# Deep Learning Facemask Detection With CNNs

**Ennio Strohauer**
ennio.strohauer@hpi.uni-potsdam.de

**Samuel Kunst**
samuel.kunst@hpi.uni-potsdam.de

## Abstract

This report presents our deep-learning project focused on classifying individuals wearing face masks using a Convolutional Neural Network (CNN) architecture. We address key challenges in dataset collection, preprocessing, and model optimization. Our CNN model achieves a commendable 94% accuracy in correctly identifying mask usage across diverse scenarios. This work underscores the significance of deep learning in public health and offers a promising tool for real-world mask detection, while acknowledging potential areas for future enhancement.

## 1 Introduction

The COVID-19 pandemic presented a global health and safety crisis of unprecedented scale. As part of preventative actions, one of the most commonly employed countermeasures taken by many institutions was the mandated wearing of face masks. Especially in critical infrastructure, for instance hospitals or high congestion points such as airports, the wearing of face masks played an important factor in slowing the virus' rate of spread. However, enforcing such requirements is often easier said than done, especially on larger scales.

Machine learning models are arguably an excellent solution candidate for this problem domain. Given properly diverse and well-constructed datasets, classification models are able to accurately distinguish many classes to an amply sufficient degree, and do so with both greater efficiency and at a greater scale when compared to alternative methods.

That being said, classification tasks are no new idea in the field of digital health, and in fact are already often employed in related processes such as medical imaging, for instance in the detection of malignant cancer cells. As such, this report aims to document our process of building upon existing state-of-the-art solutions in order to construct a classifier which is able to accurately detect an face-mask-wearing individual in a variety of contexts, for instance by mask type/color, angle, or ethnicity. We also discuss some of the challenges and considerations which were taken during this process and outline the various processing steps and technical measures which we undertook during the training of these models. Finally, we discuss our results and go into the limitations of our findings, suggesting possible further extensions to the presented work.

## 2 Related Work

Face mask detection, particularly post-global pandemic, has become a domain of keen interest. As public safety emerged as a paramount concern, there was a surge in automated systems to ensure proper face mask usage. We examine notable contributions in this domain:

Vinh et al. delve into traditional image processing techniques like Haar cascades. Their work, conducted in 2020, suggests that while these techniques have shown proficiency in controlled setups, they exhibit limitations when presented with varied lighting, diverse mask types, and multiple facial poses (Vinh et al. 2020).

Deep Learning Approaches have, by and large, been the focal point for image-based classifications inclusive of face mask detection. Among these, the use of Convolutional Neural Networks (CNNs)

has been most prevalent. Mandal et al., in their 2021 research, give weight to the argument by using *ResNet* architectures, fine-tuned specifically for this task. Their findings report exceptional outcomes, thereby placing CNNs at the forefront of mask detection techniques (Mandal et al. 2021).

Talahua et al. in 2021 explored the confluence of face mask detection and facial recognition. The challenge that emerges, as underscored by their research, is the ability of the system to recognize individuals, especially when a considerable part of their face remains concealed behind a mask (Talahua et al. 2021).

Drawing from these foundational studies, our endeavor aims to engineer an adept solution tailored to our problem domain.

## 3 Dataset

In the course of this research, we utilized the Real-World Masked Face Dataset (RMFD) developed by GitHub contributor `X-zhangyang`. This dataset was facilitated by the *National Multimedia Software Engineering Technology Research Center* at Wuhan University. Given the context of our research, it's noteworthy to mention that the RMFD, with its expansive collection of 92,651 images, is considerably extensive. The curation of this dataset was a direct response to the worldwide adoption of mask-wearing, instigated by the COVID-19 pandemic. As articulated in the documentation accompanying the RMFD, its overarching aim is to "provide data resources to aid in addressing analogous public safety occurrences in subsequent times" [1]. With regard to facial orientations and diversity in mask types, the RMFD offers a commendable range. Nonetheless, during our engagement with this dataset, we encountered certain challenges, which will be elaborated upon.

### 3.1 Data Processing

The RMFD consists of masked 90,468 and unmasked 2,183 colored RGB images. The contents of the dataset are largely unstructured, both in image dimensions as well as file structure. This made initial evaluation difficult and prompted considerable preprocessing before training.

Our preprocessing steps involved utilizing PyTorch's `ImageTransform` compositions to apply a series of transformations in order to standardize the dataset to have both consistent characteristics in addition to having compatible dimensions with our model architectures. In particular, the processing pipeline we opted for consisted of the following sequence of transformations:

1. Convert to grayscale
2. Center crop to $1/1$ aspect ratio
3. Resize to $(256, 256)$
4. Normalize pixel values
5. Convert to PyTorch `Tensor` object

We opted for conversion to grayscale with the goal of promoting structural feature detection, such as edges. Combined with normalization, we hoped that this choice would lead improved generalization performance. With this being said, the use of `ImageTransforms` offers the distinct benefit of just-in-time (i.e on-the-fly) processing during training. This makes it very easy to adapt the transformations used from one training run to the next, without having to manually re-process the entire original dataset. This flexibility proved extremely valuable, particularly during iteration on model parameters.

There are a number of assumptions and considerations which arise out of our preprocessing, most notably that the subject was placed about the center of the image. Furthermore, we assume that images were taken in comparable environments in terms of lighting, distance to subject, and/or rotation. We aimed to mitigate any outliers via pixel normalization, although this would only aid in standardizing lighting conditions rather than the other previously mentioned discrepancies.

[1] This description was translated to English from Chinese and may not be accurate word-for-word

### 3.2 Dataset Structure

Another aspect of our data-processing is the lack of distinct testing and evaluation (or holdback) sets. To address this issue, we opted for manually picking a one-time randomly chosen seed value which we utilized to remove 100 images from each class in an unbiased manner. These images were manually removed from the original dataset and were exclusively used during evaluation in order to reduce the likelihood of memorization-bias during performance evaluation.

### 3.3 Class and Demographic Imbalances

Additionally, the substantial class imbalance posed a potential issue which required addressing. We opted to create a custom sampler which favored the underrepresented class more heavily during training. Using this sampler would ensure an even split of masked to unmasked samples during training runs. Despite this, there are a number of implications which arise out of this methodology, which we discuss in further in the discussion and results section of this report. Another option we considered was to augment our dataset, however we ended up not incorporating this technique. Given the substantial volume of our dataset, our primary objective was to first and foremost develop a functional model. Introducing augmented images would have escalated both the storage requirements and the computational time during training. However, it's worth noting that our file architecture is designed with flexibility in mind. This means that integrating data augmentation in the future can be seamlessly achieved by modifying our data-loader system.

The final — and potentially most significant — consideration with regards to the RMFD is a lack of ethnic diversity. The dataset consisted of entirely East Asian subjects, which we initially thought may pose an issue in terms of model performance when applying it to other demographics. Again, we considered data augmentation with computer-generated samples in order to achieve a more representative dataset, however opted to not due to the same reasons described above. We discuss the implications of this decision in more detail in the discussion section of this report.

## 4 Methodology

The following section goes into detail regarding our choices in relation to model architectures and our training/evaluation processes.

### 4.1 Model Architectures

For our model architectures, a convolution-based approach was chosen and subsequently developed. CNNs are the de-facto industry standard when it comes to problem domains of this type, offering both excellent levels of performance in addition to parallelization across model and data dimensions alike. This allows for efficient scaling in larger datasets and/or application contexts, a feature which is particularly desirable in the context of our problem.

The primary model architecture used in this project is a modified version of the *LeNet-5* model (LeCun et al. 1998). The original *LeNet-5* model was designed by Yann LeCun and is one of the pioneering convolutional neural networks (CNNs) that significantly impacted the field of deep learning at its time, especially in image classification tasks. The modified architecture, from heron out referred to as *Model1*[2], is constructed as follows:

1. Convolutional Layer 1: This layer has 6 filters with a kernel size of $5 \times 5$. It is followed by batch normalization, a *ReLU* activation function, and a max-pooling layer with a kernel size of $2 \times 2$.
2. Convolutional Layer 2: This layer is identical to the first convolutional layer, except that it consists of 16 $5$ filter kernels rather than 6.
3. Fully Connected Layer 1: This FC layer has 512 neurons and is followed by a *ReLU* activation function.
4. Fully Connected Layer 2: This FC layer consists of 256 neurons and is followed by a *ReLU* activation function.

[2]We are not graded on creativity (at least we hope so...)

5. Output Layer: The final layer is a fully connected layer with a single neuron, which outputs the probability of a person wearing a mask.

In terms of depth and layer buildup, *Model1* and *LeNet-5* are comparable. The primary differences lie in *Model1*'s increased kernel and FC-layer dimensions, as well as the use of *ReLU* rather than $tanh$ as an activation function. *LeNet* also makes use of average-pooling rather than max-pooling.

For comparison purposes, the original *LeNet-5* architecture was also implemented and trained, as well as a modified version of the state-of-the-art *ResNet-18* model, where the final fully connected layer was modified to accommodate our binary classification problem. Comparisons between these three architectures are discussed in further depth in the discussion section.

### 4.2 Training

Each model was trained using the same base training loop (`train_model.py`). The training loop was responsible for creating and initializing data-loaders for both training and validation. We opted for a validation split of $0.2$, or $20\%$ of all images, to be randomly selected prior to each training loop. This practice is rooted in the need to prevent over-fitting in addition to ensuring that the model is routinely evaluated on data it has not seen during training. We found $0.2$ to be a good middle-ground in this regard, as a larger validation split could lead to reduced training data, potentially hindering convergence.

After experimenting with various epoch values, we settled on a value of 200 iterations. This decision was guided by the observation that our best-performing model demonstrated significant convergence in roughly half of this period, showing only marginal improvements after this period. Furthermore our problem, being a binary classification task, is in most regards quite a simple task from a technical standpoint. As such we did not feel that adding more epochs justified the increased training time for such a small return. Additionally, we also implemented early-stopping wherin training would halt after no improvement was made for 10 epochs. This resulted in our observed number of epochs being closer to 75-125.

For our binary classification task, we employed the Binary Cross-Entropy (BCE) loss function. This choice is well-suited to our problem, as it penalizes deviations between predicted probabilities and actual binary labels. Stochastic Gradient Descent (SGD) was chosen as the optimizer due to its simplicity and effectiveness in finding optimal solutions. We experimented with alternative optimizers such as Adam but did not see any significant changes when compared to SGD.

Our learning rate was set at 0.01 after systematic exploration. This learning rate strikes a balance between rapid convergence and stability in the training process. Extensive trials were conducted with learning rates spanning several orders of magnitude, and 0.01 consistently yielded satisfactory results without inducing convergence issues or slowing down training too significantly.

Furthermore, a batch size of 128 was chosen based on considerations of both computational efficiency and model convergence. This size effectively utilizes hardware capabilities and GPU memory while ensuring an appropriate balance between noise introduced by small batches and the potential for faster convergence. The chosen batch size proved effective in achieving stable and efficient training across various experiments and model iterations.

### 4.3 Evaluation

Following training, we loaded each serialized model into an evaluation script (`eval_model.py`) and used our holdout set constructed during preprocessing to evaluate model performance. As with the training script, we used PyTorch's data-loaders to transform our testset samples into the required shape expected by each model. Metrics and graphics were generated using `sklearn` and `matplotlib`, respectively.

## 5 Results

In this section you will find our evaluation results for *Model1*, *ResNet*, and *LetNet* respectively. Further analysis and evaluation of these results follows in the proceeding discussion section.

| Model | Accuracy | Precision | Recall | F1-Score |
|---|---|---|---|---|
| *Model1* | 0.940 | 1.000 | 0.880 | 0.936 |
| *ResNet* | 0.985 | 0.980 | 0.990 | 0.985 |
| *LetNet* | 0.500 | - | 0.000 | 0.00 |

Table 1: Evaluation performance of all three tested models

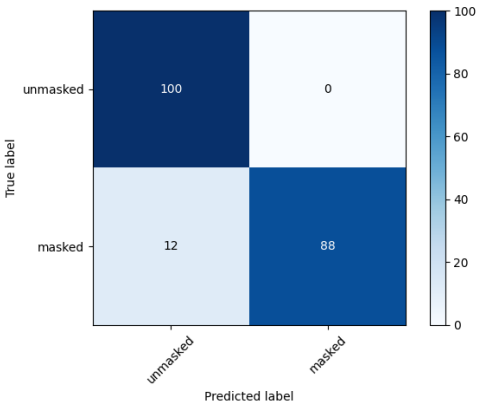


Figure 1: Confusion matrix for *Model1*

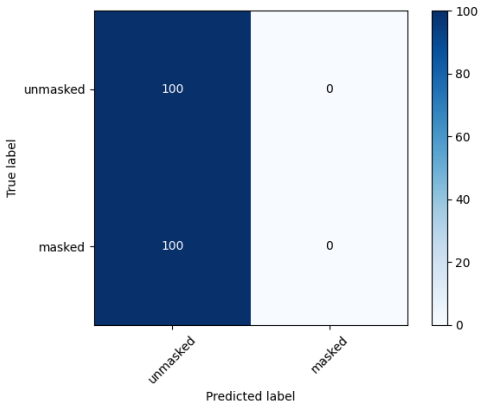


Figure 2: Confusion matrix for *LeNet*

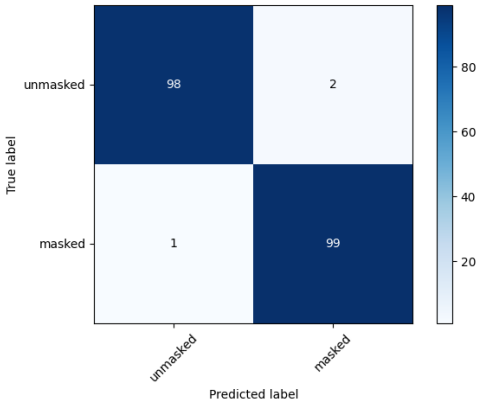


Figure 3: Confusion matrix for *ResNet*

As expected, the modified *ResNet* performed the best with roughly 98% accuracy. Not too far off behind was *Model1* with a respectable 94%. *Model1* showcased impressive precision at a perfect score of 1.00, indicating that every instance predicted as "masked" was indeed masked. However, its recall stood at 0.88, implying that it failed to identify 12% of actual masked instances. Despite this, the model achieved an accuracy of 0.94 and an F1 Score of 0.936, demonstrating a satisfactory balance between precision and recall.

*ResNet* presented stellar results, with nearly impeccable scores across all metrics. Its precision and recall were 0.985 and 0.99 respectively, reflecting an almost flawless prediction capability for the masked class. With an accuracy of 0.985 and an F1 score mirroring the same value, *ResNet* proves its efficacy in this binary classification task.

*LeNet*, on the other hand, encountered challenges in this dataset. Its recall of 0 suggests that it couldn't successfully identify any of the masked instances. This can be inferred from its F1 score,

also standing at 0, indicating a poor harmonic mean of precision and recall. Despite this, the model managed an accuracy of 0.5. This is, however, in all regards unimpressive as you may as well flip a coin to exhibit similar performance (under the assumption of an even distribution as is the case with our processed dataset).

# 6 Evaluation & Discussion

In this section we touch on various aspects of our model results, methodology, and data sampling in further detail, discussing various concerns or insights while suggesting further areas of improvement.

## 6.1 Model Performance

Clearly (and as expected), *ResNet* performed the best out of all tested architectures. This being said, *Model1* arguably held its own against *ResNet*, falling only 4% behind in accuracy. This discrepancy, although measureable, is still less than we had expected. It is likely that a binary classification problem such as ours is simply "easy" enough such that any sufficiently complex model will be able to produce satisfactory results even without much optimization, especially when combined with a large enough dataset as is the case in this project.

With this in mind, the problem was evidently not "easy" enough for *LeNet*. We suspect that the underperformance of *LeNet* is attributed to the architectural limitations of the model, as well as the contrast between the original model's design objectives and the complexity of the task at hand. We believe that being designed around the simpler and more constrained image recognition task of handwritten zip-code detection, *LeNet*'s simply did not have the capacity to properly learn off of our dataset. Unlike its original design intent, the task of mask detection necessitates the identification of comparatively more intricate spatial patterns and features. As such, we can see that due to *LeNet*'s relatively shallow depth in combination with a lower amount of total parameters, its performance ceiling was limited, hence why it opted to "learn" to always predict one class over the other in order to reach a less-than-satisfactory accuracy of 50%. Perhaps if we had used smaller image dimensions, say $(64, 64)$ or $(48, 48)$, we may have seen better results with *LeNet* due to the reasons previously outlined. Of course, this comes at the tradeoff of less information being available to the better performing models, which may reduce their inference capabilities.

## 6.2 Data Sampling

With regards to data sampling for our mask detection task, we made a conscious decision to employ a balanced sampler. As discussed in our dataset explanation, this choice was motivated by the initial overrepresentation of the masked population in the dataset. While it could be argued that the overrepresentation of the positive class might appear justified due to the prevailing context of widespread mask usage during the study period, we ultimately decided to training our model without context of actual real-world distribution of mask-wearers.

In particular, our reasoning for employing "fair" sampling was due to the fact that our goals with this project were more strongly rooted in exploring effective deep learning methods and architectures as opposed to modeling the dataset distribution as accurately as possible, despite its alignment with the mask-wearing trends at the time. That being said, investigating a possible better balance between reflecting real-world distribution and minimizing classification errors would provide an interesting and beneficial extension to our work.

In the course of our final evaluations, we identified a critical oversight regarding the labeling convention. Contrary to the initial documentation, where the *'masked'* class was denoted by the label 1, our processed data had *'masked'* labeled as 0, with *'unmasked'* correspondingly labeled as 1. Despite this inversion, the consistency across all our models and methods ensured that predictions were accurate; however, this impacted the subsequently derived metrics and prompted several re-calculations over additional trials. With the correct label orientation, we recalculated the metrics manually to reflect the genuine performance of the models. While such a deviation may lead some to contend that we inadvertently designed a "no-facemask-detector" rather than a "facemask-detector", it is crucial to underscore that the underlying architecture, training, and model performance remained robust and consistent. The models successfully discerned and classified the presented visual features,

irrespective of the labeling nomenclature. In future work, it would be beneficial to explicitly define strong label conventions in order to mitigate the additional work caused by situations such as this one.

### 6.3 Overfitting

During our training process, we encountered challenges related to overfitting, in which validation performance fluctuated wildly while training errors continued to decrease. As a possible solution, we considered implementing k-means clustering as a strategy to address overfitting. However, while the application of k-means appeared promising, we were limited in terms of time and decided not to employ this strategy for our training. Investigating the effects of this method could provide valuable insights into enhancing model generalization and combating overfitting in complex image classification tasks.

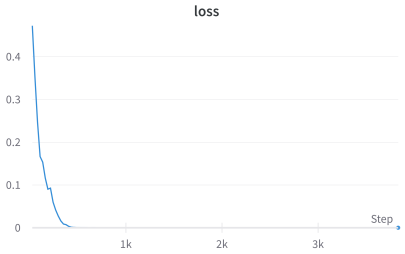


Figure 4: A typical loss chart for *Model1* during training

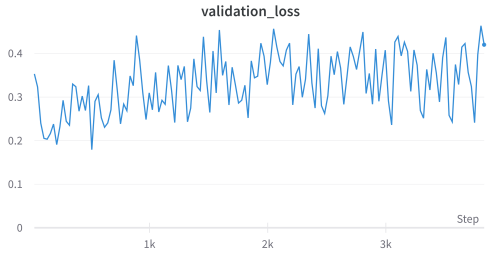


Figure 5: A typical validation loss chart for *Model1* during training

### 6.4 Dataset

As mentioned when discussing our dataset, one of the primary concerns which had come up during initial preprocessing was the lack of clear demographic diversification within the dataset. As part of addressing this concern, we tested our primary model *Model1* on a completely new facemask dataset in order to obtain an unbiased estimate on the model's performance on completely unseen data from a new distribution. The dataset chosen was from Kaggle user `pranavsingaraju`'s "Facemask Detection" dataset, and contained a more diverse set of images from which we randomly sampled 200 masked and unmasked images each. For this experiment, the same evaluation steps were taken as with the original dataset. As it turns out, this concern ended up being less of an issue than we had originally estimated it to be. At roughly 76%, accuracy had only dropped by 13%. Although noticeable, this result is still much better than we had originally estimated and is in the realm of what we consider to be an acceptable loss in performance under the context of a completely new, unseen distribution. We attribute this to be due to the fact that the model learned to focus on identifying masks in terms of contrast rather than in relation to other facial features, thus making the subject's other identifying characteristics less important. It is worth noting, however, that recall was substantially lower. We believe this is due to the fact that the dataset consisted of black and white images in which every subject had a white mask. This may have resulted in less contrast, and as such less prominent feature detection which resulted in the model over-eagerly predicting individuals to not be wearing a face-mask.

## 7 Conclusion

This deep-learning project addresses the critical task of face mask classification using a CNN architecture. By navigating challenges in dataset sampling, model convergence, and overfitting, we successfully developed a satisfactory solution capable of identifying mask-wearing individuals with an approximate accuracy of 94%. Our exploration of methodologies, including balanced sampling and the potential integration of k-means clustering, reflects the conscientious effort to balance real-world trends with model performance. While *LeNet*'s struggles underscore the significance of architectural alignment with the problem domain, our presented CNN approach leveraged more modern deep features to achieve superior results. Overall, this project underscores the potential of deep learning in addressing real-world challenges and the ongoing pursuit of accurate and meaningful solutions.

## References


LeCun, Yann, Léon Bottou, Yoshua Bengio, and Patrick Haffner (1998). In.

Mandal, Bishwas, Adaeze Okeukwu, and Yihong Theis (2021). "Masked Face Recognition using ResNet-50". In: arXiv: 2104.08997 [cs.CV].

Talahua, Jonathan S., Jorge Buele, P. Calvopiña, and José Varela-Aldás (2021). "Facial Recognition System for People with and without Face Mask in Times of the COVID-19 Pandemic". In: *Sustainability* 13.12. ISSN: 2071-1050. DOI: 10.3390/su13126900. URL: https://www.mdpi.com/2071-1050/13/12/6900.

Vinh, Truong Quang and Nguyen Tran Ngoc Anh (2020). "Real-Time Face Mask Detector Using YOLOv3 Algorithm and Haar Cascade Classifier". In: pp. 146–149. DOI: 10.1109/ACOMP50827.2020.00029.