can consider outliers all the bounding box which are too small or too big in terms of surface but also those which are too thick or too thin in terms of dimensions. When the predictor will be finalized, please find appropriate threshold values for surface and dimensions of what should be considered as an outlier.

Then, extract from your images all the annotated bounding boxes and save them as new images. For the saved images it would be convenient to adopt the following standards:

filename: give standard names to all newly produced image as follows. If the original filename of the image is `myimage.jpg`, then the new image filename shall be `myimage-bb-XxY-W-H.jpg` where (`X`, `Y`) and (`X+W`, `Y+H`) are the coordinates of the top left and bottom right points of the bounding box (recall that in `tkinter` the top left corner of your screen has coordinates `(0,0)`).

image size: in order to have simpler models, it is better to have equally sized square images in your data set. Small sizes will speedup the learning process but a too small size possibly entails low predictive power. In practice, anything square of side between 120 and 180 pixels should do.

file format: in order to simplify the code for your input layer it is a good practice to have all images in the same format, say `png` or `jpg`.

At this point your data set is ready! Now you need to step to the training phase.

2.2 The training phase

In this phase you should compile a learning model (it would be a good idea to save the model structure in a file together with some significant characteristics – like, for example, the number of layers and their types, the number of connexions, the total number of neurons, *etc.*). Then, it will be time to run your model training phase. You are supposed to make experiments to find the better set of parameter for your model.

Reports and comments on your experiments at training level will be certainly taken into account in the final evaluation of the project.

2.3 Prediction and test

You are supposed to write a predictor adapted to the model that you trained in the previous section and then evaluate its performances as we have seen in our TD classes. Practically speaking, you should write a function `predict` which takes in input the filename of an image and a string `mode` and outputs `mask`/`nomask` if `mode='category'` or it outputs a list of probabilities if the `mode='probabilities'` indicating for each category the probability the input image belongs to it.

In your report, it will be good if you take a step back from your work and judge, for example, if your model suffers from overfitting, underfitting or more in general what could be the improvements to bring.

3 Project evaluation

The project aims at representing a real application case of the concepts seen during the NNL course. The evaluation will therefore take into account three aspects:

(C) the code;

(P) the performances (predictive power, training speed, predictive speed);

(R) the report.

Details about how each of those aspects is evaluated as follows.

3.1 The code (C)

Like in all programming projects, the program should meet all the specifications detailed so far. In this part, we are going to evaluate the quality of your code, the solutions adopted, the genericity and the quality of the documentation.

3.2 Performances (P)

In this part we are going to evaluate the performances of your code in terms of *accuracy*, learning speed and reliability. In order to measure reliability, we are going to test your project against a small set of images which will be used also for all other projects.

3.3 Rapport (R)

The last (but not least!) part of your project consists in writing a short report (no more than 30 pages). This report should contain *a minima*:

- a short description of the project and its goals;
- a description of the data set used and how it was built or retrieved;
- the methodology followed to answer the project questions or to overcome the difficulties encountered during its development;
- the results obtained and, whenever possible, their graphical visualization;
- a section with conclusions and perspectives;
- a short presentation of each team member and of his role in the project.

Reports written in LaTeX will receive higher marks!

3.4 Marks

All parts of the project will be evaluated up to 5 points. This will provide a note $C + P + R$ to which the mark (also up to 5 points) for practice classes TP will be added.

Extra points are awarded to those who extend the project with extra capabilities such as recognizing the type of the mask worn (ffp2, surgical mask, homemade, *etc.*) and/or fully integrate the project with the annotator interface (see the first part of the project).

Beware! An antiplagiarism software (against the Web or the other projects) will be run on your project when evaluation `(C)` or `(R)`. Actions will be taken in case of clear plagiarism.

4 The teams

Teams will be the same as in the first part of the project. No changes are allowed.

5 What to submit and when

You should submit a compressed (only `.zip` format will be accepted) file with a name `NNL-2021-XX.zip` where `XX` is the ID of your team. This archive **must** contain the following four directories: `pre`, `src`, `img` et `rep`. These directories are supposed to contain, respectively:

- the sources of the scripts or the programs used for data pre-treatment if any;
- the Python scripts for your project;
- the graphical data or any other image used in the project;
- the report (if you make in LaTeX, please provide full source, tables and figures to be able to produce a pdf from sources).

Deadline: submit your project via the Moodle website of the course before **January 25th, 2022 at 12 o'clock**.

5.1 Late penalties and deadline extension requests

If the project is not submitted within the deadline, you should expect penalties on the rating. The penalties will be proportional to the delay. No deadline extension without penalties is allowed, except for exceptional cases which will be judged case by case.

6 How to submit your project

Projects must be submitted via `Moodle`. In the tab `Project`, you will find an activity called `Submit the project - (part 2/2)`.

ℹ The approval of a single team member is enough to complete the submission activity.

7 SOS!

If during the project you are stuck for some reason, do not stop there! Please apply the usual algorithm:

1. redouble your efforts for a finite time (1 day maximum);
2. if Step 1. does not give the expected results then ask you class mates;
3. if Step 2. does not work either then ask to former students of the class (in Master 2 at present);
4. if it still does not work, then contact your teacher urgently!

Steps 1. to 3. must not take more than 3 days!

Good luck!

References

[1] François Chollet. *Deep learning with Python*. Manning publishing, 2017.