ID 2223 - Scalable Machine Learning and Deep Learning for the 20/12/2020

Group 12 - Hugo Nilsson and Nino Segala

The Final Project

Face mask detection in real-time video stream

Presentation of the project:

The goal of our project was to detect if the persons in a video stream are wearing a mask or not. We wanted to mix what we have seen during this course (classifying an image), something new (handling a video stream and the library openCV) and a relevant topic (the obligation of wearing face masks in some countries due to the coronavirus).
To do so we will get the video stream from the webcam of our computer and analyse if the persons in the stream wear a face mask.

Architecture of the solution:

Our solution is implemented in Python and makes use of OpenCV as well as Tensorflow with Keras running on top of it.

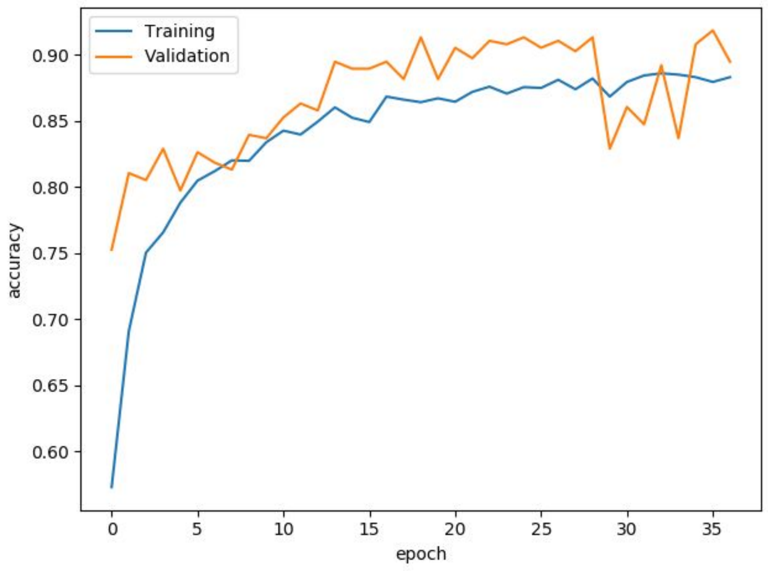
First we divided the project into two parts, one part is the face detection, and the second is to classify whether or not the face has a mask.

Let's start with the second task. To solve the binary image classification problem a Convolutional Neural Network (CNN) was built and trained using labeled data with faces with/without masks [4].

We tried several things to optimize the network, such as batch regularization and using a pretrained network, but in the end the generalization techniques we used were dropout, data augmentation and early stopping. This since, it yielded the greatest result and was also relatively fast to train.
The network uses convolutional layers, each of these layers is followed by a MaxPool layer, in the end a fully connected dense layer is used followed by a single output layer containing only one node since the model is doing binary classification.The architecture can be found in Appendix A.
For the hidden layers Relu was used as activation function and sigmoid for the output layer since it's a binary output.
To train the model we splitted the data into 80/10/10 split (training-, validation- and test-set), and didn't augmentate the validation and test set. Since the data was a bit limited we decided to use most of it in the training set.

To train the model we used the optimizer Adam with a learning rate of 1e-4, and a batch size of 32. The model was set to train for 60 epochs but early stopping stopped the training after 36 epochs.

The final loss on the test set was.0191 and the accuracy 95,6%.

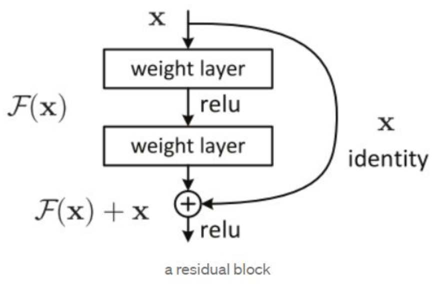


Here is a graph displaying the accuracy of the training set and the validation set during training.

The model was then saved as a tensorflow network file.

After the model was done we implemented the face detection. First we started with the “Haar Feature-based Cascade Classifiers” which is an algorithm for object detection. Unfortunately, this turned out to be problematic, since once the mask was put on the Cascade classifier would not recognize the face. However, this model could very accurately and fast tell when you weren't wearing a mask.

Therefore we needed to broaden our horizon and instead decided to use a pre-trained neural network [2]. This neural network is based on the ResNet10 network, which is a small 10 layers network [2] and was slightly modified. The ResNet network uses identity shortcut connection to avoid the vanishing gradient issue. A copy is sent 2 or 3 layers forward and added with the result that we got through these layers like in the following image.



<u>ResNet’s identity shortcut connection</u>

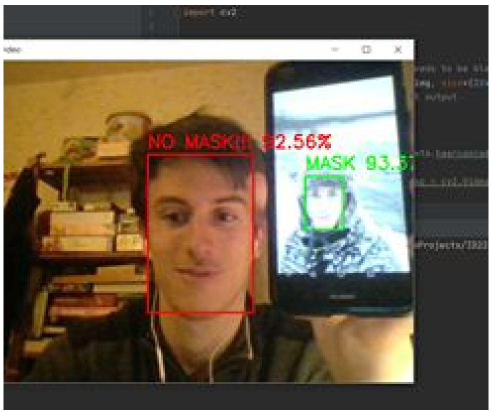
Our final mask detector ended up as following:

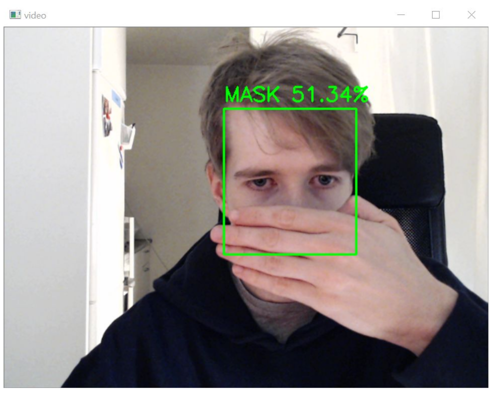
- Load the pretrained model for face detection as well as the model for mask/no mask classification
- open the webcam
- For each frame
    - detect the presence of faces using the pre-trained neural network
    - detect for each face if the person wears a mask or not using our mask/no mask classifier
    - display the result on the video stream

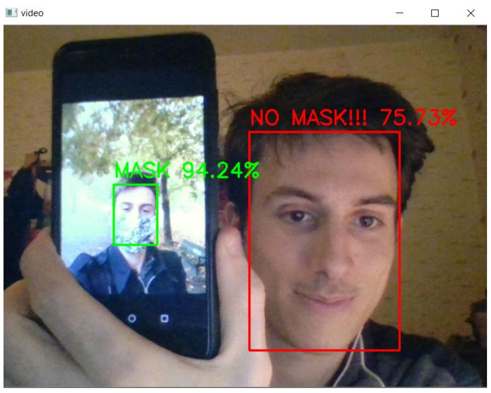
Results:

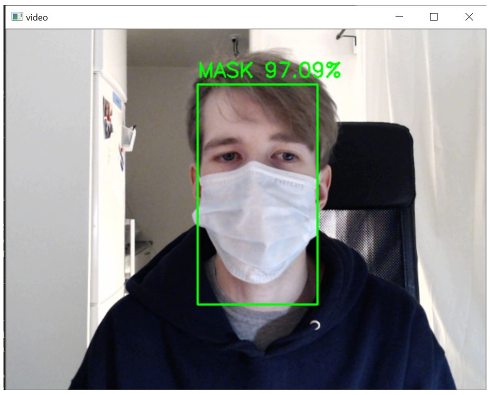
We are able to open the webcam, which starts the video stream, to get each frame and to evaluate if there are persons and if they wear a mask or not.

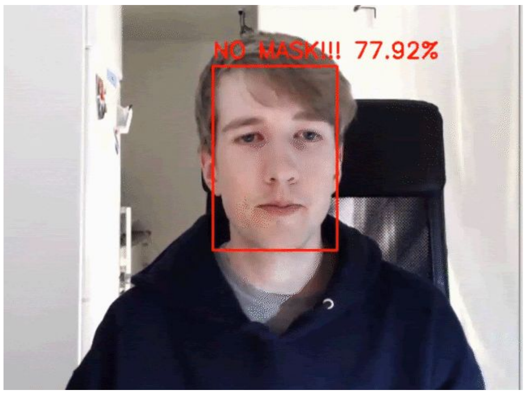
Correct detections, multiple faces detections and errors











In the first image, we can see that Nino and a photo of Nino are detected. It is possible to detect several faces at the same time and to analyze for each person if a mask is worn. The algorithm predicts correctly that Nino doesn't wear a mask, but it also predicts that Nino wears a mask on the photo, while he actually wears a hat. It is possible that the concept of mask isn't correctly learned with the dataset we use, which will explain why an ambiguous case like this one (a person wearing a hat), is evaluated as wearing a mask.

In the second image, we have shown that it is possible to trick the algorithm by covering our mouth. Still the confidence is quite low for the mask detection (51%).

In the third image, two faces are detected and both are correctly predicted.

In the fourth image, Hugo wears a mask and it is well predicted.

In the gif, we can see how fast the algorithm detects the presence of the mask when Hugo put it on.

How to run the code:

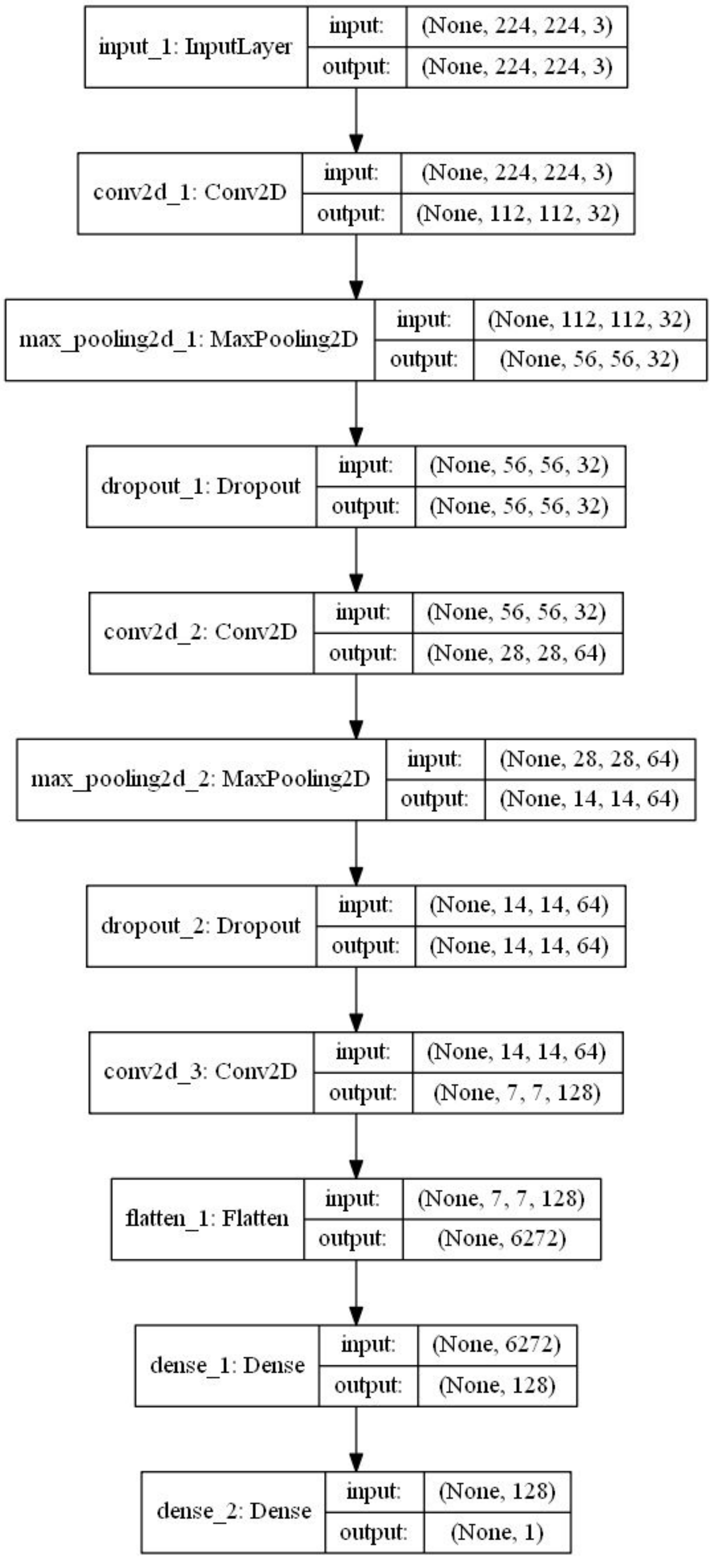
The code can be found on our github [3], and in order to run it, you also need to download the dataset [4,5].
First train the mask/no mask model using mask_detector_model.py.
Then you can run detect_mask_in_video_advance.py which will predict and display if the persons in the video stream from the computer's web camera are wearing a mask or not.

Libraries needed: Tensorflow, Keras, Numpy, sklearn matplotlib, tqdm, os and openCV2

References:

[1] https://github.com/opencv/opencv/tree/3.4.0/samples/dnn/face_detector
[2] Towards lightweight convolutional neural networks for object detection
https://arxiv.org/pdf/1707.01395.pdf
[3] https://gits-15.sys.kth.se/hugonil/ID2223-Project
[4] what data will you use and how are you going to collect it?
https://drive.google.com/drive/folders/1XDte2DL2Mf_hw4NsmGst7QtYoU7sMBVG
[5] information about the origin of the data:
https://github.com/chandrikadeb7/Face-Mask-Detection

Appendix A: Mask classifier architecture


input_1: InputLayer
input:
(None, 224, 224, 3)
output:
(None, 224, 224, 3)
conv2d_1: Conv2D
input:
(None, 224, 224, 3)
output:
(None, 112, 112, 32)
max_pooling2d_1: MaxPooling2D
input:
(None, 112, 112, 32)
output:
(None, 56, 56, 32)
dropout_1: Dropout
input:
(None, 56, 56, 32)
output:
(None, 56, 56, 32)
conv2d_2: Conv2D
input:
(None, 56, 56, 32)
output:
(None, 28, 28, 64)
max_pooling2d_2: MaxPooling2D
input:
(None, 28, 28, 64)
output:
(None, 14, 14, 64)
dropout_2: Dropout
input:
(None, 14, 14, 64)
output:
(None, 14, 14, 64)
conv2d_3: Conv2D
input:
(None, 14, 14, 64)
output:
(None, 7, 7, 128)
flatten_1: Flatten
input:
(None, 7, 7, 128)
output:
(None, 6272)
dense_1: Dense
input:
(None, 6272)
output:
(None, 128)
dense_2: Dense
input:
(None, 128)
output:
(None, 1)