Vikas Kharkwal, 2021, 9:2
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

Mask Detection Using Convolutional Neural Network

Vikas Kharkwal, Tenzin Jamtsho, Brajehdra Kumar Sharma, Yogeshwar Maithani
Computer Science and Engineering,
Tulas Institute, Dehradun,,Uttarakhand
vikas.201704075@tulas.edu.in, tenzin.201704078@tulas.edu.in, brajehdra.sharma@tulas.edu.in,
yogeshwar.201704069@tulas.edu.in

Abstract- The world is going through a tough time as we are in the middle of a pandemic caused by the corona virus (SARS- CoV-2). The guidelines issued by the WHO indicate the use of face masks in public for preventing the spread of the virus. The project aims to train a neural network to detect whether a person is complying with the requirement for a mask. This tech can be applied in entrances of high traffic areas such as metros and airports, which have the highest risk of transmission, to determine whether or not to allow entrance. This will help officials to monitor the people while maintaining a safe distance and preventing them from getting infected.

Keywords: - Face Mask Detection, Convolutional Neural Network, Data augmentation, Algorithm.

I. INTRODUCTION

The year 2020 has been really tough on mankind. With the rapid spread of COVID-19, it reached a pandemic status, affecting millions of people in each and every part of the world. It forced people to make many lifestyle changes like staying at home, observing social distancing, sterilizing the surrounding and the most important and controversial lifestyle change, use of face masks. It has been proven that wearing a mask prevents the spread of the disease considerably, but scepticism spread far and wide.

It was mainly due to pseudoscience and fake news peddlers spreading false information about the virus as well human body on how it reacts to wearing masks. Even if there are scientists, doctors and health workers vouching for face masks being a great source of prevention of this disease, there were still many people making cases against wearing of face masks citing pseudoscientific articles.

Though we cannot stop people from believing in pseudo- science and fake news, we can however enforce wearing of masks in public spaces to slow the spread of this deadly disease. This might be hard in some cases as people tend to wear masks only when they see someone monitoring them. To monitor the people to ensure proper following of rules without making it obvious that they are being monitored, a method needed to be developed.

One that leads to least amount of exposure to workers working in these tough times to enforce the rules and regulations, especially regarding the wearing of face mask To enforce the rule of mask wearing, it was not possible to hire a large number of people just to check if the people are wearing masks in public spaces. This is where our project was envisioned.

II. METHODOLOGY

1. Dataset:
The dataset we used is from Kaggle by the user Larxel. The dataset contains 853 images with multiple faces in many images. We also manually extracted images from Google search to get good quality images for our data set. We used validation split to split the data into training and validation data. The data was split in the ratio of 0.7:0.3, i.e., 70 percent of the data was used for training the model and 30 percent was reserved to validate the model after

training. The reserved data was unseen by the model so it told us the accuracy of the working of the model for unseen data.

2. Data Manipulation:

2.1 Image Libraries:

2.1.1 Pillow: Pillow is the Python Image Library (PIL) for versions 3.x. It adds the functionality of opening, manipulating and saving images in python. We used the Image module of Pillow to open and manipulate the images in Python. We extracted the faces from the images using Pillow. The images weren't converted to grayscale like many other image processing models because it's hard to detect faces behind a mask and we require skin color to detect a face. The blob detection method of CV2 that we aimed at using would require color data to identify faces behind a mask using the skin tone of the person.

2.1.2 CV2 (OpenCV): OpenCV is a library aimed at real-time computer vision (it deals with how computers understand digital images or videos). We used CV2 of OpenCV library for Python to capture live video feed from the webcam. The image was saved as a constantly updating temporary file.

CV2 uses BGR colour space, which is converted to RGB when the image is opened and manipulated using Pillow. This helped us in maintaining compatibility between training dataset and real-world usage of the model because the images we used to train the model were all in RGB colour space due to it being the most popular colour space in computing. This library was also used for blob detection which is a method to identify regions in an image that differ in properties, like brightness and colour. This made it easier for our model to detect faces behind masks.

2.2 NumPy:

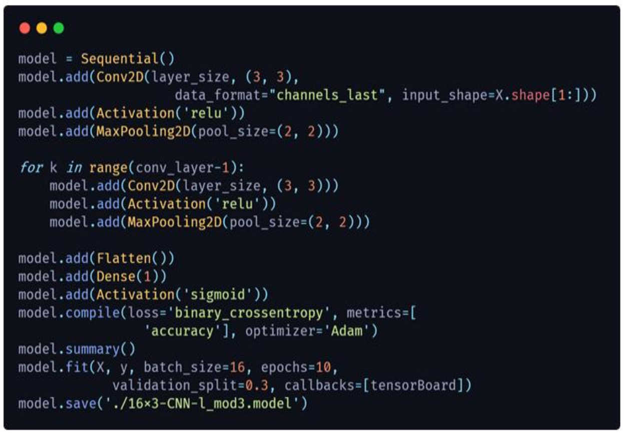
We used NumPy to convert the images manipulated using Pillow into a multidimensional array so that the data can be understood by the computer. A computer does not understand images as we humans do. To make the computer understand the images, we use NumPy to convert the RGB image into a 3-dimensional array of h × w × d (h = height; w = width; d = dimension). E.g., 255 ×255× 3 array of an RGB image.

3. Core Library Used for Generating the Model (Keras):

We used Keras library to generate our model. Keras is a library in Python that provides an interface for Artificial Neural networks. It acts as an interface for the Tensor Flow library. It contains implementations of commonly used neural-network building blocks like layers, activation functions, optimizers, etc. and make it easier to create machine learning models without writing excessive neural network code.

III. ARCHITECTURE OF THE MODEL

We trained a CNN model with 3 layers of 8 nodes each, no hidden dense layer and one bias node. The code below is the model after removing the model structure.



```python
model = Sequential()
model.add(Conv2D(layer_size, (3, 3),
                 data_format="channels_last", input_shape=X.shape[1:]))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

for k in range(conv_layer-1):
    model.add(Conv2D(layer_size, (3, 3)))
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dense(1))
model.add(Activation('sigmoid'))
model.compile(loss='binary_crossentropy', metrics=[
               'accuracy'], optimizer='Adam')
model.summary()
model.fit(X, y, batch_size=16, epochs=10,
            validation_split=0.3, callbacks=[tensorBoard])
model.save('./16×3-CNN-l_mod3.model')
```

Fig 1. Code snippet showing the Sequential model.

A Sequential model is appropriate for a plain stack of layers where each layer has exactly one input tensor.

- The first layer is a Convolution Layer of size 3 × 3. This layer creates a convolution kernel that is convolved with the layer input to produce a tensor of outputs. If use bias is true, a bias vector is created and added to the outputs. Finally, if activation is not any, it is applied to the outputs as well. Channels Last: Image data is represented in a three-dimensional array where the last channel represents the colour channels, e.g., [rows] [cols] [channels] this layer uses the 'ReLU' activation function.

- The next layer performs a max pooling operation on output of the previous layer. It down-samples the input representation by taking the maximum value over the window defined by pool size for each dimension along the feature's axis. The window is shifted by strides in each dimension. This layer has a pool size of (2 2).

- The resulting output when using"valid" padding option has a shape (number of rows or columns) of:

Output shape = ((input shape − pool size) + 1)

- The above-mentioned Convolution and Max Pooling operations are done in steps so as to avoid any feature loss. We have determined by consecutive tests that our model works with most accuracy with two additional steps of Convolution and Max Pooling.
- The next layers flatten the 2D output of the previous layer into a 1D array which can then be fed to a Dense Layer.
- The last layer is a Dense Layer. 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). This layer uses the 'sigmoid' activation function. This is our output layer.

IV. OPTIMIZATION OF MODEL

1. Optimizer/Gradient Descent Algorithm:

Adam is an adaptive learning rate optimization algorithm that utilizes both momentum and scaling, combining the benefits of RMS prop and SGD with Momentum. The optimizer is designed to be appropriate for non-stationary objectives and problems with very noisy and/or sparse gradients [1].

The weight updates are performed as:

$$w_t = w_{t-1} - \eta \frac{\widehat{\mathbf{m}}_t}{\sqrt{\widehat{\mathbf{v}}_t + \epsilon}}$$

$$\widehat{\mathbf{m}}_t = \frac{m_t}{1-\beta_1^t}$$

$$\widehat{\mathbf{v}}_t = \frac{v_t}{1-\beta_2^t}$$

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$

η is the step size/learning rate, around 1e 3 in the original paper. ϵ is a small number, typically 1e 8 or 1e 10, to prevent dividing by 0. β_1 and β_2 are forgetting parameters, with typical values 0.9 and 0.99, respectively.

2. Activation Functions:

We considered using ReLU which outputs the function itself if it is positive, but gives zero as output if it is anything else. We also used sigmoid function because it takes real values for input and outputs another value between 0 and 1. It works better when creating models for binary classification. We didn't use tanh function as it gives output over a broader range, i.e., −1 to 1, which is not ideal in our case.

2.1 ReLU: The rectified linear activation function or ReLU for short is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero. It has become the default activation function for many types of neural networks because a model that uses it is easier to train and often achieves better performance. [3]

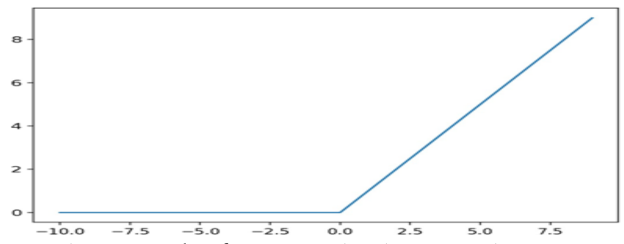


Fig 2. Graph of ReLU Activation Function [4].

2.2 Sigmoid Function:

$$\boldsymbol{Sigmoid(x)} = \frac{\mathbf{1}}{(\mathbf{1} + \boldsymbol{exp(-x)})}$$

Applies the sigmoid activation function. For small values (<5), sigmoid returns a value close to zero, and for large values (> 5) the result of the function gets close to 1.

Sigmoid is equivalent to a 2-element Soft Max, where the second element is assumed to be zero. The sigmoid function always returns a value between 0 and 1.

Thus, it is used for binary classifications; we don't use tanh function which outputs between 1 and 1 because images don't have a negative value.

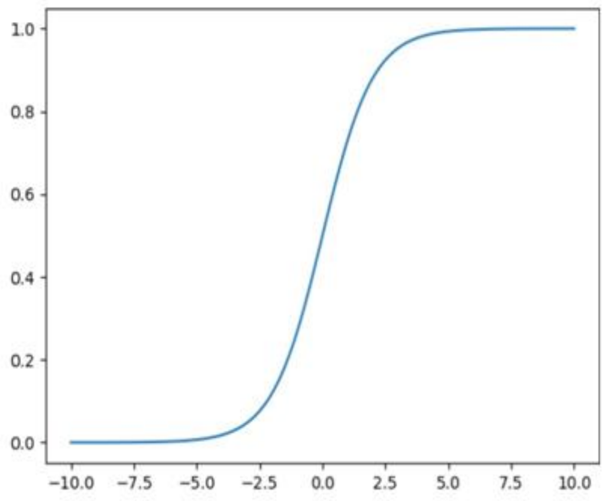


Fig 3. Graph of Sigmoid Activation Function [5].

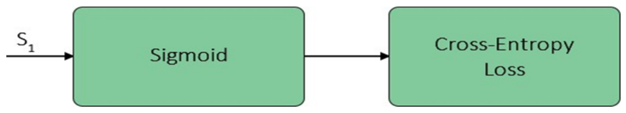
3. Loss Function:

3.1 Binary Cross-entropy: Also called Sigmoid Cross- Entropy loss. It is a sigmoid activation plus a Cross-Entropy loss. Unlike Soft Max loss it is independent for each vector component (class), meaning that the loss computed for every CNN output vector component is not affected by other component values.

That's why it is used for multi-label classification, where the insight of an element belonging to a certain class should not influence the decision for another class. It's called Binary Cross-entropy Loss because it sets up a binary classification problem between C′=2 classes for every class in C, as explained above.

So, when using this Loss, the formulation of Cross-entropy Loss for binary problems is often used:

$$CE = \sum_{c'=2}^{i=1} ti \; log(f(si))$$



$$CE = -t_1 \log (f(s_1)) - (1 - t_1) \log (1 - f(s_1))$$

$$f(s)_i = \frac{1}{1 + e^{-s_i}}$$

3.2 Validation Loss: Validation loss is the most important value to look at when optimizing the model. Validation loss is the same metric as training loss, i.e., cost function, but it is not used to update the weights. It is a key factor in determining whether a model is capable when dealing with data outside of the training set. And the most important value to have a consistent decline in as you train.

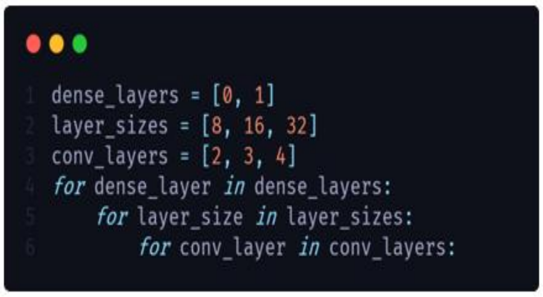
4. Tensor Board:

4.1 About Tensor Board: It is the visualization toolkit of Tensor Flow. It is an extremely helpful tool used when optimizing the model.

4.2 Approach: With the use of Tensor Board we can compare multiple trained models at a time using the method below. We consider the different possible values for each of the following:

- Number of dense layers
- Number of convolution layers
- Number of nodes

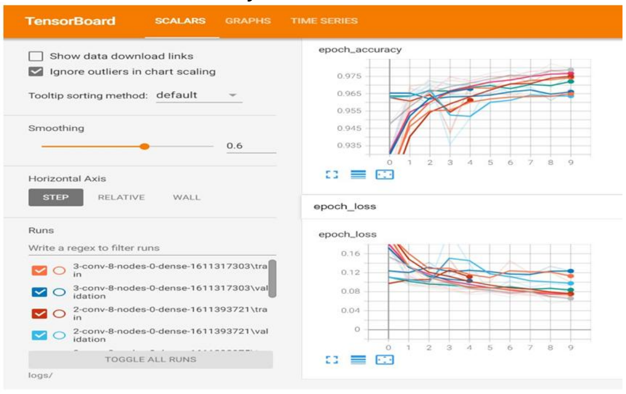
Then we created arrays of the values and used nested loops to train the model multiple times.



```
dense_layers = [0, 1]
layer_sizes = [8, 16, 32]
conv_layers = [2, 3, 4]
for dense_layer in dense_layers:
    for layer_size in layer_sizes:
        for conv_layer in conv_layers:
```

Fig 4. Code snippet for training models for comparison.

The accuracy and loss from each model were analyzed using Tensor Board and the model with best trends in accuracy and loss was chosen.



☐ Show data download links
☑ Ignore outliers in chart scaling
Tooltip sorting method: default
Smoothing 0.6
Horizontal Axis: [✓ STEP] RELATIVE WALL
Runs
Write a regex to filter runs
☑ 3-conv-8-nodes-0-dense-1611317303\train
☑ 3-conv-8-nodes-0-dense-1611317303\validation
☑ 2-conv-8-nodes-0-dense-1611393721\train
☑ 2-conv-8-nodes-0-dense-1611393721\validation
logs/

Fig 5. Tensor Board to compare accuracy and loss of trained models.

5. Feature Map:
The feature map is the output of one filter applied to the previous layer. A given filter is drawn across the entire previous layer, moved one pixel at a time. Each position results in activation of the neuron and the output are collected in the feature map.

Convolutional Neural Networks look for features such as straight lines, or cats. As such whenever you spot those features-they get reported to the feature map. A feature map helps gain insight on the CNN when determining the right filter size pool size when spooling.

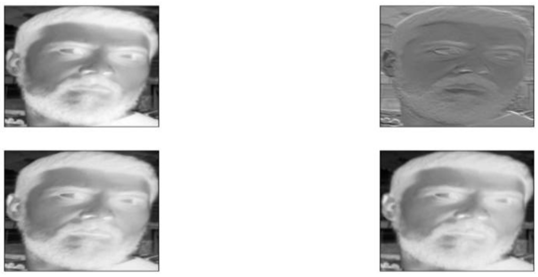


Fig 6. Generated feature map of a face.

V. INFERENCE

We verified the working of the model on live video feed using our webcam and found it to be working as expected. The model works well as it gives Not Wearing Mask as a response when you cover your mouth with your hand or have uncovered your nose while wearing a mask.

This is great as it can differentiate when someone is using their hands to deceive the model or wear the mask improperly. Some screenshots can be seen below which show the working of the model. It also identifies multiple people in the frame at the same time. Below are some screenshots showing the working of the model:

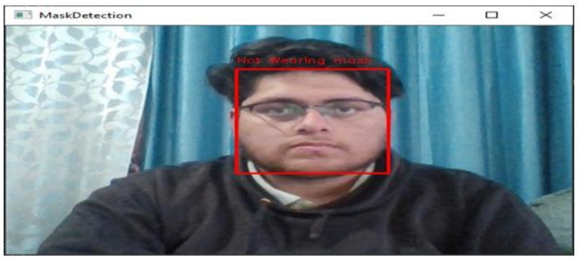


Fig 7. When not wearing a mask.

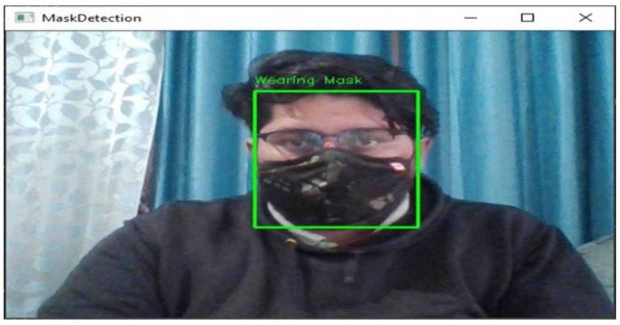


Fig 8. When wearing a mask.

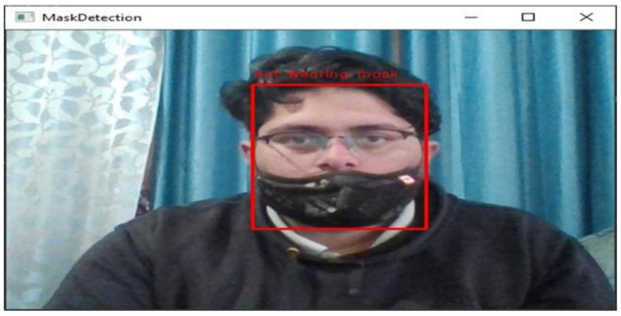


Fig 9. When wearing mask improperly.

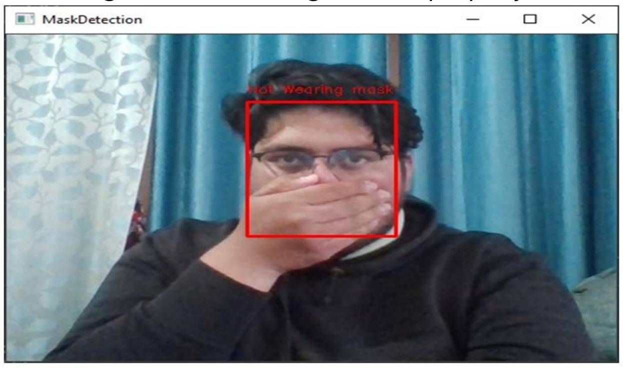


Fig 10. When covering mouth with hand.

1. Limitations:
The only limitation we faced was that of bad data. The data was insufficient for masks worn incorrectly which caused us to scrap that part of the project entirely. However, after training, the model would respond with Not Wearing Mask when the mask was not covering the nose, so that part of the problem was taken care of itself.

2. Future Expansion:
Major nations in the world made it mandatory to wear masks in public. Whether it is in transport stations, shopping malls or in markets, people are required to wear masks. This should have been easy but humans are rebellious and won't follow rules easily. The model we created can be expanded to be used in any public space using the CCTV footage as the video feed. This will allow authorities to use existing infrastructure with very little modification to

prevent unmasked people from entering the public space and increasing the risk of contamination. The authorities can be alerted in case someone without a mask enters the premises of the public space.

VI. CONCLUSION

In this world that is crumbling over rising cases of COVID- 19, using technology in any way to help humanity is something a student of Computer Science must aspire. The project we developed to help the frontline workers to enforce rules which are beneficial for safety of everyone can be of great use and can help prevent the spread of the deadly disease we are facing currently.
This project can be integrated with surveillance systems in order for it to be useful in major public spaces.

We used Pillow, NumPy, OpenCV, Tensor Flow Keras, etc. to detect proper use of face masks by people in public spaces. The model was trained and validated with images from Kaggle and a few images which were manually downloaded to have a quality data set. The final testing of the finished model was done using similar quality images as well as using the live video feed from webcam of our device.

The model can easily detect multiple people in an image or in the live video feed and detect whether they are wearing masks. The model is well optimized so it can be implemented in embedded systems. The model can be used in public places like transport stations, shopping malls, etc. to ensure people are wearing proper masks without the need of extra "eyes" in the authority, thus keeping risk of contamination to the minimum.

REFERENCES

[1] Dataset-https://www.kaggle.com/andrewmvd/face-mask-detection/, 02/11/2020.

[2] Diederik P. Kingma and Jimmy Ba (2014) 'Adam: A Method for Stochastic Optimization', https://arXiv.org/abs/1412.6980

[3] Ian H. Witten, Eibe Frank, Mark A. Hall, Christopher J. Pal (2017), 'Chapter 10 - Deep learning', Data Mining (Fourth Edition), Morgan Kaufmann, pp 417-466

[4] https://machinelearningmastery.com/rectified-linear-activation-function-fordeep-learning-neural-networks/, 14/01/2021

[5] https://www.geeksforgeeks.org/implement-sigmoid-function-using-numpy/, 19/01/2021