

# 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. A key motivator is the goal of better enforcement of mask mandates, many recent contributions on this topic focus on face-detection using deep learning approaches. We aim to contribute to the discourse by extending the scarce research on the performance of conventional Machine Learning algorithms. Therefore, we investigated the performance of different Machine Learning algorithms for image classification. We assess performance in terms of accuracy prediction and training speed. We achieved accuracy scores of well above 99% with a tuned RF classifier for portrait images. A relevant result in light of possible resource constraints is that an SGD classifier performs more stable when tuned for accuracy and speed at around 96%, at a fraction of the training time. The results indicate that neither classifiers usability is constrained by prediction time.*

*The GitHub-repository can be found in the footnote<sup>1</sup>.*

## 1. Introduction

The Covid-19 Pandemic shifted the focus of the scientific community towards new research questions that have not been focused on before. One of these newly arising questions concerns the issue of face-masks. Being recommended by the WHO as an effective measure against the spreading of Covid-19 [18], masks are an essential part of national pandemic response strategies all around the world [11]. Moreover, research has shown that face mask do significantly reduce the spread of Covid-19 [9]. Today, masks are mandated in grocery stores, at schools, office build-

ings, theaters and public places. These public places however lack an easy way of control and enforcement of mask mandates. Automated face-mask detection can play an important role here. 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 a visual feedback, whether one is wearing a mask correctly or even if one wears the right type of mask.

*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 control mask mandates, being a help to enforcement.

*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 public pressure to adhere to the rules.

These three use cases are certainly debatable - but the question whether and were to ethically use such technology is it out of scope of this paper. As we believe that there are ethical use cases in the realm of self- and group control, we do not see a problem of further developing such algorithms.

Current Methods of automated face mask detection are mostly deep learning approaches using convolutional neural networks. Regarding standard Machine Learning (ML) techniques, however, there is not much research to be found. In this paper, we are developing on the very scarce literature on the use of ML for automated face-mask detection. We analyze and compare the performance of different classifiers based on usability considerations. Our envisioned use case of an ML based classifier would be the monitoring of adherence to face-mask mandates at a public Hertie event. With this application in mind our analysis focuses on pre-

---

<sup>1</sup>The GitHub repository can be found at [https://github.com/the-palmo/hertie-ml-project\\_face-mask-detection](https://github.com/the-palmo/hertie-ml-project_face-mask-detection)

diction and training time, next to accuracy and is limited to portrait style pictures.

We started testing 6 different ML classifiers (Random Forest, Decision Trees, Stochastic Gradient Descent, Logistic Regression, K-Nearest Neighbours and Support Vector Machines) without hyperparameter tuning. We used the results to establish a baseline of the best performing ones and tune them on accuracy and prediction speed. As our project as well as real world applications are faced with resource limits in terms of computation time (see section 4), low training speed also led to dropping of classifiers.

Our results show that the Random Forest classifier performs best in terms of accuracy and precision, when run time is not a limitation. However, if used with a Principal Component Analysis (PCA) the accuracy drops significantly. The Stochastic Gradient Descent classifier is way more robust when tuned for speed. Our analysis found that while prediction speed can be boosted it is not an issue regarding usability for either classifier.

## 2. Related Work

**Face-Mask Detection with ML & DL** The task of face-mask detection is 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, 8, 17]. Contributions addressing the effectiveness of conventional machine learning classifiers remains scarce. Where conventional ML is addressed, the selection of methods appears arbitrary, the datasets that are used are small (apprx. 3200 images in [15]) and the code is not publicly available. In their recent study Vijitkunsawat and Chantngarm compared classification based on the CNN MobileNet 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%) [15].

**Performance analysis of ML classifiers** Our extension of the literature using conventional ML classifiers, analyses their performance with regard to training speed, accuracy and prediction speed. An approach that has been employed before for example in the selection of sparse triangular system solvers on GPUs [3]. Prediction speed especially has been analyzed on different occasions in ML-research, when usefulness or flow of the application hinges on timely execution. This has for example been the case for research in the context of algorithmic investing [16].

**Preprocessing of images** Other papers rely on object detection methods to identify multiple faces in pictures, and augment their data with flipped and de-noised images[4, 15]. In light of our limited resources we employed a different, pixel based approach to image classification and limit our scope to portrait style pictures. Building on the approach of Aurelien Géron we define a vector of features based on the pixel-values for every image [5]. Instead of grey-scale values we use RGB values. The resulting vectors serve as inputs for the machine learning algorithms.

## 3. Proposed Method

We employ the following research process:

- (1) Establishing the baseline with untuned ML algorithms
- (2) Tuning Phase I : Accuracy (by altering hyperparameters)
- (3) Tuning Phase II : Speed (by PCA)

### Establishing the baseline with untuned ML algorithms

In line with common research practice, we argue that it is worthwhile to test different algorithms to establish our baseline and find 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, their explainability<sup>2</sup>, and the prior use for face-mask detection in literature.

As our data is labeled, the chosen algorithms are all classifiers. In light of 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 to perform well on at least one of the criteria. This approach will allow us to comprehensively search for the best performing candidate for further tuning.

Support Vector Machine and K-Nearest-Neighbours classifiers are included in our initial set as they were used in the scarce existing publications on ML-classification performance regarding face-mask detection. The other classifiers are chosen due to the following reasons: For datasets larger than 100,000 observations using the stochastic gradient descent is advised by sci-kit learn [12]. 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 regarding speed [7]. We include these different classifiers initially to evaluate whether speed or accuracy is the larger problem in our project.

Based on fitting each of our criteria with at least one classifier, our initial set of conventional ML classifiers are

---

<sup>2</sup>As our research is not theory generating but application based, we value performance over explainability.

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 4. After this first evaluation the suitability of the algorithms and the data is analyzed and possible steps for refinements are derived.

After running the six different classifiers without hyperparameter tuning, we select our baseline as follows. We evaluate the results for accuracy and training time. We are dropping classifiers that are rather low in accuracy compared to the other classifiers or have an exceptionally long training time. As after this process only two classifiers were used for the next phase, we decided not to combine these two to one baseline but rather using both untuned classifiers as the baseline for the corresponding tuned classifier.

**Tuning Phase I: Accuracy** In the first phase of our tuning, we are optimizing the hyperparameters for our most promising classifiers to boost their accuracy. This is done via `Scikit-Learn`'s `GridSearchCV` method and accuracy affecting hyperparameters are tuned in the grid search.

**Tuning Phase II: Speed** In the second tuning phase, we are tuning our algorithm on processing speed by reducing the dimensionality of the data. Tuning on speed will negatively affect accuracy as typically more features demand computation time but offer accuracy. In our project, the decision for the best trade-off will be based on our assumed use case - monitoring mask mandate compliance at a Hertie event. In a first step we check who many dimensions are explaining most of the variance in our data and how limiting the analysis to these impacts both accuracy and speed. For this purpose a PCA is performed.

## 4. Experiments

**Data** In order to conduct the project described above, we rely on labelled datasets that include images of people wearing masks correctly, incorrectly or not at all. Luckily, there were suitable datasets available.

Our *initial dataset* was provided by Cabani et al. published in March 2021 in the *Journal Smart Health* [2] and made available via `GitHub` [1]. The authors rely on the dataset of face images `Flickr-Faces-HQ3 (FFHQ)`, publicly made available online by `NVIDIA Corporation` [10]. 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 look into the pictures seem to val-

idate 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 resulting dataset contains 66,948 pictures with correctly worn artificial face-masks (example: figure 1a) and 66,634 pictures with incorrectly worn artificial face-masks (example:figure 1b).<sup>3</sup> Pictures from the same persons are used once for a correct worn image and once for an incorrect worn image.

In the course of the project we decided to supplement the initial dataset with real-world data, to address any potential issues arising from the artificially masked faces and the high homogeneity of the data including only blue op-type masks. Our *supplementary dataset* is available through `kaggle` and was published in the context of a data science challenge [6]. It is intended for educational purposes only. It contains different portrait style images of real-life masked and unmasked faces with varying pixel resolutions. The images were scraped from multiple sources e.g. Google images, `pexels.com`, etc. The result is a dataset containing 5,632 pictures with correctly worn face-masks and 5,632 pictures of people without a face mask.

**Software** We implemented the whole project with `Python` using `Jupyter Notebook` as an development environment. Furthermore, we used `GitHub` and `GitHubDesktop` for version control and file sharing of smaller files. Lastly, we relied on `Dropbox` to share larger files like the image data folders and the numerical input files resulting from the data preprocessing.

<sup>3</sup>In the abstract of the published paper the authors speak of roughly 3,350 pictures more, but the dataset on `GitHub` is constantly cleaned where the face detection algorithm produced wrong results.

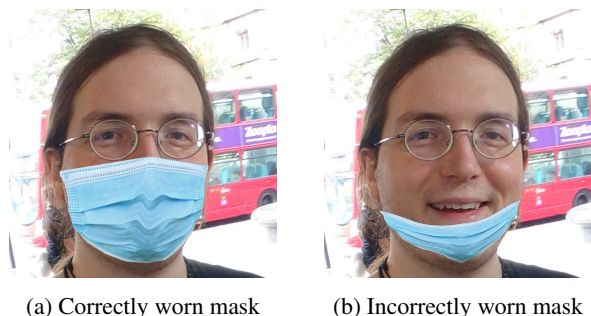


Figure 1: Example pictures of initial dataset

**Hardware** The code development took place on normal laptops of the team members, relying on the described toy dataset for making sure that the code runs properly. The preprocessing as well as the actual modelling with the full dataset was performed on one of the team members laptops. This laptop is a Lenovo ThinkPad X1 Carbon with an Intel i7 processor and 16GB RAM in 8 CPU cores. Where possible, Scikit-Learn's build-in feature to perform parallel runs was leveraged on 4 CPU cores.

**Data wrangling** In order to get machine interpretable data the images of the dataset have to be preprocessed. The raw data is organized in separate folders. For the initial dataset the data is organized in one folder 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 pictures of the second dataset (also .jpg-files) are not separated, but a .csv-file including the filenames and the corresponding labels is provided with it. These portrait pictures have different pixel resolutions.

The data wrangling is performed in the first Jupyter Notebook<sup>4</sup>, as we wanted to separate this one time operation from the process of model tuning that occurs later in the project process. As the folder structures for the two datasets are slightly different the first steps to load the data are also slightly different. In the following, the most important steps taken during the data wrangling are described: 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 approx. 41 GB of data) and get to a reasonable speed of processing, a pixel reduction is necessary. We decided to reshape all pictures to a starting resolution of 24x24 pixels. The last and final point is how to save the transformed dataset so that another Jupyter Notebook can access the numerical representation of the pictures without having to run through the data wrangling process every time. We decided to organize the numerical data of the pictures together with the corresponding labels and picture IDs 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 three 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 (144846, 1728). 144846 is the number of observa-

tions (i.e. correct plus incorrect pictures) and 1728 is the number of features included (i.e. 24\*24 pixels times 3 for the RGB representation). Additionally, the dictionary contains another array with the corresponding labels and additionally one including the picture IDs.

**Evaluation method** For the evaluation of our classifiers, we use the following metrics: *the cross validation score, the confusion matrix, the accuracy, 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. For the final evaluation on the test set, we however use the accuracy score.

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 corrected 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.





		classification as	
		no mask	mask
reality	no mask	TN  ▲ No mask detected ▲	FP  mask detected
	mask	FN  ▲ No mask detected ▲	TP  mask detected

Figure 2: Graphical representation of the confusion matrix

<sup>4</sup>hertie-ml-project.face-mask-detection/01\_Data-Collection-Processing.ipynb

The **precision & recall scores** are the scores which determine the share of true positives in all instances classified as positive (precision) and in all positive instances in reality (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.

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

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 or every hour. 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 others to test and further develop our solution.

A slow **prediction speed** is even more problematic as it would directly inhibit real-time detection of mask mandate compliance, that needs predictions within about less than a second for usability in our assumed use case.

**Experimental details** For establishing our baseline code, we used a toy dataset with 202 pictures, that we extracted from the two datasets at the beginning. We did this to check and develop our code in shorter time periods, as working with the whole dataset takes a lot of run time. We provide this toy dataset so that our code can be run and checked without the need of downloading excessive amounts of data.

*Preparation:* 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 approx. 14,500 instances as reasonable for testing the performance.

*Establishing the baseline with untuned ML algorithms:* After splitting, we trained and evaluated our six untuned classifiers to establish a baseline and evaluate the initial performance of each classifier. The only tuning made are the setting consistent random states and the setting of higher maximum iterations for LinearSVC, as convergence could not be reached with the default value of 1000. Unfortunately, the K-Nearest-Neighbor and Logistic Regression al-

gorithms did not finished as the kernel died several times before finishing. Thus, we only report the results of the other four algorithms, see figure 4 in appendix A.

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 the overall run time. Based on the results we only continued to investigate the Random Forest Classifier and the SGD Classifier as the former is the most precise and the latter the fastest classifier. We did not further investigate the Decision Tree Classifier, because in terms of accuracy it will only become a version of the Random Forest Classifier and presumably will not perform as good. We dropped the Linear SVC Classifier due to its extraordinary long run-time of evaluation as we were limited in time resources.

*Tuning Phase I - Accuracy:* As the results of the baseline for Random Forest and SGD were already very good, tuning the hyperparameter will only result in a small increase in accuracy. The hyperparameter tuning was performed via Scikit-Learn's GridSearchCV method. The grid search was performed with 3 folds of the training set. The scoring was set to a list of the order precision, recall, accuracy. Thus the grid search tuning evaluated the folds first and foremost on precision. This is in line with our assumed use case, as false positive classifications of people who are not wearing masks correctly is worse than the opposite.

For each classifier a hyperparameter grid was defined that resulted in 18 different combinations each. Only hyperparameters that potentially affect accuracy are taken into account. For the SGD Classifier the parameters `penalty`, `alpha` and `max_iter` were varied. For the Random Forest Classifier the parameters `n_estimators`, `max_features` and `bootstrap` were used. Due to the limited space in this report, we refer to the Scikit-Learn documentation for an explanation of these hyperparameters [14, 13]. The precise hyperparameter grids that we used are shown in appendix B. As the grid search tests 18 different hyperparameter configurations on 3 folds of the training set this step is very resource intensive. Thus, we decided not to run several runs of hyperparameter tuning but only one, as our overall precision and accuracy levels are already high. <sup>5</sup>

<sup>5</sup>We acknowledge that there might be still room for improvement when further grid search runs with more refined grids are done. Furthermore, we did not consider the effect of other pixel resolutions than our starting configuration of 24x24 on the performance in terms of precision and accuracy, as this process would involve even more run-time. Additionally, we considered and implemented a Bagging Classifier of SGD Classifiers and Voting Classifier of the SGD and the Random Forest Classifier. However,

*Tuning Phase II - Speed:* After optimizing the algorithms in terms of accuracy we included a second step in order to improve these tuned algorithms with respect to processing speed. As stated in the introduction short training and prediction times are very important for a reasonable application. Images are prone to having a lot of features. As every pixel in our case is represented by 3 RGB values we have in the baseline configuration  $24 * 24 * 3 = 1782$  features. Presumably, a lot of these features - especially at the edges - will not play a major role in detecting a mask. A plot of the feature importance of the Random Forest Classifiers confirms the fact that only a few features play an important role. Figure 6 in appendix C shows impressively that only the pixels in the lower middle area of the pictures, where the mask in portrait areas most often is located, play a role.

Based on this fact a PCA was performed in order to reduce the dimensionality of the dataset without losing too much variance. In figure 7 in appendix C the cumulative variance of the dataset is plotted over the number of features. This analysis reveals that only 184 dimensions are necessary to explain 95% of the dataset variance (shown as a dotted line in the figure).

The classifiers were run on this reduced dataset with the same hyperparameters that we used for the evaluation on the full dataset without dimensionality reduction. Thus, there might be further room for improvement when performing a second round of accuracy hyperparameter tuning after the Pappx. A slightly different configuration of hyperparameters might improve the performance in terms of accuracy. However, as the scope of this project is limited we did not investigate this further and stuck to the same classifier configuration applied to different datasets. Another way of improving the speed is to change the pixel configuration of the input. Given the limit scope of this project and its similarity to PCA discussed above, we did not further investigate this path - however the code offers the way to investigate this path in future work.

**Results** The final step of the project is to evaluate the different classifiers on the test set. For a comparison the two different algorithms are each trained on the training set with three different configuration: The baseline configuration with default hyperparameters, an accuracy optimized configuration and a configuration where the accuracy optimized configuration was applied to dimensionality reduced datasets. The evaluation results are listed in figure 3.

*Discussion of results for the baseline:* The untuned baseline classifiers both performed very well in terms of precision on the test set with both performing well above 99%.

---

as the two classifiers individually perform already very good the Bagging and Voting Classifiers basically had no improvements to offer while causing a lot of run-time.

However, the SGD baseline classifier seems to be trained extremely biased towards precision performance. While it is very important to have very few false positive classifications in our use case, this comes with a price of very poor performance in terms of recall. The baseline SGD classifier seems to classify a mask as correctly worn only when it is absolutely sure. Due to the poor recall performance, the overall accuracy is also rather bad. The baseline Random Forest (RF) classifier is much more balanced and performs very good on both metrics.

In terms of run-time the SGD classifier clearly outperforms the RF classifier - in this baseline configuration as well as for all others. Considering our assumed case, it is important to note that the prediction speed virtually plays no role: The prediction time in the table gives the amount of time necessary to predict the class of all test observations (approx. 14,500 in our case). The training time on the test set is considerably longer, but both finish training in a couple of minutes. Here again the baseline SGD classifier is about 5 times faster than the baseline RF classifier.

*Discussion of results for tuning phase I - Accuracy:* The results for the accuracy tuned classifiers were obtained with the hyperparameter configuration, specified in appendix B. These were the output of the grid search. For the RF classifier only the recall score improved slightly. As all scores for the baseline configuration were already very high (above 99%) this is no surprise, especially as the grid search basically confirmed that the default values were nearly optimal. However, the increase in estimators also resulted in a considerable longer run-time for both training and prediction time without improving the score significantly.

For the SGD Classifier the results are different. First of all the grid search was performed with respect to precision, recall, accuracy - in this order. Obviously, the optimization of the hyperparameters resulted in trained classifier that generalizes much better on new data than the baseline classifier. Although, there is a small decline in the precision score the overall accuracy score improves extraordinarily due to the much better performance on recalls (Baseline: 64.0% - Accuracy Tuned: 98.7%). Interestingly, the accuracy improvements does not come with the price of higher run-time. The training time is even cut by half with the new hyperparameter configuration. This is likely the case because the grid search for most accurate hyperparameters recommended 100 instead of the default value of 1000 maximum iterations.

*Discussion of results for tuning phase 2 - Speed:* The results for third set of classifiers are obtain by using the same hyperparameter configuration as described in the previous section, but on a feature reduced dataset obtain by a Pappx. As expected the overall accuracy of both classifiers decreases,



	Accuracy Score	Precision Score	Recall Score	Prediction time in seconds	Training time in seconds
<b>RF: Baseline</b>	0.993	0.994	0.991	0.478	341.599
<b>RF: Tuned Accuracy</b>	0.993	0.994	0.993	1.164	1360.095
<b>RF: Tuned Speed</b>	0.929	0.917	0.942	1.045	698.753
<b>SGD: Baseline</b>	0.821	0.999	0.640	0.019	69.213
<b>SGD: Tuned Accuracy</b>	0.976	0.966	0.987	0.019	32.936
<b>SGD: Tuned Speed</b>	0.965	0.968	0.963	0.003	5.135

Figure 3: Overview Evaluation Results

as the reduced dataset contains only those 184 dimensions that account for 95% of the variance. However, the two classifiers behave differently. The RF classifier loses about 6.5%-points in terms of accuracy and even 7.7%-points in terms of precision. The training time is roughly cut in half. In terms of training and prediction speed the baseline configuration significantly outperforms both tuned models.

In contrast, the SGD classifier’s accuracy only decreases about 1.0%-points and the precision even slightly increases. The classifier configuration shifts to more ”no mask” classifications and thus producing less false positives. This might be due to different dimensionality as a PCA introduces a perfectly orthogonal n-dimensional space, whereas before not all 1782 dimensions were orthogonal. The run-time effect is considerable. Both run-times are cut by a factor of approx. 6 and the training time now only takes about 5 seconds for the SGD classifier. Thus the SGD classifiers does not suffer much from the dimensionality reduction in terms of accuracy performance, but benefits extraordinarily in terms of speed.

Potentially, further improvements in terms of accuracy, precision and recall can be achieved by optimizing the hyperparameters on the new reduced dataset (see section 4). However, these will most likely come with run-time costs. For an overview of the amount of false positives and false negatives, we added the confusion matrices in the Appendix D.

*Overall comment on quantitative results:* Overall, our quantitative results in terms of accuracy, precision, and recall are way better than we initially expected. This might still be due to the fact that the dataset is dominated by pictures of the same artificially inserted blue medical mask in different pictures, as seen in figure 1. Thus, it is necessary to have a qualitative look at the errors and whether the algorithms perform better on one part of the data than on another. This qualitative check is discussed in the next section.

## 5. Qualitative Analysis

Next to our quantitative assessment we looked more deeply at errors and compared the type of errors, their origin in our datasets, overlap between the algorithms and possible reasons for the errors. This analysis is for sake of clarity limited to the accuracy tuned classifiers.

*Data origins of errors and overlap:* Considering the mismatch in size of our datasets, there is a large amount of errors stemming from our supplementary dataset. While it is only make up less than a 10th of our data it accounts for 69% of the errors of the RF classifier and for 47% for the SGD classifier. This might be an indication of issues stemming from low data diversity and a need for improving the variety of the data further. There is considerable overlap regarding the errors as 52 of the 95 total errors of the tuned RF classifier are shared with the SGD classifier.

*Reasons for errors:* Considering the common errors between the classifiers there are some patterns recognizable. First, especially in the supplementary dataset there are pictures at ”off-angles” that do not qualify as portrait style which can be considered outliers. Secondly, the algorithms seem to have problems, if the contrast between the mask and the surrounding image is low. This was the case for pictures with a very low initial resolution, colorful artificial lighting, masks that have prints on them or match the skin colour. The latter effect was more often visible concerning people of color. Especially the last point should be addressed through a refinement of the data foundation and iteratively tested, to prevent introducing racial-bias in the algorithm.

## 6. Conclusions

This project successfully implemented ML algorithms for face mask detection. It showed that ML algorithms based on image classification of pixel representations can indeed solve the problem at hand in a very reliable and fast manner. We achieved accuracy scores of well above 99% with

the tuned RF classifier trained on all features. If resource constraints are not much of an issue in the application, e.g. when the classifier is only trained once per day, this classifier is likely the best pick. However, when time and resource constraints are a significant limitation the SGD classifier might be worth considering. It has a much better run-time and can be trained very fast and still has an overall accuracy of well above 96%. Especially in our assumed use case of large gatherings this is beneficial, as the algorithm can be constantly updated and trained even several times per hour. This conclusion is of course limited by some factors described below.

Besides the main findings we want to note that we ourselves learned a lot in this project. Next to improving our coding skill set significantly, we also learned how the different algorithms perform and what their limitations are. On top of that we learnt how to answer policy and society related questions with ML techniques.

### **Limitations / Scope for Improvement**

Our research was bound by resource constraints, in terms of time, computational power and available data. Against this backdrop there is room for improvement.

*Data limitations:* As discussed above high quality data is pivotal for real-world application. We limited our approach to portrait style pictures consisting of a large set of images artificially augmented with masks and a smaller real-world set. Thus our classifiers are only performing on portrait style pictures and are not able to identify masks in groups of people or if the input is not a portrait.

*No use of ensemble methods:* When analyzing and tuning multiple classifiers implementing an ensemble approach can be beneficial. However, we decided against this for two reasons. Firstly, our results were already very accurate and there was considerable overlap in the errors, only offering limited scope for improvement. Secondly, by design ensemble classifiers would yield longer prediction and training times.

*Limited hyperparameter tuning:* The project could be improved with additional runs of hyperparameter tuning and by including further hyperparameters.

*No comparison with deep learning performance* As discussed above deep learning approaches dominate the literature. Thus it would have been interesting to check the performance of a CNN classifier to benchmark our solutions.

## **7. Acknowledgements**

As explained above any ML project hinges on the availability of high quality data. We are grateful to Cabani et. al, the NVIDIA Corporation and the kaggle user Manish KC for the provision of the datasets. We emphasize that the data is only used for educational purposes and any derivative work must be published under Creative Commons BY-NC-SA 4.0.

## **8. Contributions**

In general, we were very happy concerning collaboration, division of tasks and the learning aspect for all of us. Very often we discussed our approach and code together before someone implemented and pushed it to Github. The major tasks, as conceptualisation of methodology and project planning have been tackled as a team. Nonetheless, we set different foci based on our skillsets and learning interests.

Thilo is the most adept coder of us, therefore he took over major parts of the data wrangling and advised the others with regard to coding practices and issues. Moreover, he was responsible for debugging, simplifying and running the code on his machine.

Fabian and Tobias, as they are less experienced in coding, did a lot of co-coding to better deep dive in the matter together. They focused on the second Jupyter Notebook with respect to training and evaluating the different classifiers. They co-coded the evaluation outputs and graphics you can see in the appendix. Later, Fabian took charge for the development of the hyperparameter tuning with respect to accuracy and Tobias was responsible for developing the code for optimizing run-time and implementing the PCA and subsequent analysis.

Moreover, Tobias and Fabian had the lead in writing the reports conducting the literature research and review and creating the supporting graphics. Thilo supported them on this task by writing major parts of the data and experimental sections.



## 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] E. Dufrechou, P. Ezzatti, and E. S. Quintana-Orti. Automatic selection of sparse triangular linear system solvers on gpus through machine learning techniques. In *2019 31st International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*, pages 41–47, 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. pages 1–5. IEEE, 2019.
- [5] A. Geron. *Hands-on machine learning with Scikit-Learn and TensorFlow : concepts, tools, and techniques to build intelligent systems*. O’Reilly Media, Sebastopol, CA, 2017.
- [6] M. KC. Face mask dataset. [https://www.kaggle.com/manishkc06/face-mask-dataset?select=train\\_labels.csv](https://www.kaggle.com/manishkc06/face-mask-dataset?select=train_labels.csv). Accessed: 2021-04-28.
- [7] 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.
- [8] 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.
- [9] T. Mitze, R. Kosfeld, J. Rode, and K. Wälde. Face masks considerably reduce covid-19 cases in germany. *Proceedings of the National Academy of Sciences*, 117(51):32293–32301, 2020.
- [10] NVIDIA Corporation. NVlabs/ffhq-dataset: Flickr-Faces-HQ Dataset (FFHQ). <https://github.com/NVlabs/ffhq-dataset>, last accessed: 2021-03-27, 2019.
- [11] H. Ritchie, E. Ortiz-Ospina, D. Beltekian, E. Mathieu, J. Hasell, B. Macdonald, C. Giattino, C. Appel, and M. Roser. Covid-19: Face coverings. <https://ourworldindata.org/covid-face-coverings>. Accessed: 2021-04-20.
- [12] scikit learn. Choosing the right estimator. [https://scikit-learn.org/stable/tutorial/machine\\_learning\\_map/](https://scikit-learn.org/stable/tutorial/machine_learning_map/). Accessed: 2021-03-28.
- [13] scikit learn. Ensemble methods. [://scikit-learn.org/stable/modules/ensemble.html](https://scikit-learn.org/stable/modules/ensemble.html).
- [14] scikit learn. Stochastic gradient descent. <https://scikit-learn.org/stable/modules/sgd.html>. Accessed: 2021-04-15.
- [15] 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.
- [16] F. Wang, Y. Zhang, Q. Rao, K. Li, and H. Zhang. Exploring mutual information-based sentimental analysis with kernel-based extreme learning machine for stock prediction. *soft computing*, 21(12):3193–3205, 2017.
- [17] 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.
- [18] 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. Appendix: Baseline results on train set

	LinearSVC	RandomForestClassifier	SGDClassifier	DecisionTreeClassifier
precision score	0.992	0.994	0.977	0.98
recall score	0.937	0.989	0.984	0.979
cross validation scores	[0.971, 0.966, 0.957]	[0.992, 0.991, 0.992]	[0.981, 0.979, 0.98]	[0.98, 0.979, 0.979]
cross validation mean	0.965	0.992	0.98	0.979
cross validation std	0.00579	0.00047	0.00082	0.00047
eval run time in seconds	6807.78	1355.63	375.12	1015.83

Figure 4: Baseline results

## B. Appendix: Hyperparameter Grids and used configurations

	Parameter	Values
random forest	n_estimators	[ 10 , 100 , <b>250</b> ]
	max_features	[ <b>"auto"</b> , "log2" , None ]
	bootstrap	[ True , <b>False</b> ]
SGD	penalty	[ "l1" , "l2" ]
	alpha	[ 1e-4 , <b>1e-2</b> , 1 ]
	max_iter	[ <b>100</b> , 1000 , 10000 ]

Figure 5: Hyperparameter grid for grid search. **Bold:** output of Grid Search

### C. Appendix: Feature Importance and Principal Component Analysis

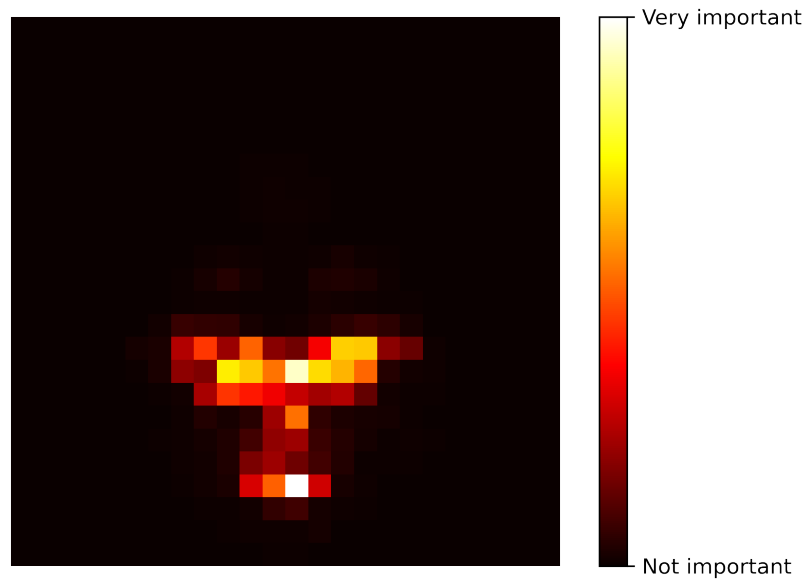


Figure 6: Feature importance of Random Forest Classifier

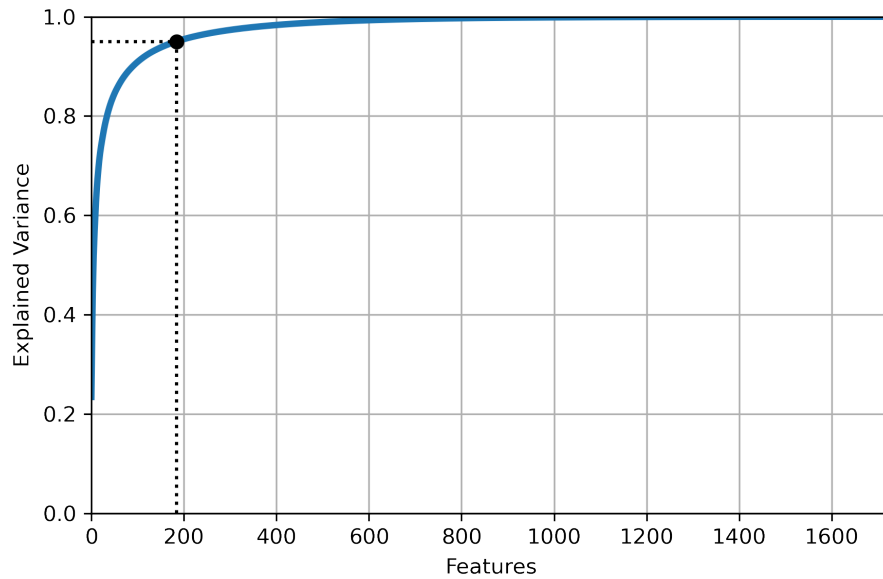
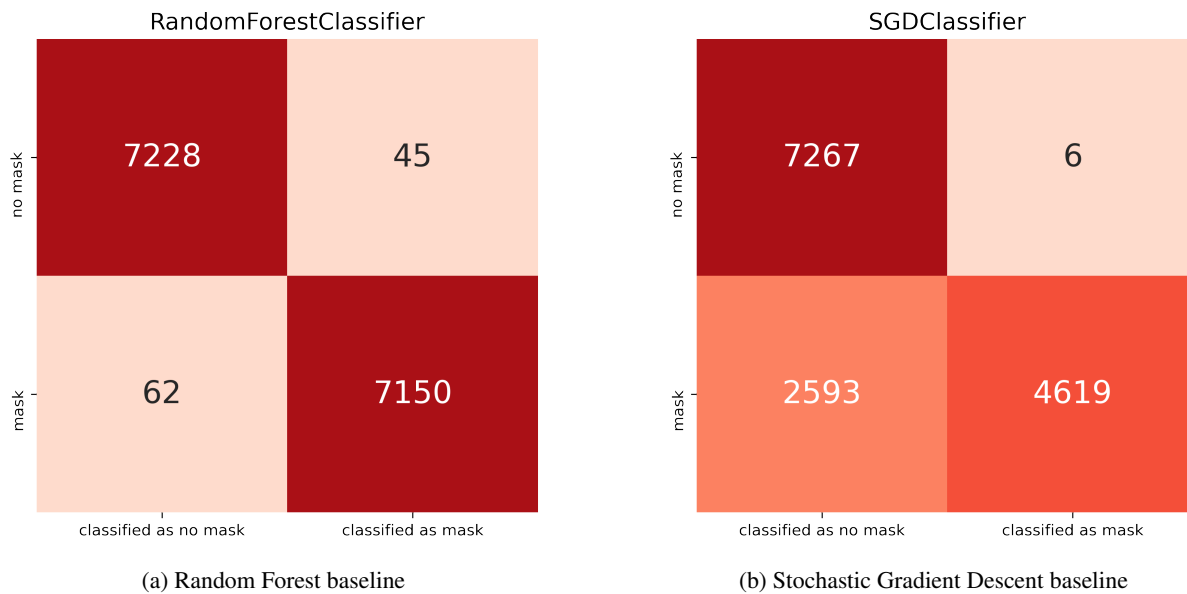


Figure 7: PCA - Explained variance

## D. Appendix: Confusion Matrices

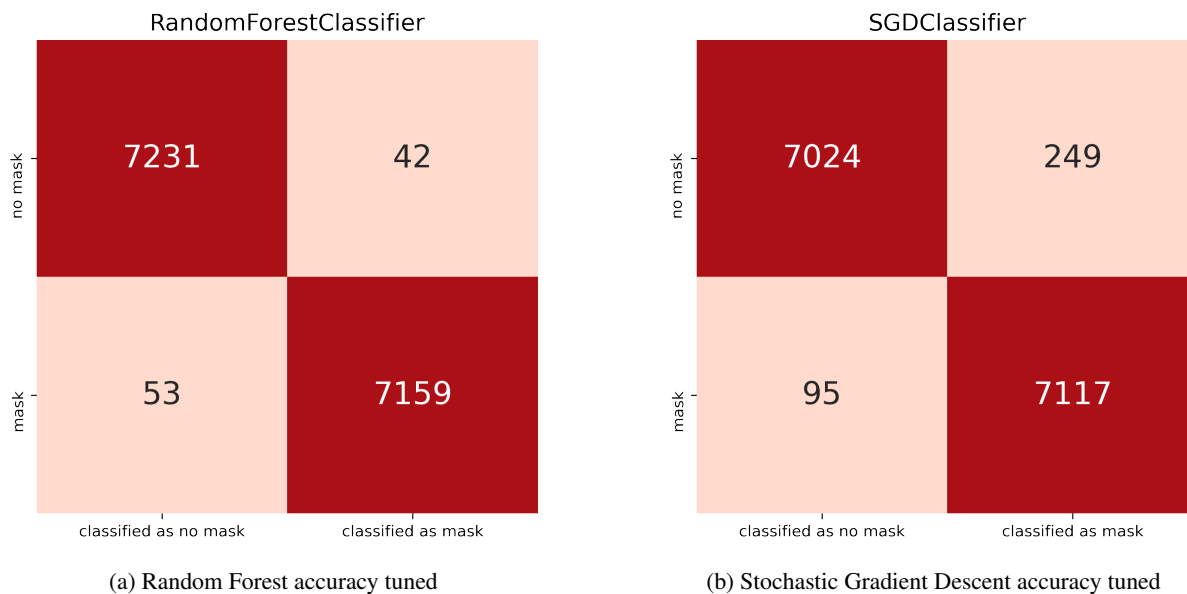
### D.1. Baseline

Figure 8: Baseline classifiers - 24x24 Pixels - Dataset: Unreduced - Confusion Matrices



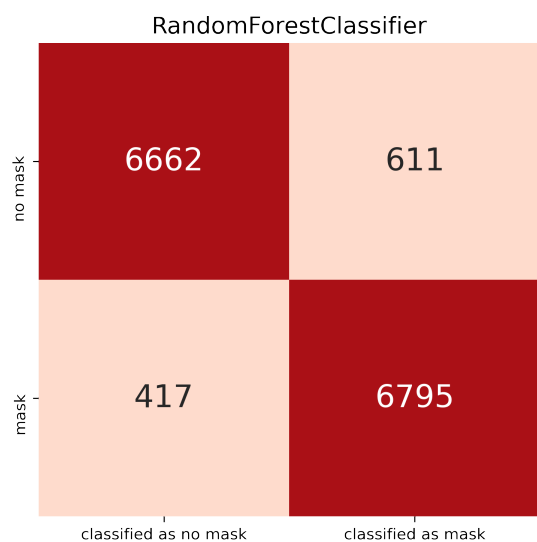
### D.2. Accuracy Tuned

Figure 9: Accuracy tuned classifiers - 24x24 Pixels - Dataset: Unreduced - Confusion Matrices

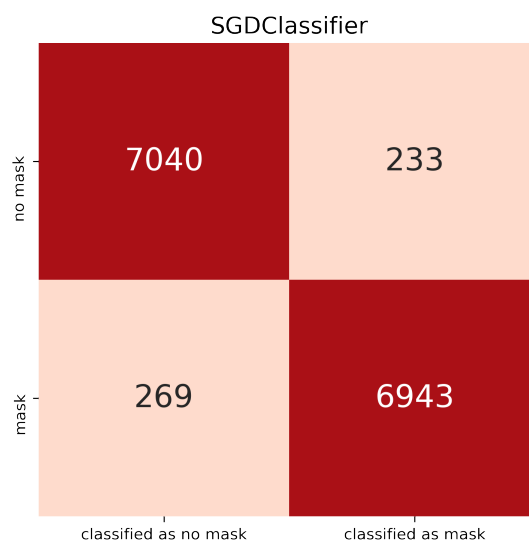


### D.3. Speed Tuned

Figure 10: Speed tuned classifiers - 24x24 Pixels - Dataset: Reduced by PCA - Confusion Matrices



(a) Random Forest speed tuned



(b) Stochastic Gradient Descent speed tuned