*Is There a Mask on Your Face?*

*Authors: Brian Tonnies, Michael McGaw, Baxter Brown*

Abstract:

COVID-19, otherwise known as SARS-CoV-2 or *the* Coronavirus, has a worldwide

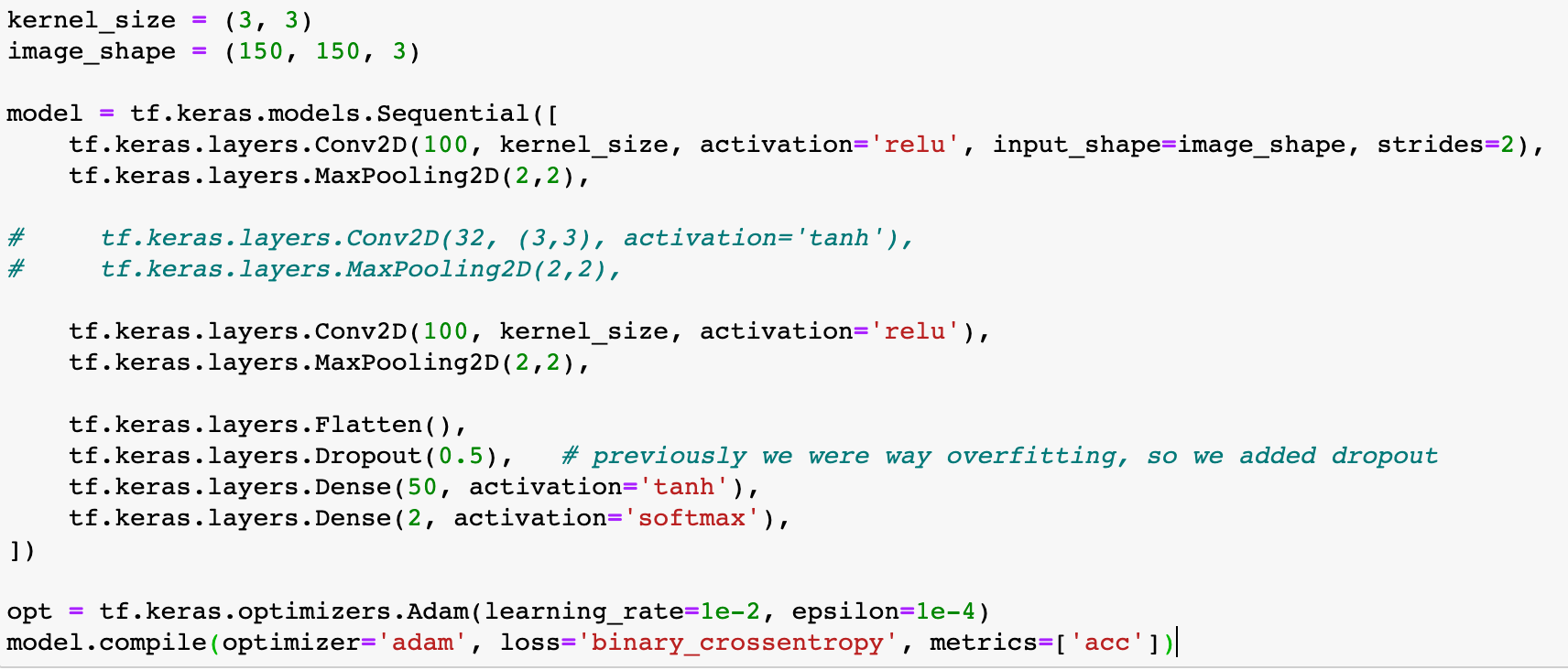
death-toll of 1.04 million and over two hundred thousand lives in the United States of America have fallen victim to it [1]. Our project, *Is There a Mask on Your Face?*, looks to see if We the People are doing our part to prevent the tolls from getting any higher: checking for masks. According to a study at U.C Davis, masks cut your own risk of getting the virus by 65% as well as limit the vectors that you may spread COVID-19 if you are unfortunate enough to have it [2]. The project first aims to detect a face in a given picture, and from there the objective is to determine if the user is wearing a mask and if the user is intelligent enough to use the mask correctly. Using TensorFlow, Keras, layers of Convolution Neural Networks, and OpenCV, this goal was achieved. The model that performed the best out of the several tested was the one with a combination of ReLU activation functions and softmax with a small learning rate of 1e-3. The ultimate goal of this project is to be a litmus test for the common man: will he have decency and respect for life around him, or will he refuse to wear a mask and put others in danger?

Methodology:

Data:

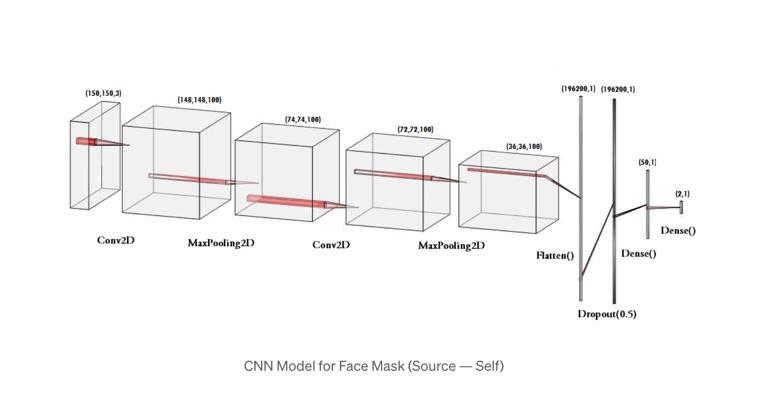
The data was gathered from Kaggle and amounted to over 18,000 images total [3]. The images were split up into training and testing sets: training being 80% of the data image-data, and testing being the remaining 20. Afterwards, the images were grayscaled and “shrunk” down to 150 x 150 pixels. The data was then left for the neural network to ruminate and digest.

Model Creation:

The model was created with the tensorflow Keras API, since “[the] model is appropriate for **a plain stack of layers** where each layer has **exactly one input tensor and one output tensor**.” The ease at which Keras lets you change parameters and save previous models with its attributes greatly attributed to the selection of it. While finding which model worked best with our dataset and goal, we changed the models learning rates (except for the last, softmax), and epsilon value for the Convolution Neural Network. Due to the usage of the “softmax” activation function, the Mean Error Squared calculation was removed and instead loss is determined by cross entropy, as “softmax is technically best used...with *cross entropy*” [4] For comparison purposes, all of the models were trained with equal depths, epochs, and with the “Adam” optimizer, opposed to the normative scholastic classical descent, where Adam is basically a combined form of Adaptive Gradient Algorithm that is optimal for computer vision and Root Mean Square Propagation that is optimal for “noisy” problems (non-stationary objects) [5].

The results given by this model-rubric always over-generalized and over-fit the data, so we created a new model. The difference between this one, which is Model 4, and the 3 others is the order in which dropout, dense, and flatten are called. The old models had the pattern of: flatten, dropout, dense, dense, while the new one had the ordering of: flatten, dense, dropout, dense. Another difference is the image size, which is now 50x50 instead of 150x150.

Diagram of the neural network [6]:



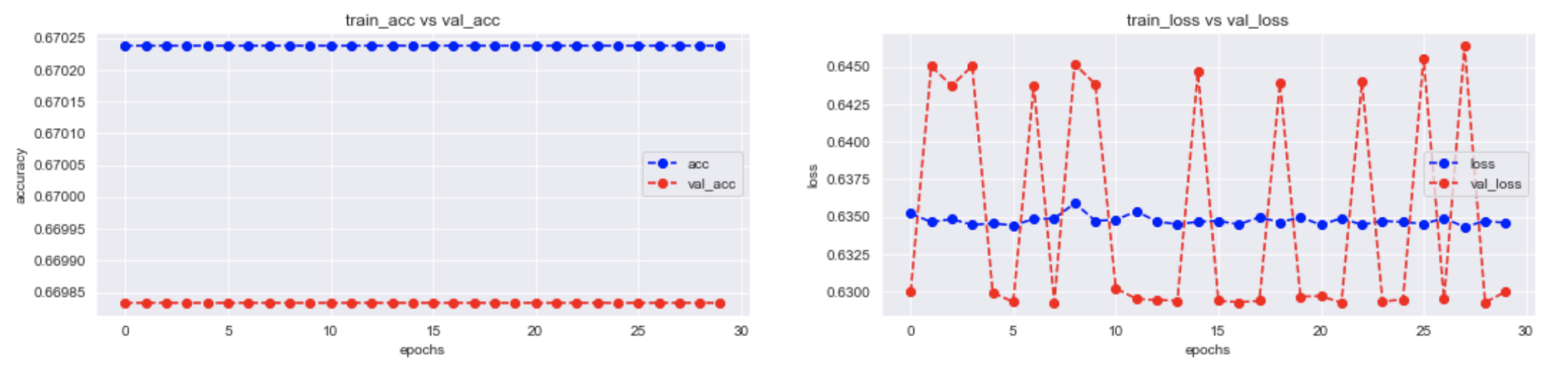
Live-Feed Testing:

Taking the technique used from Hussain Mujtaba, we utilized the MTCNN (Multi-Task Cascaded Convolutional Neural Networks) facial-detection tool to not only detect faces but to also mark key-points such as each eye, corner of mouth, and nose [7]. Another facial-detection tool often used is Haar-Cascades, where the difference in the two is a matter of accuracy versus speed. Using OpenCV to capture each frame given by our webcam, we transformed the frame into something the neural network can ruminate and give feedback to. This method of testing was used as a pseudo-cross-validation technique to check whether the model was overfitting.

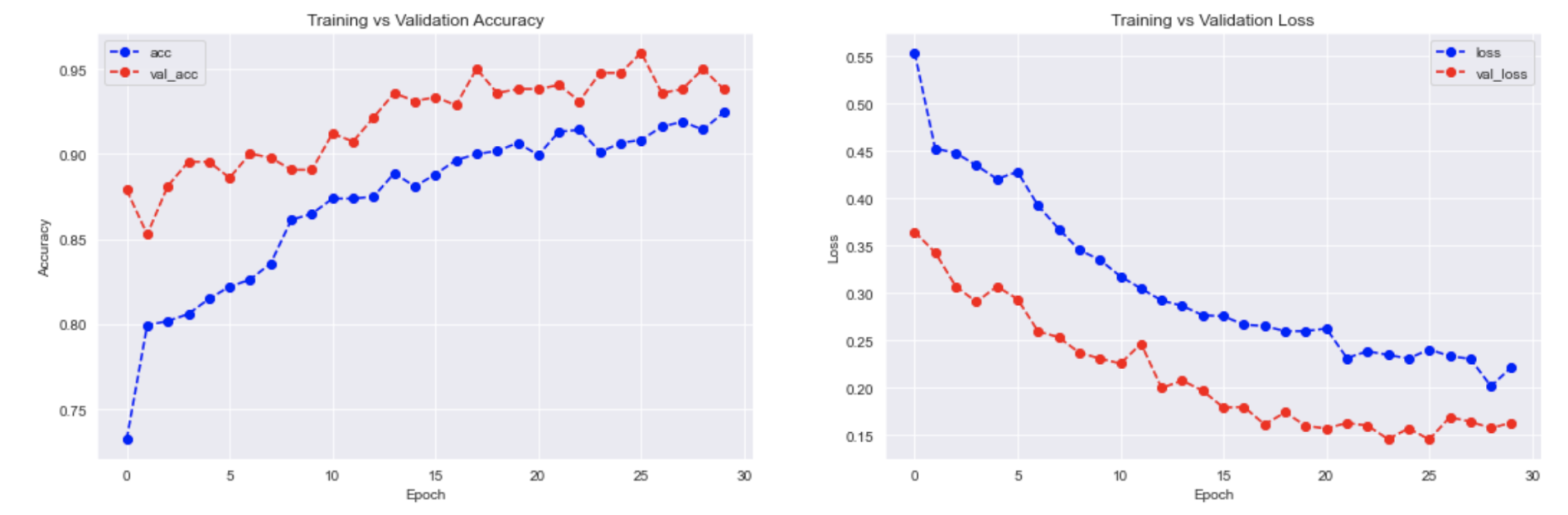
Results:

We calculated the “best” model by evaluating the loss and the accuracy values: loss being the summation of errors made by the model and accuracy being a percentage of correctly versus incorrectly identified masks.

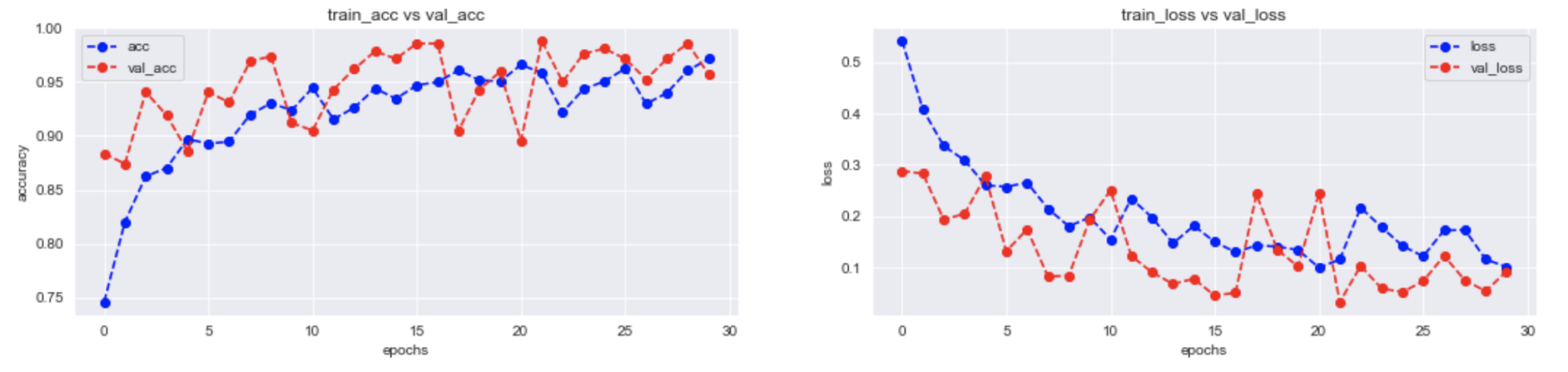
Model Attribute: Learning Rate of 3e-4 with Sigmoid and Softmax Activations



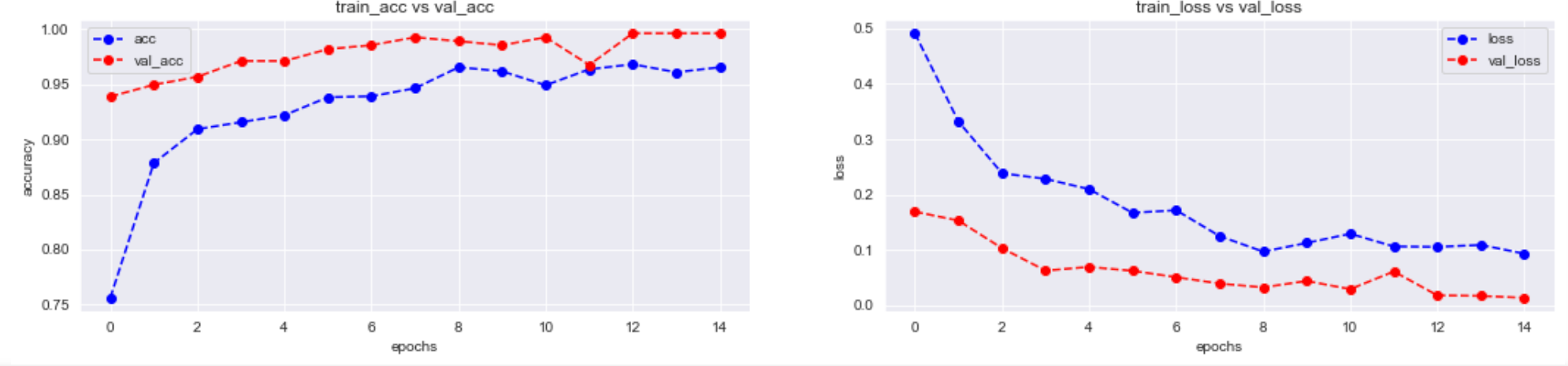
Model Attribute: Learning Rate of 3e-4 with TANH and Softmax Activations



Model Attributes: Learning Rate of 3e-4 with ReLU and Softmax Activations



Model Attributes: Learning Rate of 1e-4 with Relu and Softmax Activations



Best Fit Model: Learning Rate of 1e-4 with Relu and Softmax Activations

As can be seen from the images above, the model that performed the best is the fourth model, the one with the ReLU activation function and a learning-rate value of .0001, with an accuracy of 0.9964 and a loss value of 0.0134. This model, we’ll call it Model 4, was the only model created with different layer patterns compared to the previous 3 and is also the only model that didn’t seem to overfit the data. We believe this is due to the dropout method being called right before the results are output so that the predictions are much more “tampered” down. Prior to this, in the live-feed testing, all of the models were either continually predicting “mask on” or failing to recognize that a face was in the frame . Model 4 is the only model that can accurately (comparative to the other models) tell whether or not there’s a mask on a face. Model 4 is still far from perfect however, and given the comparison between the accuracy on the dataset (99%), and the accuracy of the live-feed data (far less), the model could definitely use more training on an even larger scale.

Discussion:

Many models were torturously trained in search for the most optimal attributes for mask detection, but the models with the best accuracy and loss calculations were always the models that had the activation function of relu and softmax -- although tanh looked like it was putting up a fight at first. There are many things that can be adjusted and fine-tuned within a Convolution Neural Network and many of the adjustments were made, but for future work we would like to try a different Keras model: instead of using Sequential, we would instead use MobileNetv2. The results of the live-feed test give an indication that the other models were either victims of overfitting to the data or simply trying to hurt my self-esteem by refusing to recognize my face as a face. Another change that could be beneficial if explored is the changing of epochs; perhaps a shorter epoch would cause the model to overfit less. Another avenue of approach would be to use a different optimization rather than Adam, as “[Adam] often leads to worse generalization performance than SGD” whenever the neural network is deep [8]. This generalization would explain the overfitting mountain our previous models couldn’t climb. Overfitting very nearly tanked our entire enterprise. The number of different sources and blogs that have attempted something similar to this project is staggering. Their approaches are incredibly similar -- as is our own. More than once we tried to recreate the results that these sources had claimed. Anecdotally, it seems that they actually suffered the same overfitting nightmare that we did even when their published results indicated otherwise.

A future for this project would be a future where the Convolution Neural Network not only detects masks, but also recognizes distinctions between correctly or incorrectly worn masks, beards, or facial-tattoos. Another avenue this project could eventually approach would be determining distances between people (given their faces), and from there, based on whether or not the person(s) were wearing masks, whether or not a possible exchange of viruses occured. There is also a likelihood that, given the accuracy of the model in regards to face detection, it can be used to identify more than just *faces*, but perhaps many things in a photo and could be used to pick out objects in a given Captcha and bypass its anti-bot security.

### Bibliography[1] :

[1] *Coronavirus Death Toll*. [https://www.worldometers.info/coronavirus/coronavirus-death-toll/.](https://www.worldometers.info/coronavirus/coronavirus-death-toll/)

[2] Fell, A., & Staff, D. (2020, July 10). *Your Mask Cuts Own Risk by 65 Percent*. UC Davis. <https://www.ucdavis.edu/coronavirus/news/your-mask-cuts-own-risk-65-percent/>.

[3] Intelligence, Wobot. Face Mask Detection Dataset. 14 June 2020, www.kaggle.com/wobotintelligence/face-mask-detection-dataset.

[4] Trask, A. W. (2019). Pg. 174. In *Grokking deep learning*. Shelter Island: Manning.

[5] Brownlee, Jason. “Gentle Introduction to the Adam Optimization Algorithm for Deep Learning.” *Machine Learning Mastery*, 20 Aug. 2020, machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/.

[6] K, Gurucharan M. “COVID-19: Face Mask Detection Using TensorFlow and OpenCV.” *Medium*, Towards Data Science, 9 June 2020, [towardsdatascience.com/covid-19-face-mask-detection-using-tensorflow-and-opencv-702dd833515b.](https://towardsdatascience.com/covid-19-face-mask-detection-using-tensorflow-and-opencv-702dd833515b)

[7] Hussain Mujtaba, et al. *Real-Time Face Detection: Face Mask Detection Using OpenCV*. 28 Oct. 2020, [www.mygreatlearning.com/blog/real-time-face-detection/.](http://www.mygreatlearning.com/blog/real-time-face-detection/.)

[8] Zhang, Zijun. “Improved Adam Optimizer for Deep Neural Networks.” *Improved Adam Optimizer for Deep Neural Networks - IEEE Conference Publication*, 24 Jan. 2019, [ieeexplore.ieee.org/abstract/document/8624183](https://ieeexplore.ieee.org/document/8624183).

[9] <https://github.com/brtonnies/face-mask-detection>: the repository for the project.