Project Paper

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 [2]. 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 [3]. 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, we expect the results to show either the *direct* correlation to the amount of blood, sweat, tears, and excrement put into this project or, at the very worst, we expect the project to *still* detect a face even after the possible convoluted nets become added. 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:

Dataset:

The dataset, consisting of six-thousand-twenty-four images with arbitrary faces, is dissected and split into fifteen thousand four hundred twelve photographs: one photograph for every *singular* face. The images are then reshaped to a fifty-pixel by fifty-pixel image. The faces are then segregated into feature-sets, i.e “face\_no\_mask”, “mask\_surgical”, and “mask\_colorful”. Of the twenty available features/classes, the current model is focused on only two: face\_with\_mask and face\_no\_mask.

Model Creation [For Face Detection]:

The model chosen was Keras Sequential Model since, as the documentation says, “[the] model is appropriate for **a plain stack of layers** where each layer has **exactly one input tensor and one output tensor**.”[x] The Convolutional Neural Network takes tensors as input, and the Keras API has parameters for stride (the x\*x amount you want the “window” to move) when creating the model, a Flatten method to *flatten* out the x\*x array created. The *learning rate* and *epsilon* are set to 1e-3 and 1e-5 for the optimizer Adam, which is an “optimization as a stochastic gradient descent method that is based on the adaptive estimation of first-order and second-order moments.” The model utilizes the *softmax* activation function, which “encourages the network to predict ***one*** output with a very high probability”, which is exactly what we want: an either-or prediction. The purpose of loss, which was set to “categorical\_crossentropy”, is to determine the quantity that a model should seek to minimize during training. The categorical\_Crossentropy class computes the cross-entropy between the labels and the predictions. The Dropout is set to twenty percent to prevent overfitting within the model. The density method, according to the documentation, “implements the operation: output = activation(dot(input, kernel) + bias) where activation is the element-wise activation function passed as the activation argument, the kernel is a weights matrix created by the layer, and bias is a bias vector created by the layer (only applicable if use\_bias is True).”[x] After the training was completed, the model was saved and can be later applied to future work.

Results and Very-Near-Future-Work:

As our project has yet to be completed, the results are almost lacking. The model gives an accuracy of 97.36% and a loss of .0678%; however, at the time of writing this, there have been no cross-validation methods utilized on the model. We will be using k-Folds Cross-Validation, and perhaps an even larger data-set to retrain the model with a larger dataset, brought to us by Kaggle, of one hundred thirty-seven thousand images. Given that dataset, a k value of 10 for k-Fold would be an appropriate size for the model.

Discussion:

If given the grant-money and time, the project could be aimed towards detecting not just surgical masks but any sort of mask that covers the face and even determine if the facial-covering is correctly or incorrectly. Another potential application is determining the distance between two people (faces) in an image, whether or not they’re wearing an appropriate face covering, and scoring the respective behaviors based on their likelihood to propagate and/or spread COVID-19. There is also a likelihood that, given the accuracy of the model, 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; however, I do not want to imply that this can only be used for images. An interesting prospect of this project is to design it to work for live-feed footage, which just in the lens of the medical field is very important, especially given the ramifications Sars-19 has had on America. Much of this work would mean that the structure of our models will need to change. As far as activation functions, softmax is best at predicting “which one” of a selection of choices is most likely to be correct. If we have several potential correct predictions, then the softmax function is going to overfit early on and our model’s predictive power will end up being reduced. Further, wiithout softmax, the cross-entropy loss function is also much less appropriate.

Just within the scope of computer vision alone, there are many potential applications of simple neural networks to draw relatively minor conclusions that can lead to larger predictive power for any number of scenarios. It is easy to see that, with this project, *Is There a Mask on Your Face?* we have come one step closer to monitoring every-day citizens in their everyday shenanigans, and are truly becoming the utopia George Orwell envisioned in *1984*. Perhaps even more so we are the animals of *Animal Farm*.

### Bibliography[1] :

[1] N. Bai. Still Confused About Masks? Here’s the Science Behind How Face Masks Prevent Coronavirus. (June 26, 2020). Retrieved October 4, 2020 from<https://bit.ly/3ir8YT3>

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

[3] 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] Cabani, A., & Hammoud, K. *cabani/MaskedFace-Net*. <https://github.com/cabani/MaskedFace-Net>. → this is hosted on sharepoint, we have not accessed the dataset it contains yet.

others:

<https://www.kaggle.com/jainamshah17/face-mask-detection-using-faster-r-cnn-pytorch> (the pytorch implementation -- have not done this yet)

<https://www.mygreatlearning.com/blog/real-time-face-detection/> (kinda says it all, eh?)

<https://www.mygreatlearning.com/blog/viola-jones-algorithm> (this is the algorithm that uses the haar cascades)

<https://www.mygreatlearning.com/blog/face-recognition> (just another article related to facial recognition/opencv/viola-jones algo)

<https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_objdetect/py_face_detection/py_face_detection.html> (how opencv uses haar cascade detection for facial/feature recognition)

<https://keras.io/api/layers/convolution_layers/convolution2d> (keras documentation for the type of CNN our keras models use -- they refer to it as a ‘layer’)

<https://keras.io/api/layers/pooling_layers/max_pooling2d> (keras documentation for the max pooling layer for 2D spatial data)

<https://keras.io/guides/sequential_model/> (keras documentation for our base model/layer: the sequential model)

<https://keras.io/api/layers/core_layers/dense> (keras docs for the dense nnet layers in our model definition):

Dense implements the operation: output = activation(dot(input, kernel) + bias) where activation is the element-wise activation function passed as the activation argument, kernel is a weights matrix created by the layer, and bias is a bias vector created by the layer (only applicable if use\_bias is True).

NEED SCHOLARLY SOURCES

P. 174 of ‘Grokking’ -- “softmax is best used with an error function called *cross entropy…..*” -- we do that!