Automated face-mask detection

Applying Machine Learning image recognition to detect wearing of face-masks

Fabian Metz
f.metz@mpp.hertie-school.org

Tobias Palmowski
t.palmowski@mpp.hertie-school.org

Thilo Sander
t.sander@mpp.hertie-school.org

Abstract

The Covid-19 Pandemic invigorated research into the issue of face-mask detection on images. Motivated by either the goal of better enforcement of mask mandates or addressing the problems masks pose for facial-recognition software, many recent contributions on this topic focus on face-detection using deep learning approaches. We aim to contribute to the scientific work by extending the scarce research on the performance of conventional Machine Learning algorithms. Therefore, we investigated the performance of different Machine Leaning algorithms for image classification. Performance in the scope of this study is not limited to accuracy but also includes testing and tuning prediction speed as it is impacting usability, i.e. for applications that aid in monitoring the adherence to mask mandates. Testing our initial set of classifiers yields very good result in terms of accuracy (abvoe 99 % across the board) but are probably tainted, as our dataset is not representative of the real-world.

The `GitHub`*-repository can be found in the footnote*[1].

Preface

As the acquisition of suitable data sets has proven to be impossible due to uncertainties regarding legal procedures at Hertie, we had to switch the project topic on short-notice. We are using this preface to give a brief overview about the motivation, the background and the contribution of our new project. It is meant to cover the main points of the project proposal in order to provide the grounding of our idea. We do not consider this preface as part of the midterm report grading.

[1]The GitHub repository can be found at `https://github.com/the-palmo/hertie-ml-project_face-mask-detection`

Motivation The Covid-19 Pandemic shifted the focus towards new research questions that have not been focused on before. One of these newly arising questions concerns wearing face-masks. Being recommended by the WHO as an effective measure against the spreading of Covid-19 masks are an essential part of national pandemic response strategies around the world [12]. They are mandated in grocery stores, at schools, office buildings, theaters and public places. These public places however lack an easy way of control and enforcement of mask wearing.

Here, automated face-mask detection can play an important role. Using cameras and algorithms to detect whether people do or do not wear a (medical) mask correctly can have a number of use cases in different settings.

Self-control At the entrance of for example supermarkets, book stores, stadiums and theaters, cameras and monitors could give you a visual feedback, whether you are wearing your mask correctly or even if you wear the right type of mask (f.e. medical one).

Group Control In settings where an automated entry system is already in place (e.g. soccer stadiums or concert halls) additional cameras could be used to take images and detect not correctly worn masks of individual people, being a help to enforce mask mandates. This could cause a notification to the staff that then might control for incorrect predictions by the algorithm. This will help to accelerate the entry process without rely on large numbers of support staff controlling the mask mandate.

Peer Control Such technologies could also be used in places like stadiums and public transport combined with public visibility. Using monitors to show and flag the persons that don't wear a mask could lead to peer/ public pressure to adhere to the rules.

These three use cases are certainly debatable and the question whether society wants to introduce more surveillance

for such use cases is one to be answered politically, but this it out of scope of this paper. We believe that there are ethical use cases in the realm of self- and group control, especially if the technology is distributed publicly and employed locally.

Background/ Related Work The problems connected with mask wearing are twofold: On the one hand, masks cause a problem with facial recognition software as they are obscuring parts of the faces which poses a challenge to many existing facial recognition algorithms [3]. On the other hand, there is the problem of enforcing compliance to mask mandates. Creating a face-mask detection software can help with both of these problems. For the problem of adherence to mask mandates, the benefit works via (self) enforcement. Regarding the problem of facial recognition software, the classification of an image as "including a masked face" might help address this challenge.

The task of face-mask detection in not entirely new to the research world. Even before COVID-19, solutions were proposed on how to detect whether people are wearing face-masks. In light of the pandemic however the focus on this question intensified leading to more solutions: Most of the proposed solutions to this problem have been based on Convolutional Neural Networks (CNN) and related deep learning methods [4, 6, 10]. Contributions addressing the effectiveness of conventional machine learning classifiers remains scarce. One recent study compared classification based on the CNN Mobile-net with conventional K-nearest neighbour and support-vector machine classifiers. This study found the results of the CNN to outperform the conventional classifiers by 5% percent (90% vs 95%) [9].

Our Contribution As discussed above, there are some approaches to solve the problem publicly available, but public code remains scarce and the proposed solutions largely draw on deep learning approaches [11]. Furthermore where conventional Machine Learning (ML) is addressed, the selection of methods appears arbitrary, the datasets that are used are small (apprx. 3200 images in [9]) and the code is not publicly available. We thus aim to extend the existing approaches by building solutions using conventional Machine Learning classifiers.

Thereby, we add value in three ways:

1. By providing public code that can be employed and advanced locally with relative ease.

2. By focusing on prediction speed to enable real time application.

3. By extending literature on ML classifiers on face-mask detection.

We imagine a use case for such a solution in the realm of monitoring adherence to mask mandates, for example when entering a conference or panel at Hertie. Most importantly we believe that this project provides us with ample learning opportunities, regarding both the theoretical perspective on the classification task and the hands on data wrangling and coding of the final project.

The Scope The minimum we want to achieve in scope is a working classifier that outperforms our initial baseline [see below]. In order to be valuable it would probably need accuracy of around 99% and arguably be even less forgiving regarding false positives. Creating a multi-class classifier that is able to distinguish different types of (wrongly) worn masks and has improved prediction and training speed would be considered a valuable addition.

1. Proposed Method

To address our research problem, we employ the following process.

1. Testing different ML algorithms without hypterparameter tuning
2. Establishing the baseline with the best performing classifier(s)
3. Tuning Phase I : Accuracy (by altering hypterparameters)
4. Tuning Phase II : Prediction Speed (by altering pixel resolution (i.e. number of features)

Testing different ML algorithms without hyperparameter tuning In line with common research practice, we argue that it is worthwhile to test different algorithms to establish our baseline and find the candidate(s) for tuning. The selection of these initial algorithms is based on the following criteria: the type of data, the size of the dataset, the accuracy-speed trade-off, the explainability, and the prior use for face-mask detection in literature.

As we use labeled data, the algorithms to be chosen are all classifiers. In line with the "No Free Lunch Theorem", we are aware that possibly different classifiers perform better regarding one criterion and worse for others. Thus, we include classifiers in our initial set that are likely performing well on at least one of the criteria. Furthermore, we plan to include classifiers based on Support Vector Machines and K-Nearest-Neighbours in our initial set as they were used in the scarce existing publications on ML-classification performance regarding face-mask detection. This approach will allow us to comprehensively search for the best performing candidate for further tuning.

The other classifiers are chosen due to the following reasons: For datasets larger than 100 000 using the stochastic gradient descent is advised by sci-kit learn [8]. Regarding the accuracy-speed trade-off, random forest is good in predicting accuracy, while decision trees, logistic regression and linear support vector machines are better in speed [5]. We include these different classifiers in the initial phase to get a glimpse whether speed or accuracy is the larger problem in our project. Regarding explainability, we want to include classifiers that are no mere "blackboxes" in order to better comprehend the ML process and check for the origin of potential biases.

Based on fitting each of our criteria with at least one classifier, our initial set of conventional ML classifiers are therefore the following ones: *Random Forest, Decision Trees, Stochastic Gradient Descent, Logistic Regression, K-Nearest-Neighbours, Support Vector Machines*

This initial set of classifiers is then trained and evaluated, in terms of accuracy and training time. The findings and test configurations are specified in Section 2. After this first evaluation the suitability of the algorithms and the data is analyzed and possible steps for refinements are derived.

Establishing the baseline with the best performing classifier(s) After running the six different classifiers without hypterparameter tuning, we select our baseline as follows. We evaluate the results for accuracy and training time. We are dropping classifiers that are exceptionally low in accuracy or have an exceptionally long training time compared to the other classifiers. For the remaining classifiers, we are creating an average of the accuracy as the main the main performance indicator of our baseline.

Tuning Phase I: Accuracy In the first phase of our tuning, we are optimizing our most promising classifier(s) regarding accuracy.

Tuning Phase II: Speed In the second phase of our tuning, we are tuning our algorithm on prediction speed, altering the number of features via resizing the pixel ratios of the individual pictures. Tuning on speed will negatively effect accuracy as typically a higher accuracy needs more time to be reached. In our project, the decision will be based on our primary application - monitoring mask mandate compliance at a Hertie event.

2. Experiments

Data In order to conduct the project described above, we rely on labelled datasets that include many images of people wearing mask correctly and many images of people not wearing any mask or at least not correctly (e.g. not covering the nose). Luckily, Cabani et al. published such a dataset in March 2021 in the Journal `Smart Health` [2] and made the links available via `GitHub` [1].

The authors rely on the dataset of face images Flickr-Faces-HQ3 (FFHQ), publicly made available online by `NVIDIA Corporation` [7]. The pictures are crawled from `Flickr` when they had the appropriate license (Creative Commons or Public Domain). The authors emphasize that the original FFHQ dataset contains a considerable variation in terms of age, ethnicity and image background and a superficial look into the pictures seem to validate that statement. However, as the pictures are crawled from `Flickr` the data will inherit all biases from the platform, e.g. the number of older people or people from poor countries might not be representative. However, as we focus on image recognition and face-mask detection without relying

on facial recognition software, the bias should have only a minor effect on performance.

Cabani et al. apply face recognition software to set several landmarks in the pictures where they subsequently place a virtually inserted face-mask. The result is an dataset containing 66,948 pictures with correctly worn artificial face-masks (see example in figure 1a) and 66,634 pictures with incorrectly worn artificial face-masks (see example in figure 1b).[2] As you can see, pictures from the same persons are used once for a correct worn image and once for an incorrect worn image.

(a) Correctly worn mask (b) Incorrectly worn mask

Figure 1: Example pictures of data set

Data wrangling In order to get machine interpretable data the images of the dataset have to be preprocessed. The raw data is organized in two separate folders, one for the correctly worn face-mask pictures and one for the incorrectly worn face-mask pictures. The pictures are stored as `.jpg`-files in several numerated subfolders, all with a resolution of 1024x1024 pixels. The data wrangling is performed in the first Jupyter Notebook[3], as we wanted to separate this one time operation from the process of model tuning that occurs later in the project process. Next to rather small challenges (like how to deal with `.jpg`-files in Python code at all, how to iterate through inconsistently numbered files and defining a label vector) several data wrangling issues were sorted out in this first Jupyter Notebook that are addressed in the following:

First of all, creating a numeric representation of the image files is key for running a Machine Learning algorithm. The package `NumPy` can extract all numerical information stored in the color scheme (RGB in our case) by just applying the array method to a picture opened and stored in a variable.

In order to handle the amount of data (the `.jpg`-files amount to ca. 41 GB of data) and get to a reasonable speed of processing, a pixel reduction is necessary. If all 1024X1024 pixels with the 3 RGB values were used the resulting dataset would have more than 3.1 million features and six pictures with the combined size of 1.8 MB would be translated to a machine interpretable representation of 151 MB - a factor of 84. As such a high resolution does not seem necessary for good performance, we decided to reshape all pictures to a starting resolution of 32x32 pixels. We will explore the accuracy and run-time performance at a later stage of the project.

As the data set is already quite large in terms of disk space the data wrangling needs to be run-time efficient. The target is an `Numpy` array where every row contains a 1D array with all RGB values as features (like the `MNIST` dataset discussed in class). The normal go-to methods `numpy.append`, `numpy.concatenate` and `list.append` produce an array where all RGB values of all pictures are stored in a 1D-array. In the end, the problem is solved with the `NumPy` method `vstack`. To use this method properly the `NumPy` array containing the numerical representation of the pictures need to be initialized with the proper number of columns right away and not as an empty array - which of course needs more run-time. However, if an empty array is used, the first stacking needs to be a horizontal one with the `NumPy` method `hstack` before using `vstack` from the second iteration on. Thus, in every for-loop iteration it would be necessary to evaluate an if-clause every time, significantly increasing run-time when large data sets are used.

The last and final point is how to save the transformed data set so that another Jupyter Notebook can access the numerical representation of the pictures without having to run through the data wrangling process every time (which needs roughly 8 hours for the 32x32 pixel configuration). We decided to organize the numerical data of the pictures together with the corresponding label array into a dictionary. This dictionary is then stored as a `.pkl`-file using the Python package `pickle`.

The result of the data wrangling process is a dictionary containing two arrays: One array of the numerical representation of the pictures containing all RGB-values at every pixel. With the starting resolution this amounts to array of shape (133582, 3072). 133582 is the number of observations (i.e. correct plus incorrect pictures) and 3072 is the number of features included (i.e. 32*32 pixels times 3 for the RGB representation). Additionally, the dictionary contains another array with the corresponding labels. This array is a 1D array of shape (133582,) where every observation has exactly one label.

Evaluation method For the evaluation of our classifiers, we

[2] In the abstract of the published paper the authors speak of roughly 3,350 pictures more, but the data set on GitHub is constantly cleaned where the face detection algorithm produced wrong results.

[3] hertie-ml-project_face-mask-detection/01_Data-Collection-Processing.ipynb

use the following metrics: *the cross validation score, the confusion matrix, the precision & recall scores, and training & prediction speed.*

To check the overall accuracy of our classifiers, we use the **cross validation score**. With these outputs, we are getting a first glimpse of how good the individual classifiers perform on different folds of the training set. This allows us to check for overfitting.

For the confusion matrix and the precision & recall scores, we predict the labels based on different folds of our training data using `cross-val_predict`. Then we compare them to the actual vector of labels of the respective test folds. Using this way of predicting, we get initial accuracy measures without using our test set.

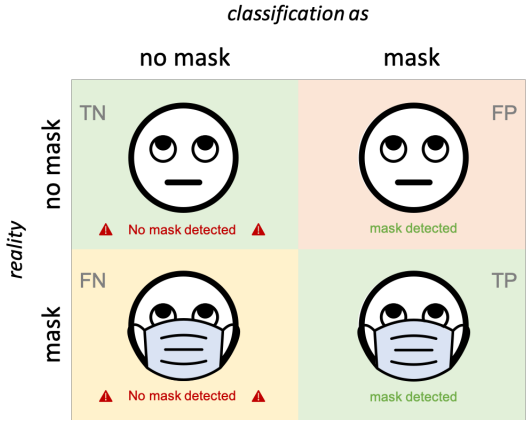

Figure 2: graphical representation of the confusion matrix

The **confusion matrix** is used to give a graphical representation of how well the classifiers are predicting the true state. Regarding our use case, next to general accuracy, a very low number of false positives (FP in figure 2) is especially important to increase the safety for all attending individuals. As our classifier would still rely on a human for enforcement, false negative classifications could easily be correct by the human, seeing that a mask actually worn correctly. This only causes short delays, but no health security issues. Therefore, a desired outcome would prioritize a lower number of false negatives against a lower number of false positives.

The **precision & recall scores** are the scores which determine the share of true positives in all instances classified as positive (precision) and in all instances that in reality positive (recall). In line with the description of the importance of the low number of false positives over the low number of false negatives, a higher precision score is more important than a higher recall score regarding the quality of our outcome. Accordingly, we are not using an F1-score to combine these two scores, as it disregards our considerations regarding the "preferred" error.

$$precision = \frac{TP}{TP + FP} \qquad recall = \frac{TP}{TP + FN}$$

Besides these general ML evaluation metrics, there are two more metrics we use to measure classifier performance in our project: training time & prediction speed.

A long **training time** is problematic for three reasons. First, in the scope of this project we have to work with limited resources in terms of computational power and time. Thus, tuning a model over several iterations can quickly exceed our capacities, if the model is computationally too demanding. Second, constantly digesting the feedback regarding FP and FN classifications in from of an online or mini-batch learning is in our use case very desirable to improve the algorithm further. Thus, the training time have to be reasonable long to train the algorithm for example every night for a couple of hours. Third, it is generally an advantage if an algorithm can be trained locally and applied with ease to different datasets/ surroundings. A short training time will make it easier for other people to test and further develop our solution.

A slow prediction speed is even more problematic as it would directly inhibit our envisioned main use case, i.e. detecting the compliance to mask mandates in real time, which needs predictions within about less than a second for reasons of usability. This evaluation metric however has not been included in our first experiment but will be implemented along other features later (see section 3).

Experimental details For establishing our baseline code, we used a toy dataset with 200 pictures, that we extracted from the whole dataframe at the beginning. We did this to check and develop our code in shorter time periods, as working with the whole dataset already takes about half an hour to train and evaluate individual classifiers. We provide this toy dataset together with our code so that it can be run and checked without the need of downloading an excessive amount of data.

The first thing after data wrangling in our pipeline is the split into train and test dataset with a test size of 10%. We consider ca. 13,300 instances as reasonable for testing the performance. After that, we trained and evaluated our six mainly untuned classifiers to establish a baseline and evaluate the intial performance of each classifier. The only tuning made are the setting consistent random states and the setting of higher maximum iterations in cases, where convergence could not be reached with the preset value of 1000. Unfortunately, the K-Nearest-Neighbor and Logistic Regression algorithms did not finished as the kernel died several times

	RandomForestClassifier	LinearSVC	DecisionTreeClassifier	SGDClassifier
precision score	0.997	0.995	0.993	0.996
recall score	0.998	0.996	0.994	0.997
cross validation scores	[0.998, 0.997, 0.998]	[0.996, 0.995, 0.995]	[0.994, 0.993, 0.993]	[0.997, 0.996, 0.997]
cross validation mean	0.998	0.995	0.993	0.997
cross validation std	0.00047	0.00047	0.00047	0.00047
run time in seconds	1720.01	566.44	2365.89	213.26

Figure 3: Results

before finishing. Thus, we only report the results of the other four algorithms.

For the evaluation of the classifiers we used a cross validation prediction with 3 folds in our training dataset for the confusion matrix as well as the precision & recall scores. The cross validation score was specified with 3 folds as well. Moreover, we tracked the time needed for training and evaluating each classifier. This serves as an initial proxy for training time. In a later stage of the project we will decouple training and evaluation time. (see section 3)

Results

For a comparison of accuracy measures of our initial classifiers running in a (close to) standard configuration (see figure 3). The classifiers only vary marginally regarding accuracy, with cross validation, precision and recall scores above 99% across the board. Regarding training time there are larger discrepancies between the classifiers with some running for over half an hour while others are finished within 5 Minutes (see figure 3). For an overview of the amount of false positives and false negatives of the two classifiers yielding the highest precision scores, we added the confusion matrices in Appendix A.

Commenting on quantitative results Our quantitative results were way better than we initially expected. We believe that these results hint at an issue with the artificially constructed dataset. As it only contains variations of the same artificially inserted blue medical masks on portrait photos it appears to be very easy for the classifiers to detect a large blue area in the lower part of the image.

Due to these ”too good to be true” accuracy results, including other data into our project should be the next logical step. Including different masks and real world pictures, with possibly different angles, we might get more relevant accuracies and can train a valuable algorithm.

Regarding training time there are larger discrepancies as stated above. Based on our proposed approach for setting the baseline, we will drop the classifiers with an exceptionally long training time. However, the concerns regarding the suitability of our dataset we decided to wait until we have results on more refined data.

3. Future Work

Given the time constraints that followed our struggles acquiring data and the subsequent refocusing of our project, a lot remains to be done. This section provides an overview of the main challenges and milestones laying ahead:

More/ real world data & establishing a baseline As stated above, we have substantive doubts regarding the accuracy of our classifiers in a real world setting, with different masks / pictures. Thus, we plan to train our classifiers with other data sets that contain less but more appropriate data to re-establish a suitable baseline and choose our best performing candidates for tuning. [4]

Tuning accuracy & speed After choosing the best performing candidates we will focus on the two tuning phases. We did not yet touch upon one major goal of our project which is the assessment of time needed for a one image classification. We have to set up the code to measure this feature and assess the impact of image size on the accuracy-speed trade-off for the best performing classifier(s).

Image Identifier To get a better qualitative insight on the performance of our classifiers and to see where the algorithm failed to rightly identify a masked face, we will add an image-identifier to our data dictionary.

Evaluation using the testset As with every ML project we will evaluate our findings using data the algorithm(s) have not seen before. As we refine our dataset we will accordingly have to refine our testset.

Stretch Goal: Multiclass classification Our initial dataset includes different types of wrongly worn masks. This allows us to train a classifier that can specify in what way a mask is wrongly worn. We consider an implementation of this feature a stretch goal as it hinges on our refined data also including these specific labels.

[4] We have some fitting datasets readily available.

References

[1] A. 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). *https://github.com/cabani/MaskedFace-Net*, last accessed: 2021-03-27.

[2] A. Cabani, K. Hammoudi, H. Benhabiles, and M. Melkemi. Maskedface-net – a dataset of correctly/incorrectly masked face images in the context of covid-19. *Smart Health*, 19:100144, 2021.

[3] M. Ejaz, M. Islam, M. Sifatullah, and A. Sarker. Implementation of principal component analysis on masked and non-masked face recognition. pages 1–5, 05 2019.

[4] M. S. Ejaz, M. R. Islam, M. Sifatullah, and A. Sarker. Implementation of principal component analysis on masked and non-masked face recognition. In *2019 1st international conference on advances in science, engineering and robotics technology (ICASERT)*, pages 1–5. IEEE, 2019.

[5] H. Li. Which machine learning algorithm should i use? `https://blogs.sas.com/content/subconsciousmusings/2020/12/09/machine-learning-algorithm-use/`. Accessed: 2021-03-28.

[6] M. Loey, G. Manogaran, M. H. N. Taha, and N. E. M. Khalifa. A hybrid deep transfer learning model with machine learning methods for face mask detection in the era of the covid-19 pandemic. *Measurement*, 167:108288, 2021.

[7] NVIDIA Corporation. NVlabs/ffhq-dataset: Flickr-Faces-HQ Dataset (FFHQ). *https://github.com/NVlabs/ffhq-dataset*, last accessed: 2021-03-27, 2019.

[8] scikit learn. Choosing the right estimator. `https://scikit-learn.org/stable/tutorial/machine_learning_map/`. Accessed: 2021-03-28.

[9] W. Vijitkunsawat and P. Chantngarm. Study of the performance of machine learning algorithms for face mask detection. In *2020 - 5th International Conference on Information Technology (InCIT)*, pages 39–43, 2020.

[10] Z. Wang, P. Wang, P. C. Louis, L. E. Wheless, and Y. Huo. Wearmask: Fast in-browser face mask detection with serverless edge computing for covid-19. *arXiv preprint arXiv:2101.00784*, 2021.

[11] Z. Wang, P. Wang, P. C. Louis, L. E. Wheless, and Y. Huo. Wearmask: Fast in-browser face mask detection with serverless edge computing for covid-19, 2021.

[12] World Health Organisation. Coronavirus disease (covid-19): Masks. `https://www.who.int/news-room/q-a-detail/coronavirus-disease-covid-19-masks`. Accessed: 2021-03-27.

A. Further Results

(a) Random Forest

(b) Stochastic Gradient Descent

(c) Decision Tree

(d) Linear Support Vector Classifier

Figure 4: Confusion Matrices