LABARRE Vincent (only 1 person for this project)

Detection of mask wearing during the Covid-19 health crisis:

Step 1 - Find the data:

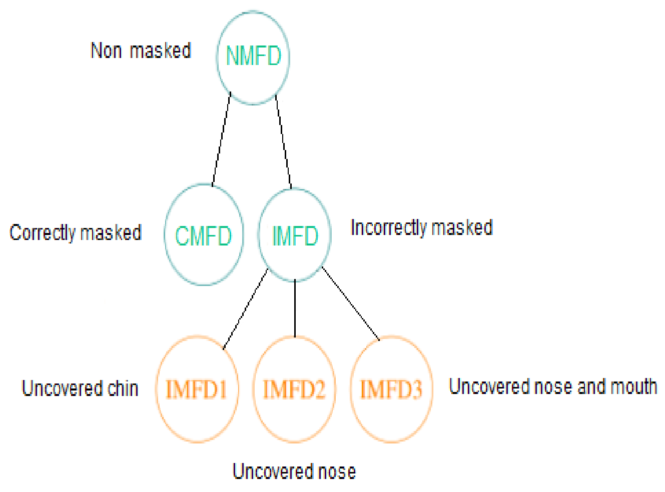
With the emergence of the covid-19 health crisis and the urgent need to put in place barrier gestures to combat this virus, many establishments, particularly those that cater for the elderly, need to ensure that visitors or workers wear masks. Therefore, a mask monitoring system can be put in place to ensure that masks are worn properly. In this case, datasets with a large number of images must be available to set up these systems. Thus, I could easily find 3 datasets [1] that I needed:

- a **Correctly Masked Face Dataset (CMFD)**: 67,193 images at 1024×1024 [2]
- an **Incorrectly Masked Face Dataset (IMFD)**: 66,900 images at 1024×1024 [3]
- a **Non-Masked Face Dataset (NMFD)**: 70,000 images at 1024x1024 [4]

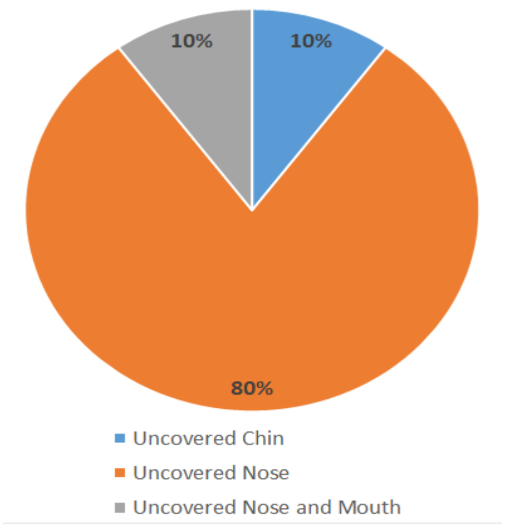
However, the first two datasets are from the third dataset, so I will have to take this into account when establishing the training, testing and validation sets in order to avoid bias and to facilitate the generalization of the model.

As a result, with these datasets, I will have in inputs images of size 1024x1024.

Organization of the 3 datasets:



Incorrectly Masked Face dataset structure:



Step 2 - Determine the good model:

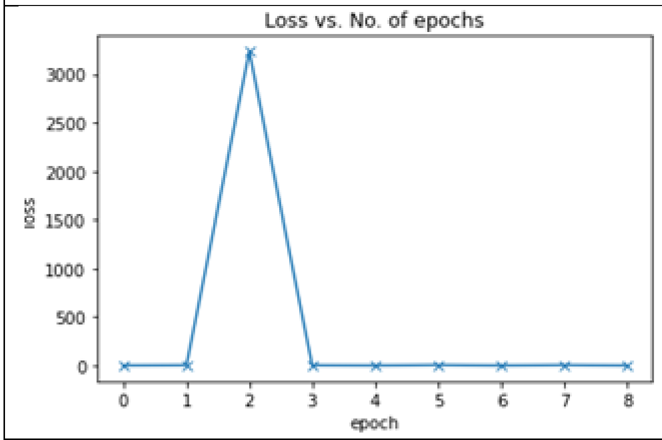
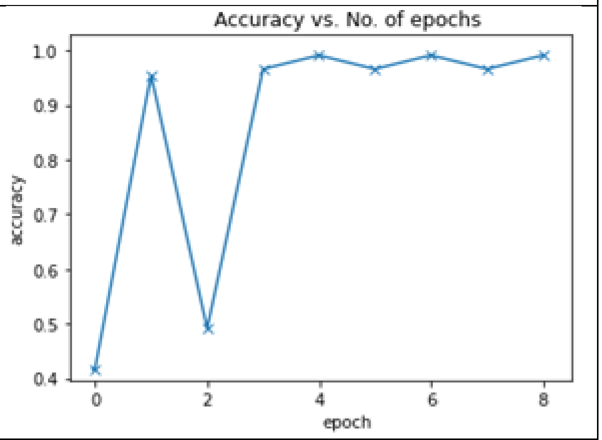
In the literature is used mainly models from Deep Learning such as CNN, R-CNN or YOLO [5] but here we have to take a machine learning method. Therefore, I chose the **logistic regression** because it is a semi-parametric method, so we do not make assumptions directly on the distribution but on a distribution ratio, which makes the assumption less restrictive.

Step 3 - Implementation of the chosen model:

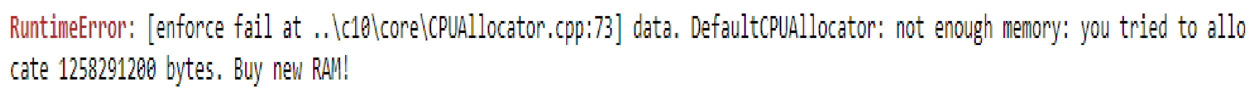
To implement the logistic regression, I first took a look at a video [6] to see how it works with an example (here it was with MNIST). Moreover, I saw more in details this example with another document [7]. This helped me a lot to write the code for this part of my project.

In order to write it, I had to get acclimatized to Pytorch, which took me some time. But in the end, I was able to write the script without too much difficulty. More precisely, I used in the code (like the document [7]) the **softmax function** to convert the output into probability and the **cross-entropy function** for the loss function (fits well with the softmax function outputs).

Step 4 - The results of the chosen model:

Constitution:	
No mask dataset (from NMFD)	200 images (**labeled as 0**)
Mask dataset (from CMFD)	200 images (**labeled as 1**)
Repartition:	
Training dataset	280 images (70% of all images, including 50% with mask and 50% without mask).
Validation dataset	40 images (10% of all images, including 50% with mask and 50% without mask).
Testing dataset	80 images (20% of all images, including 50% with mask and 50% without mask).
Batch size	40
Epoch	8
Results for the training phase:	
	
Results for the testing phase:	
Loss value: 4.3797	**Accuracy value**: 0.9750

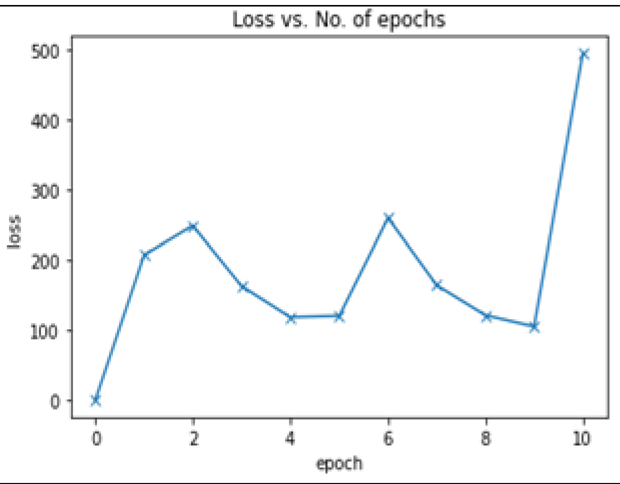
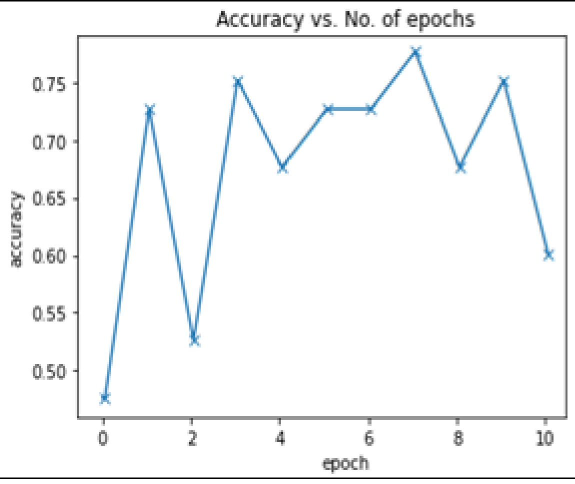
Note: I cannot take more images than 400 because of my insufficient RAM (which is quite frustrating considering the number of images I could use ..., I do not know if there would be a way to remedy this problem.).



```
RuntimeError: [enforce fail at ..\c10\core\CPUAllocator.cpp:73] data. DefaultCPUAllocator: not enough memory: you tried to allo
cate 1258291200 bytes. Buy new RAM!
```

The obtained results are very good. But afterwards I decided to take some images from internet of masked and unmasked people (with a mask of different types and colors). And therefore, I realized that I have a serious bias in my model: the masks I can recognize are only surgical type blue masks (indeed, in the CMFD and IMFD there are only this type of masks). As a result, I have to make a choice: either I decide to be able to detect only masks of this type (which are the mandatory ones to go to the most vulnerable places), or I decide to be able to detect any type of mask (or at least, any mask of a different color).
For the project, I can take the first choice, but I thought about something that maybe could improve the model. Indeed, I did the model with RGB images, but what will happen if I take grayscale images?

Then I rewrote the code to work with grayscale images and I got this (with same parameters as above except: <u>*epoch*</u> = 10):

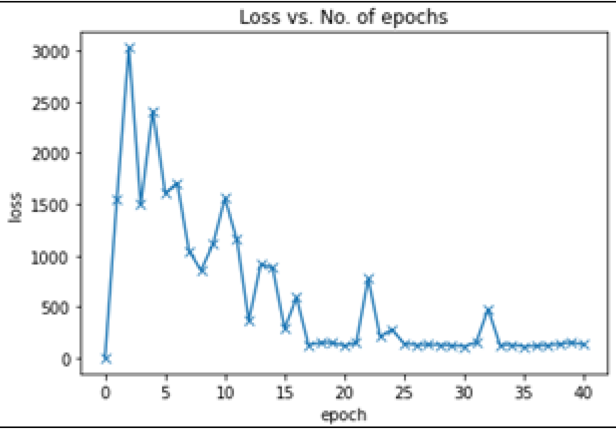
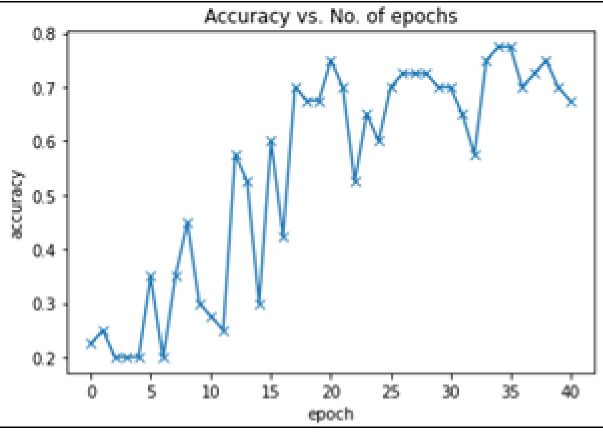
Results for the training phase:	
	
Results for the testing phase:	
Loss value: 410.4377	**Accuracy value**: 0.6000

For consequence it was a bad idea: I had worse results than before (with the color images), with a model that is not stable here. This is explained by the fact that when switching from a color image to a grayscale image, one loses 2/3 of the image information, so it is much more difficult for the model to differentiate mask pixels from other pixels.

Finally, I applied this model (logistic regression with RGB images) to determine if the masks were well worn or not: therefore, I created 5 classes (in order of decreasing severity of non-compliance):

- Class of **Non masked faces** (100 images) with **label = 0**.
- Class of **Incorrectly Nose and Mouth masked faces** (100 images) with **label = 1**.
- Class of **Incorrectly Nose masked faces** (100 images) with **label = 2**.
- Class of **Incorrectly Chin masked faces** (100 images) with **label = 3**.
- Class of **Correctly masked faces** (100 images) with **label = 4**.

Then, I kept the same parameters as above (except *epoch* = 40) and I obtain this:

Results for the training phase:	
	
Results for the testing phase:	
Loss value: 59.5049	**Accuracy value**: 0.7250

Therefore, we see that the results are not as good as in the case where we only try to determine whether the mask is worn or not. This could be explained by the fact that not enough images are used to train the model or simply because the model is not powerful enough.

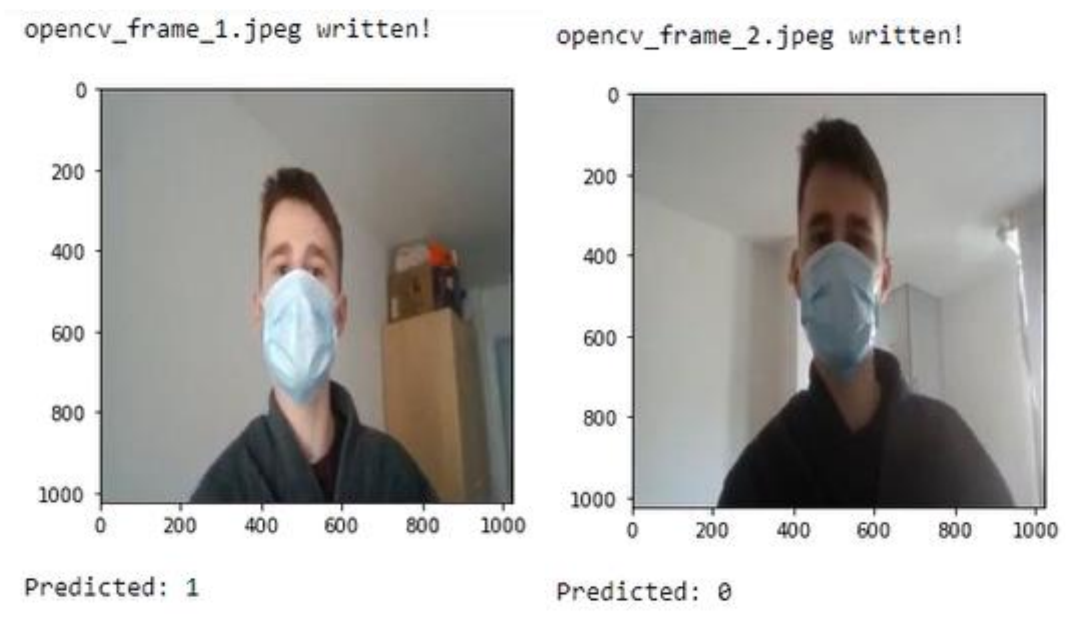
To conclude this part, since I clarified and specified the goal of this project which is to detect surgical type masks (those required for a good protection), I had to verify if the model worked well on any images of faces with surgical mask. For that, I took images of this type from the Internet and I did the test. However, it was really hard to find images of faces with surgical mask in format 1024x1024 with a good quality. Actually, I only find 7 images of this kind and on these 7 images, the model correctly labeled 3 of them. Afterwards, by adding 13 other images with a low quality, I possessed 20 images in total and on these 20 images, only 5 of them got correctly labeled (therefore 2/13 for the added images).

Finally, even if the model works well on images taken from the datasets CMFD, IMFD or NMFD it has difficulties to generalize on any images of faces with surgical mask. The problem may be that I cannot use enough images because of the RAM to train the model optimally or because the model is not sufficiently powerful. However, for the second case, I do not think that another classification model would be better than the logistic regression. In my opinion, to drastically improve the model, deep learning methods have to be used (as the litterature suggests this).

<u>Step 5 – Switch from static to dynamic detection:</u>

To find the way to do it, I looked to many youtube videos. In one video [8], they did a web interface to display the video (from a phone or webcam) and they captured an image from it to be able to apply their model. But the drawback here is that you have to click on a button to make a screen of the video. In my mind, I imagined that the detection would be done directly in the video. And actually, the aforementioned video is like a "semi-dynamic" detection. But after some researches, I found it that would be a good compromise to rapidly implement a solution.

Thus, I kept my logistic regression model that worked well on the training, validation and testing set and I just wrote code which enabled me to open my webcam and get the video streaming, take photos by pressing the space button and display the photos (recorded in a special file) with the associated predicted label (and to stop the video, I had to press the esc button). For example, I obtained this:


```
opencv_frame_1.jpeg written!
Predicted: 1
```
```
opencv_frame_2.jpeg written!
Predicted: 0
```

Note: To able the model to analyse as better as possible the images, I crop images taken from the webcam and I resized them.

Eventually, the model didn't work very well: to obtain a good predicted label, we have to be in the middle of the photo with a very high luminosity on the face, especially on the mask.

After few tests, results weren't as good as I hoped but it was not so bad (with many tests, I can state that in the good position described just before the model correctly predicted in 40% of the cases).

Conclusion:

This project allowed me to understand and assimilate the essential steps for the implementation of a machine learning solution: first of all, it is necessary to have at my disposal a large source of data in order to facilitate the training of my model. Then, I need to determine a model that is suitable for solving my problem and to do this, I need to define precisely what I want to do: for example, in my case, I had to specify that I wanted to detect a person wearing a surgical mask. Finally, I must analyze the results of the chosen model in a very critical way in order to realize the main problems it causes and correct them.

Therefore, in order to improve this project, a way should be found to train the model with as much data as possible (here I trained the model with only 0.4% of the data I had at my disposal because of RAM problems). To do this, one would have to look at the different solutions offered by the cloud computing.

Moreover, we could try another machine learning technique such as decision trees to see if it fits better or not our problem. But I think that the best way to improve the model is to use deep learning methods (as suggests the litterature). Then maybe in the deep learning course I will have the opportunity to redo/improve this project by adapting it to deep learning techniques to see if there is a (big) improvement (or not) of the results.

LINK GITHUB TO THE CODE

[1]: GitHub - cabani/MaskedFace-Net: MaskedFace-Net is a dataset of human faces with a correctly and incorrectly worn mask based on the dataset Flickr-Faces-HQ (FFHQ).

[2]: CMFD - OneDrive (sharepoint.com)

[3]: IMFD - OneDrive (sharepoint.com)

[4]: NFMD – Drive Google

[5]: Fighting against COVID-19: A novel deep learning model based on YOLO-v2 with ResNet-50 for medical face mask detection

[6]: Logistic Regression using Python (Sklearn, NumPy, MNIST, Handwriting Recognition, Matplotlib)

[7]: https://jovian.ai/aakashns/03-logistic-regression

[8]: Deep Learning tutorial – Neural Network on Mnist Dataset and Webcam Detection