
Fontys University of Applied Sciences

# Face Mask Detection

*Data Challenge II Report*
*Applied Data Science Minor*

Flyn Heijmans, Mario Lucassen

Eindhoven, 21st March 2021

# Abstract

In the current corona situation people are obliged to wear masks. In this challenge, experiments have been done in order to create deep learning models capable of predicting whether a person is wearing a face mask, and if they are wearing it correctly. To assist building these models, a data preparation pipeline has been made using the TensorFlow data API. Explainable Artificial Intelligence has been applied to visualise the actions taken by the detection model during prediction. Finally, an interactive demo has been made to demonstrate the models in live action. In the end, the best performing face mask detection model has achieved an accuracy of around 95%, the best performing face mask judging model has achieved an accuracy well over 99%. Both of these models have been validated using images of ourselves, which the model has predicted correctly. The results suggest we have completed the challenge successfully, as the accuracies of the models vastly surpass our expectation.

# Contents

# Revision History

| Revision | Date | Author(s) | Description |
|---|---|---|---|
| 1.0 | 26/11/2020 | FH, ML | Created |
| 1.1 | 01/12/2020 | FH, ML | Complete initial version |
| 1.2 | 03/12/2020 | FH, ML | Added feedback from Simona |
| 1.3 | 22/12/2020 | FH, ML | Added XAI research and face mask judging model |
| 1.4 | 23/12/2020 | FH, ML | Added Interactive demo and added performance metrics |
| 1.5 | 07/01/2020 | FH, ML | Added feedback from Simona |

# Acronyms

**API** Application Programming Interface. 6–8, 10, 22

**CNN** Convolutional Neural Network. 12, 13, 22

**DL** Deep Learning. 6, 12, 14

**DVC** Data Version Control. 8, 9

**ReLU** Rectified Linear Unit. 12, 13

**TF** TensorFlow. 6, 7, 10, 22

**XAI** Explainable Artificial Intelligence. 6, 7, 22

# 1 Introduction

The objective of this report is to describe the approach and findings of the Deep Learning (DL) challenge. For the Deep Learning challenge, we will take the opportunity to work on multiple deep learning models that classify face mask data. To use the face mask data efficiently, we will make use of the TensorFlow (TF) data Application Programming Interface (API) [5] for building a data input pipeline for training of the models. After building the models, we want to focus on finetuning them by applying data augmentation and optimising hyperparameters to increase accuracy. During the modelling, we would like to apply Explainable Artificial Intelligence (XAI) to visualise the different choices the models make.

This report starts with the approach chapter, it entails how we approach this Deep Learning challenge. Followed by the data preparation chapter where the preparation of the data is explained. Afterwards, in the modelling chapter, the different models to tackling this challenge is elaborated up on. Then, in the evaluation chapter, the results of this challenge are presented using XAI. Lastly, a conclusion chapter is written with the highlights of the results and what we have learned during this challenge.

## 1.1 Goal

The goal of this DL challenge is to reach an accuracy as high as possible for the tasks at hand. The accuracy we aim for is 80%. A side goal is to see if we could correctly classify on pictures of ourselves whether we are wearing face masks, and if time permits, make an interactive demo of our model where you can try to classify whether you are wearing a face mask and if it is being worn correctly, live on webcam.

## 1.2 Datasets

The datasets used for this deep learning challenge all govern people wearing or not wearing a face mask. To start, only the Kaggle face mask dataset is used to start with this challenge. If we deem the dataset not to be enough for the training of the model, we can search online for more datasets on face masks.

### 1.2.1 Kaggle face mask dataset

The data we are using is from a Kaggle dataset [1] on people that are or are not wearing a face mask. The dataset is very recent, uploaded on 2020-10-28. But the quality of the data is very low. It has a usability score of 1.3 on Kaggle, meaning that it does not pass any of the standards set by Kaggle for uploading a dataset. It consists of 3846 images of people with or without face masks. Where 1930 images are without mask, and 1916 images with mask. The images are all in different sizes, which is a problem to tackle during preparation phase.

### 1.2.2 MaskedFaceNet dataset

The second dataset we are using is the MaskedFaceNet dataset [3], which contains 130K faces divided into 51% of faces correctly wearing a face mask and 49% of faces incorrectly wearing a face mask. The dataset is made by generated a virtual face mask on faces. Correctly wearing a face mask means that the face mask is covering the nose, mouth and chin. Incorrectly wearing a face mask means that any of those parts are not covered by a face mask. The dataset is also very recent, it is made available on 2020-11-15 on GitHub with an open license. The images all have the same size 1024x1024. In total, the dataset is 40GBs in size.

# 2 Approach

We are using Jupyter notebooks for our code and final report (beside this document). We plan to use TensorFlow as our deep learning library. The methodology we will be using is derived from the IBM Data Science Methodology [6], starting at the data gathering phase and skipping over the business proposal formulation phase. The reason we skip this phase is because we do not have to make a business proposal and we can define our own challenge.

During the ***data gathering phase***, we want to gather the necessary data from Kaggle, using a Jupyter notebook instead of manually downloading it.

During the ***data preparation phase***, we want to use the TensorFlow data API to learn how to use it, and of course to optimise our model training time. In this phase we also want to research which models we want to use in the modelling phase (the next phase).

In the ***modelling phase***, we will work on building the models and optimising them by using TensorFlow. After the (first) modelling phase we will go over to the ***evaluation phase***, where we will apply XAI to visualise the different choices the models make and decide if the models meet their goal. In case the models do not perform well, we will either go back to the modelling or data preparation phase for a reiteration. In case we go back to the data preparation phase we can search the internet for other datasets to add to our dataset or we could apply techniques such as data augmentation to enrich our dataset. If we decide to go back to the modelling phase, we will work on creating new models or optimising existing models to then further evaluate them.

After the final evaluation phase, we will work on the ***deployment phase***, in which the report will be finalised and the presentation will be created, and if time permits, we will create an interactive demo of our face mask detection model.

## 2.1 Time table

We followed the same time table formed in our data challenge plan (see table 2.1).

| **Week** | **Flyn** | **Mario** |
|---|---|---|
| Week 11 | Plan + request feedback | Plan + request feedback |
| Week 12 | Look into modelling | Creating data preparation pipeline |
| Week 13 | Building deep learning models | Building deep learning models |
| Week 14 | Data augmentation | XAI |
| Week 15 | Previous tasks + request feedback | Previous tasks + request feedback |
| Week 16 | Optimising models + applying feedback | Tuning models + applying feedback |
| Week 17 | Finalising report | Finalising report |
| Week 18 | Spare week + presentation | Presentation (live interactive demo) |

Table 2.1: Planned time table.

# 3 Data gathering

This chapter entails how the used datasets are gathered before the preparation phase.

## 3.1 Kaggle face mask dataset

The Kaggle face mask dataset is hosted on the Kaggle platform. Meaning that there is no need to use advanced solutions like Data Version Control (DVC) [4] for sharing and versioning the dataset. The Kaggle Application Programming Interface (API) can be used to easily download the dataset of face mask images. Afterwards, the unzip command is used to unzip the images into the data folder (see listing 3.1).

```
kaggle datasets download -p data/ -d altaga/facemaskdataset
unzip -n data/facemaskdataset.zip -d data/
```

Listing 3.1: Example code to download and unzip the dataset from the Kaggle api.

As noted in the datasets chapter, the images in the dataset are not the same size (see figure 3.1).

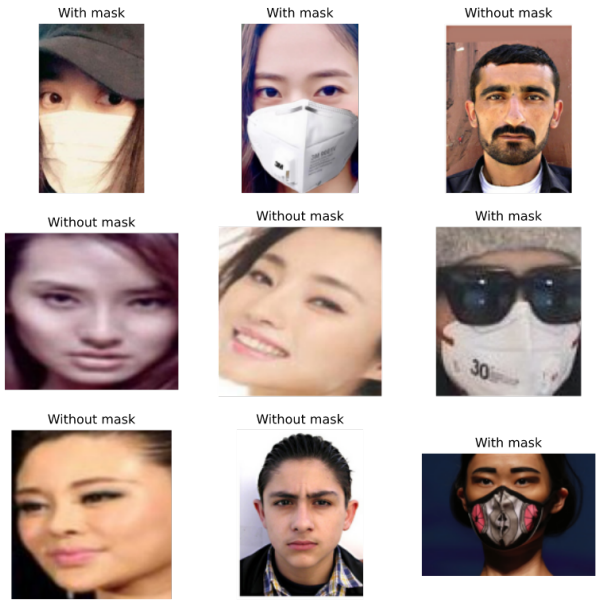


Figure 3.1: Example images from the dataset (not same sizes).

In the data preparation chapter this problem will be tackled.

## 3.2 FaceMaskNet dataset

The FaceMaskNet dataset is hosted on 2 separate OneDrive folders, respectively containing the correctly worn and incorrectly worn face masks datasets. The folders are too big for OneDrive to zip and download them for you, they have to be downloaded in smaller subsets manually. After the dataset is downloaded, the images had to be structured again in two separate folders for the correctly worn and incorrectly worn face mask classes. We could choose to work with DVC here, but we have chosen not to do so, since it will be a one time download for us and we will not be modifying the images for later use.

The dataset is almost equally divided into correctly and incorrectly masked faces. The incorrectly masked faces have been divided into 3 separate categories (see figures 3.2 and 3.3). With 80% of the images having the mask below the nose and covering the mouth and chin. This is because that is the most commonly wrong way to wear a face mask.

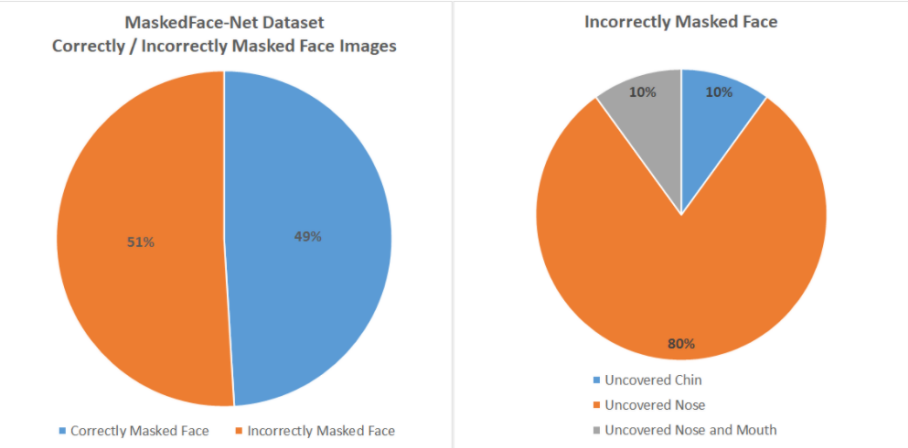


Figure 3.2: Information about the FaceMaskNet dataset.

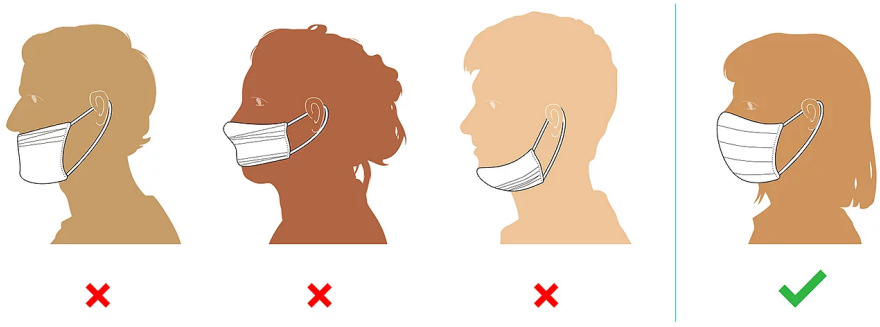

Figure 3.3: All types of correctly and incorrectly masked faces.

# 4 Data preparation

This chapter entails how the used datasets are prepared for the modelling.

## 4.1 Kaggle face mask dataset

As mentioned in the introduction, for this challenge we want to make use of the TensorFlow data API for an efficient data input pipeline. The dataset is created by reading and processing the images by using multiple threads. During the processing of the images, the width and height are scaled to 200 pixels by 200 pixels to fix the problem of different size of images (see figure 4.1).

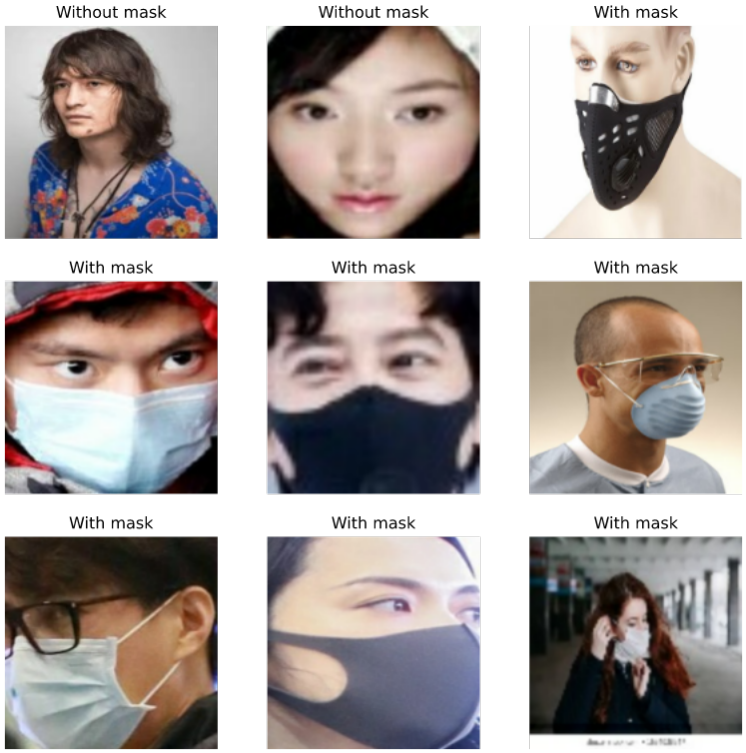


Figure 4.1: After the processing of the images in the data input pipeline, the image sizes are fixed.

The dataset is split into a training set of 60%, a validation set of 20% and a test set of 20%. The TensorFlow data API allows for efficient usage of the dataset by prefetching the next batch of images while the model is training on the current batch. Because of using the TensorFlow data API, the training time is reduced by 30% due to it prefetching the batches of images while training.

## 4.2 FaceMaskNet dataset

The dataset is created by reading and processing the images by using multiple threads. During the processing of the images, the width and height are scaled to 128 pixels by 128 pixels to slim down the dataset for easier usage.


Figure 4.2: Example images from the FaceMaskNet dataset.

The dataset is split into a training set of 60%, a validation set of 20% and a test set of 20%.

# 5 Modelling

This chapter entails the Deep Learning models we have built. To tackle this Deep Learning challenge, we have built a model for each problem. We have made a Convolutional Neural Network (CNN) for both problems: detecting and then judging, which will be shown in the upcoming sections.

## 5.1 Super face mask detection model

To tackle this problem, we experimented with a standalone CNN to see if this would be good enough to solve the problem. To our surprise, the model reached an accuracy of roughly 95% on the validation set. See figure 5.1 for the training history.

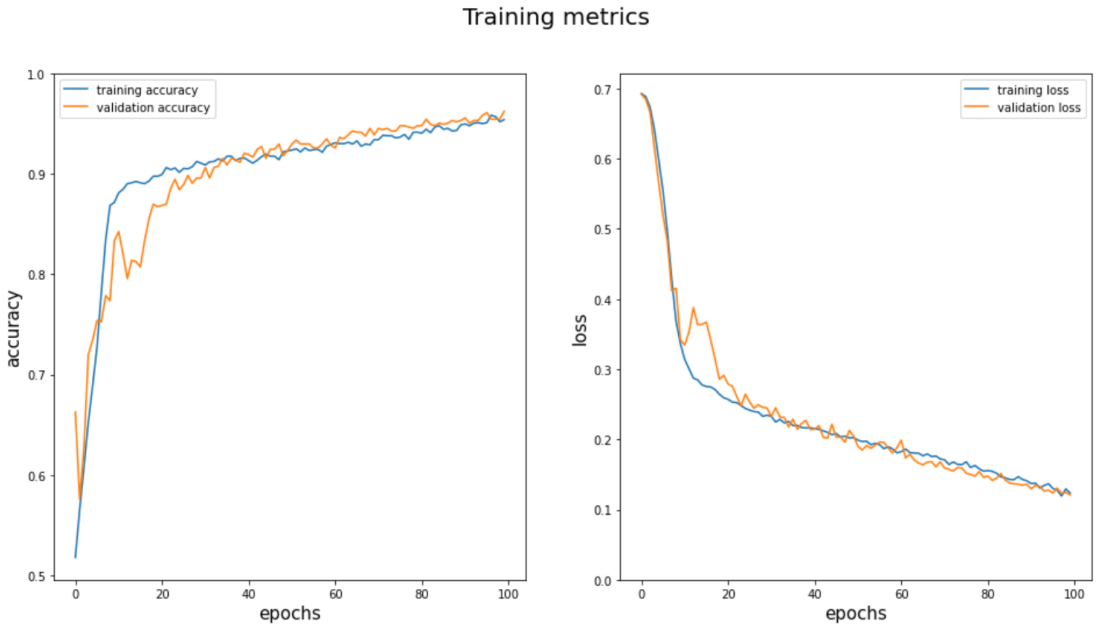


Figure 5.1: Training metrics of the model's training history

The core of the model we have built consists of three convolutional blocks with a max pool and dropout layer in each of them. The dropout layers are added to prevent overfitting. Followed by a flattening layer, to flatten the feature maps to a one dimensional array. Afterwards, a dense layer with 128 dense units with the Rectified Linear Unit (ReLU) activation function. Then, another dropout layer has been added. Lastly, another dense layer with a singular dense unit has been added which converts the output to a probability (using the sigmoid activation function) whether the person in the image is wearing a mask. The full architecture that was used to build this model can be found in figure 5.2

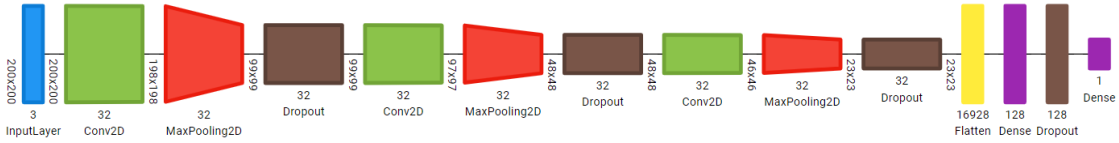


Figure 5.2: Architecture of our detection CNN model. (Visualisation by Net2vis [2]).

## 5.2 Omega face mask judger model

To start with detecting whether a person is correctly wearing a face mask, we have used a similar CNN architecture as the detection model. The model was able to converge really fast because of the large amount of training data. The model reached an accuracy of 99.8% on the validation set. See figure 5.3 for the training history.

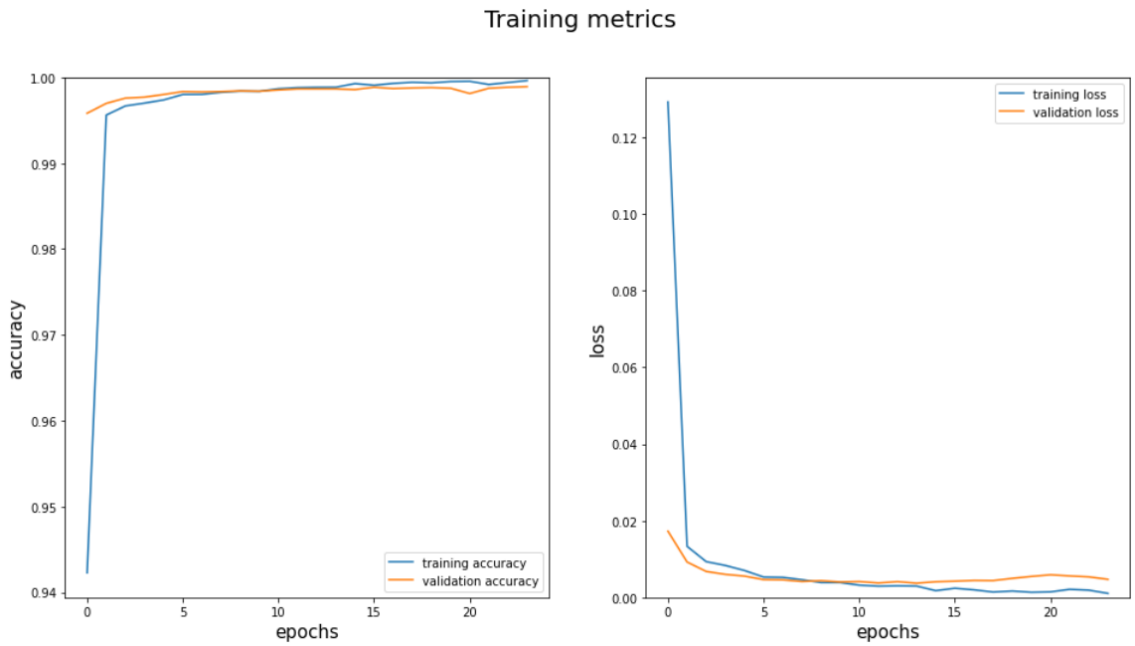


Figure 5.3: Training metrics of the judging model's training history

The core of the model we have built consists of three convolutional blocks with a max pool and dropout layer in each of them. The dropout layers are added to prevent overfitting. Followed by a flattening layer, to flatten the feature maps to a one dimensional array. Afterwards, a dense layer with 128 dense units with the ReLU activation function. Then, another dropout layer has been added. Lastly, another dense layer with a singular dense unit has been added which converts the output to a probability (using the sigmoid activation function) whether the person in the image is wearing a mask correctly. The full architecture that was used to build this model can be found in figure 5.4

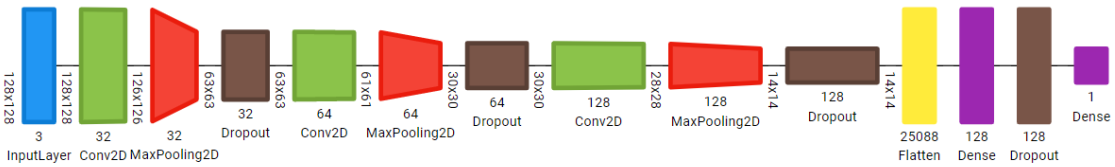


Figure 5.4: Architecture of our judge CNN model. (Visualisation by Net2vis [2]).

# 6 Evaluation

This chapter entails the evaluation of the Deep Learning models and the performance metrics.

## 6.1 Classifying pictures of ourselves

We have confirmed the accuracy of our model using images of ourselves with and without masks. The model was 100% accurate in classifying our images. See figure 6.1 for the results.


Figure 6.1: Model predictions on our own images. Green text means correctly classified, red means incorrectly classified.

## 6.2 Explainable Artificial Intelligence

To explain how our face mask detection model works, we have used Grad-CAM [7] to visualise the activations within the model. We were interested to see how the model responds to different poses of the face in the image input (see figure 6.2).

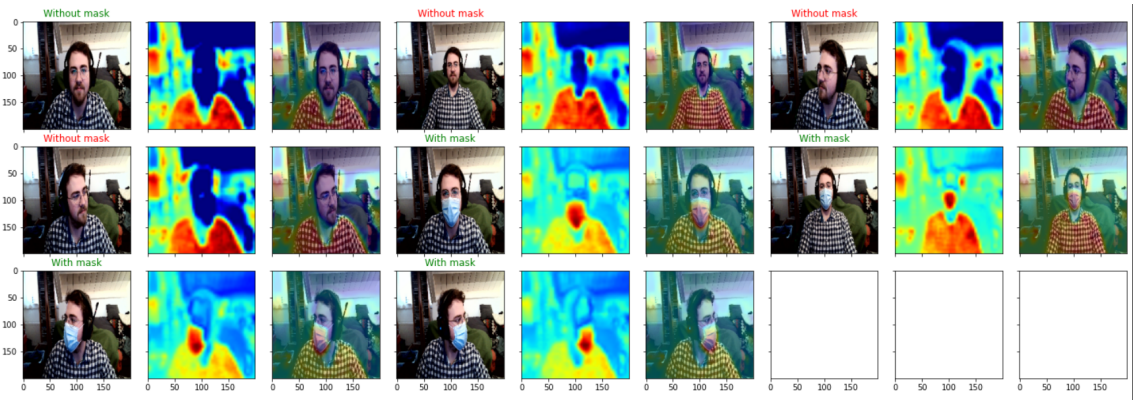


Figure 6.2: Activations on several poses: close-up, far-away, looking left and looking right with and without face mask on. Green text means correctly classified, red text means incorrectly classified.

Two out of the eight poses were incorrectly classified. Let us take a closer look at the first incorrectly classified pose (see figure 6.3).

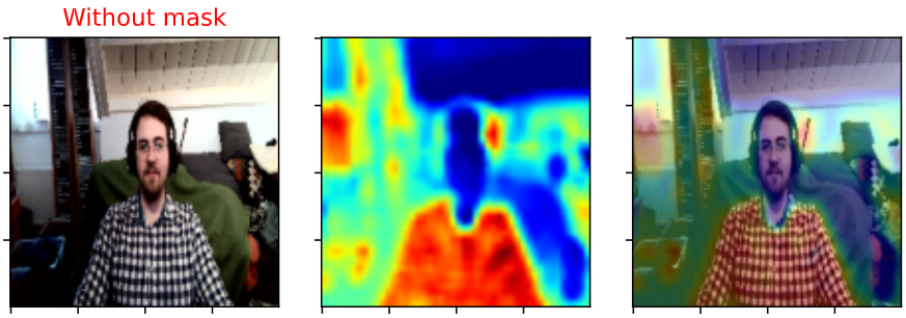


Figure 6.3: Incorrectly classified pose far-away without face mask.

The blouse Mario is wearing, is the most interesting part to look at according to the activations of the model. It has the darkest red colors, meaning it has a high activation there. Based of the other inputs, it seems that the model activates on lighter colors. This could be because the model has learned that these masks are mostly light blue or white, which explains the activations on the blouse and the wall in the left top of the pictures. And, our initial idea is that Mario's face is not big enough in this image to recognise it as a face. If we take a look at the training data (see figure 4.1), most of the images are zoomed in on the face. This could indicate that the model only properly works on close-ups of faces. To confirm that idea, we have used these input images again but zoomed in on Mario's face (see figure 6.4).

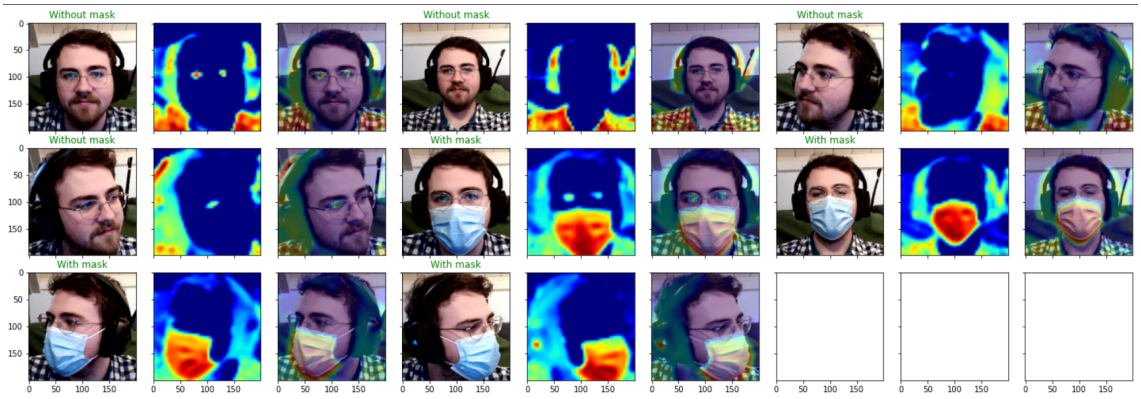


Figure 6.4: Activations on several poses zoomed-in: close-up, far-away, looking left and looking right with and without face mask on. Green text means correctly classified, red text means incorrectly classified.

The model now predicts all poses correctly. To confirm whether the model also works with darker colored face masks, the same poses are taken with a dark face mask worn. (see figure 6.5).

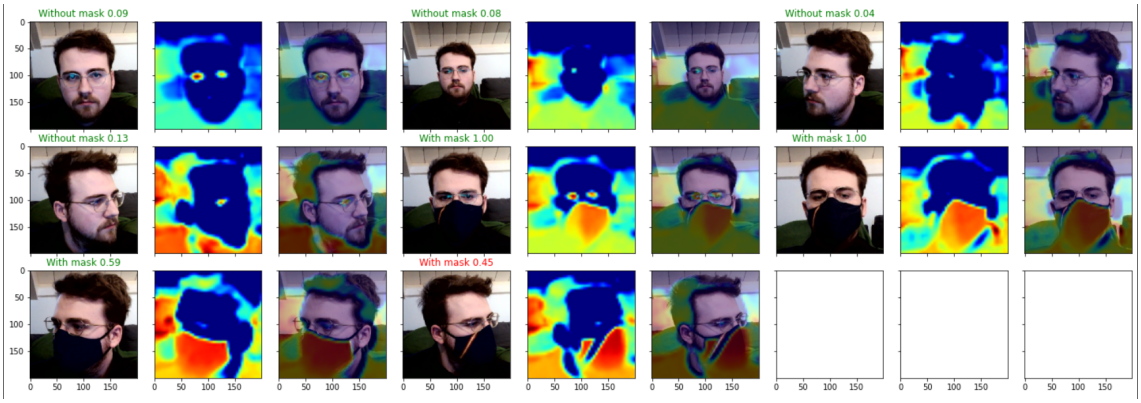


Figure 6.5: Activations on several poses zoomed-in (dark face mask): close-up, far-away, looking left and looking right with and without face mask on. Green text means correctly classified, red text means incorrectly classified.

The model still activates on the darker colors and manages to classify every pose but the last one correctly. The mask has a colored stripe on the right side of the mask causing the model to be thrown off. Also, an interesting thing to note is that the Mario's skin does not get activated on but his eyes/glasses sometimes do because of light color reflections. This could indicate that the presence of high activations around the face could indicate that a face mask is present. Based on these tests, we can say that the model is not great at detecting face masks on people far away in the image but is pretty good at close ups. Should this model be used, then the input of the model should only be the face for better performance.

## 6.3 Performance metrics

The accuracy of the face mask detection model on the test set was 92.71%. A confusion matrix is plotted to see whether there are a lot more false positives or false negatives (see figure 6.6). The model is almost 3 times more likely to have a wrong prediction when wearing a mask.

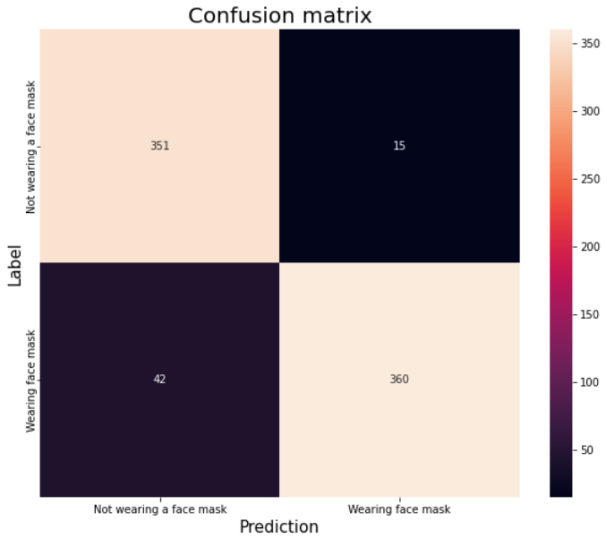


Figure 6.6: Face mask detection model confusion matrix.

The accuracy of the face mask judging model on the test set was 99.89%. A confusion matrix is plotted to see whether there are a lot more false positives or false negatives (see figure 6.7). Because of the large amount of correct predictions, the false positive and false negatives are negligible.

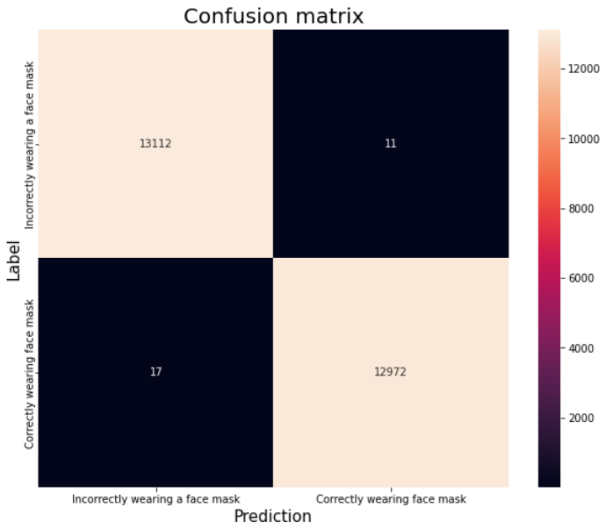


Figure 6.7: Face mask judging model confusion matrix.

For both models, we have ensured that the same seeds are used as during training ensuring that the test set does not include any data from the training or validation sets.

# 7 Interactive Demo

This chapter will show the steps taken to implement the interactive demo.

## 7.1 General workflow

Our first step was defining what we wanted to showcase in the demo. Since we have implemented two different types of models in our modelling phase (a mask detection model and a mask judge model), it would be interesting if we could use both during the demo. We came up with the idea to first detect if a face is on the screen, then perform face mask detection, then perform face mask judging (displaying if the mask if correctly worn). See Figure 7.1 for a visualisation of the general workflow.

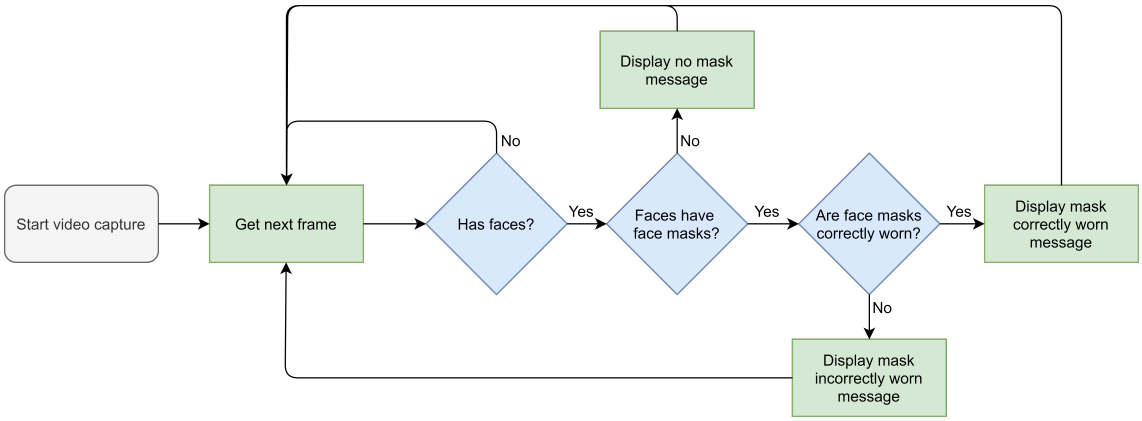


Figure 7.1: Visualisation of the general workflow, details are purposefully left out.

The steps showcased in Figure 7.1 explained in further details:

1. Start video capturing. Since the assumption of this project is the video capture is a continuous process, an end point is not included in the visualisation. However, the software does have this built in, in case it would be needed.
2. Get the next frame.
3. Perform face detection on the frame and collect the x,y,w,h values of every face, in case no faces are detected go to step 2.
4. For every face, crop the face out of the frame and perform face mask detection. For every face that does not wear a mask, display a no mask message (note: every message is displayed above the current face). If none of the faces wear a mask, go to step 2.
5. For every face with a face mask, perform face mask judging. For every incorrectly worn face mask, display an incorrectly worn mask message. For every correctly worn face mask, display a correctly worn mask message. Finally, go to step 2 to repeat the cycle.

## 7.2 Used models

The following models were used to built the interactive demo:

| Step (workflow) | Model | Creator |
|---|---|---|
| 3 | Face detection model [8] | OpenCV |
| 4 | Super face mask detection model | Ours |
| 5 | Omega face mask judger model | Ours |

Table 7.1: All implementations used in the interactive demo ordered by step

Information on the OpenCV model: Object Detection using Haar feature-based cascade classifiers is an effective object detection method. It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects (in our case, faces) in other images. There are many haarcascades available, for our problem we have picked the frontal-face model.

## 7.3 Examples

To verify that our interactive demo is working correctly on all stages, we have manually tested this and took some screenshots of the latest testing phase. See Figure 7.2 for the results.

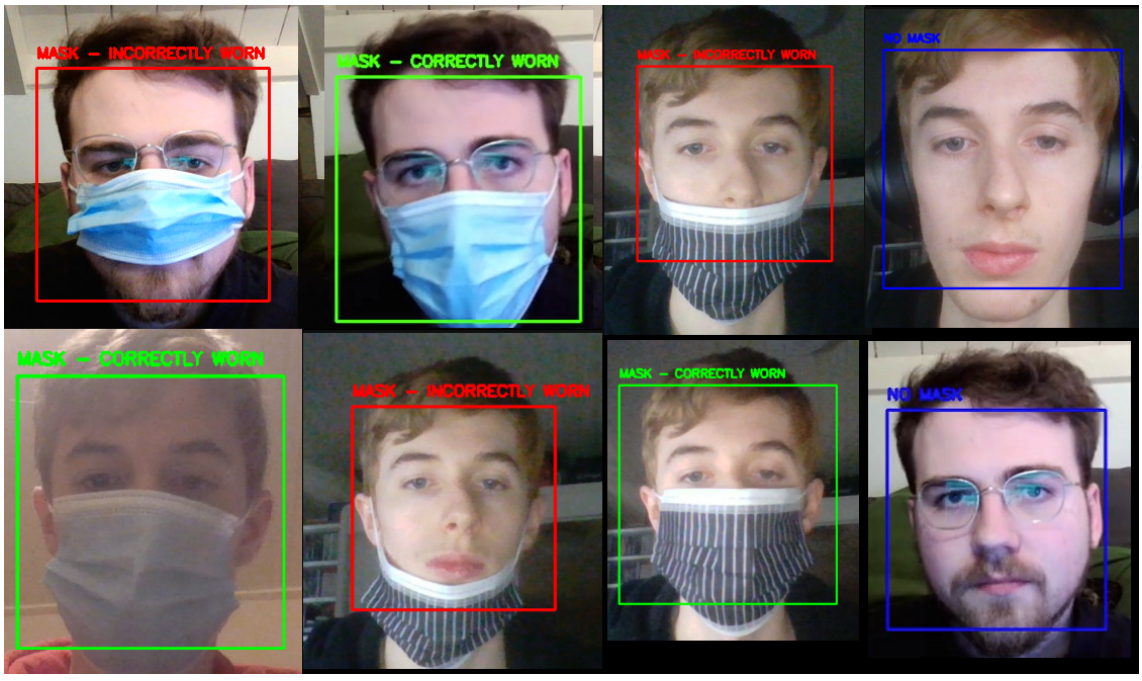


Figure 7.2: Example predictions of the latest testing phase

Since the demo is supposed to work for multiple faces, we have verified this using two people, the results can be seen in Figure 7.3.

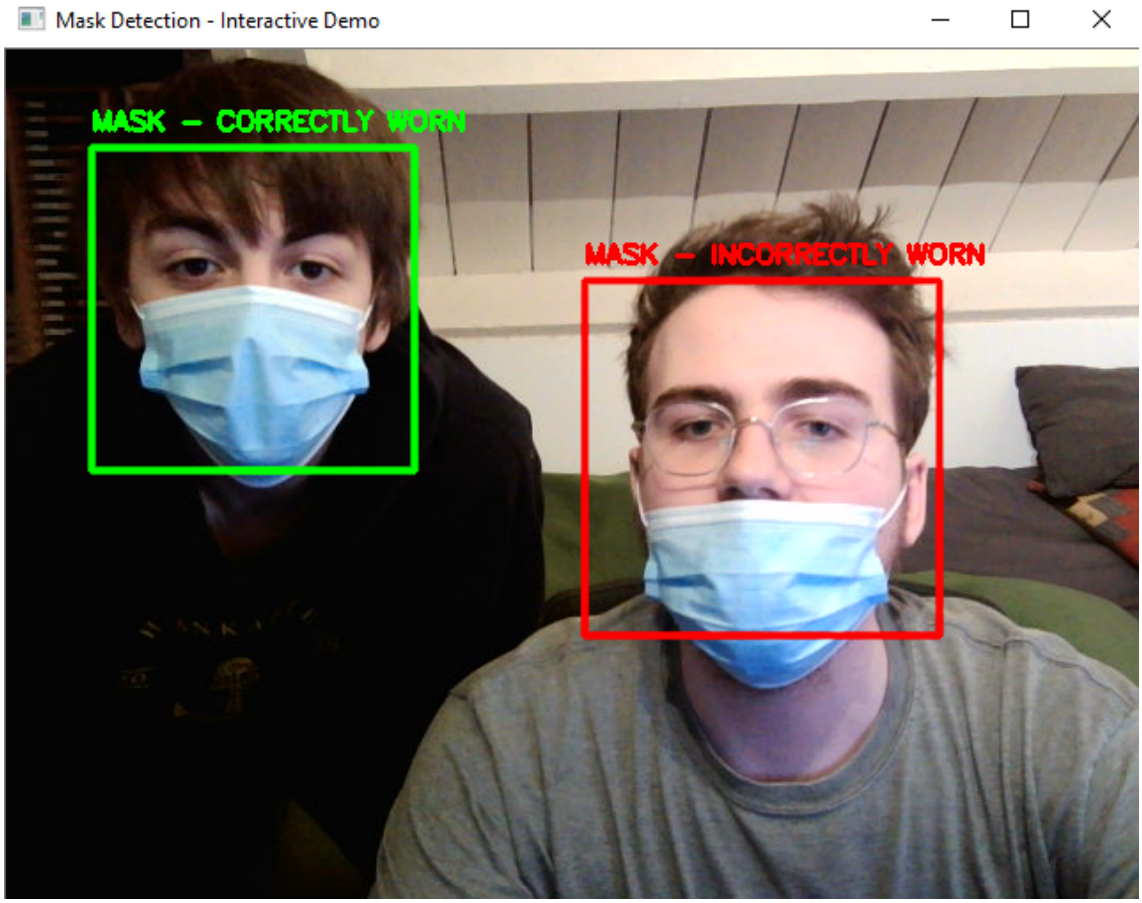


Figure 7.3: Example predictions with two people

# 8 Feedback log

## 8.1 Feedback on plan (Bartosz) 20-11-2020

Bartosz has given us feedback on our plan, telling us we have a clear path and a clear way how we want to approach our challenge. The only note was, that we did not tell that we were following the IBM Data Science Methodology, but the rest of the plan was really nice. Shortly after, we have applied the feedback, including adding the IBM Data Science Methodology in our approach.

## 8.2 Feedback meeting (Simona) 03-12-2020

We had a meeting with Simona, showing her our work so far. She gave a few insightful points which we will apply in the next version:

- She liked our determination but said certain points were maybe too ambitious, like GANs for data augmentation.
- If applying XAI turns out to be too difficult she told us to consider skipping it.
- An interactive demo is a nice addition because it gives us something to show at the end of the challenge.
- She liked our idea of expanding the challenge to face mask detection that also checks if the masks are being worn correctly (when detected).

## 8.3 Feedback meeting (Simona) 07-01-2021

A final meeting with Simona was held to conclude our work so far. She complimented us on all the work we have done, adding new data, another model, successfully applying XAI, and making the interactive demo. One thing she found a bit suspicious was the near 100% accuracy of the judge model on the test set. We believe this is because the task is relatively easy for the model and we have a lot of data. We will double check the test set to make sure that these metrics are reliable and update them if necessary.

# 9 Conclusion

The main goal of this project was to reach an accuracy of 80% on the tasks at hand, but our models were able to reach an accuracy of 95% for the detection model, and an accuracy over 99% for the judge model. Which far surpasses the goal we have initially set. We met those goals by making efficient use of the TF data API, creating a data preparation pipeline for the face mask datasets. Another goal was to predict on images of ourselves, which we were able to do successfully using our models. To visualise the choices the models make, XAI has been applied. The insight we gained was that our detection model performs better on close-ups of faces, since this is what they were trained on. Finally, we built an interactive demo, in which we have incorporated the insight from applying XAI on our detection model. This demo has been made to showcase the models in action, showing that the detection and judgement models are able to be applied on a real-time feed from a webcam.

## 9.1 Reflection Mario

I have learned to work with the TF data API to efficiently process the images for the training the model. The XAI on the detection model was really interesting to see where the activations were. Based on the XAI results, we were able to make decisions for building the interactive demo a lot better. I was surprised by how fast the models worked in classifying the images for the interactive demo. Overall, I am very happy with the results we have come up with and I had a lot of fun working on this data challenge together with Flyn.

## 9.2 Reflection Flyn

I have learned that a CNN can be very accurate in detecting and judging face masks (given the right data). I have learned to work a with OpenCV and how to string multiple models together into a pipeline capable of solving our problems. (See Interactive Demo.) Furthermore, I found it interesting to see Mario was able to apply XAI successfully, to see the actions of the models. Finally, I had a lot of fun working on this challenge together with Mario and I have learned many new things.

# Bibliography


[1] Victor Alta. Kaggle facemask dataset, 10 2020. https://www.kaggle.com/altaga/facemaskdataset. 6

[2] Alex Bäuerle and Timo Ropinski. Net2vis: Transforming deep convolutional networks into publication-ready visualizations. *CoRR*, abs/1902.04394, 2019. 12, 13

[3] Adnane Cabani, Karim Hammoudi, Halim Benhabiles, and Mahmoud Melkemi. Maskedface-net – a dataset of correctly/incorrectly masked face images in the context of covid-19. *Smart Health*, 2020. 6

[4] Ruslan Kuprieiev et al. Dvc: Data version control - git for data & models, November 2020. 8

[5] Google. Tensorflow data api. https://www.tensorflow.org/api_docs/python/tf/data. 6

[6] IBM. CRISP-DM, data science methodology. https://www.ibmbigdatahub.com/blog/why-we-need-methodology-data-science. 7

[7] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. *International Journal of Computer Vision*, 128(2):336–359, Oct 2019. 15

[8] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. volume 1, pages I–511, 02 2001. 19