Face Mask Detector using Conventional Neural Network (CNN)

Herve Ngomseu Fotsing (40067741) - Data Specialist
Isaac Oluwole Olukunle (40163296) - Training Specialist
Akinlolu Ojo (40190767) - Evaluation Specialist

COMP 6721 - Applied Artificial Intelligence Project Report, Part 1

Group: AK_4

June 8^{th}, 2022

1 Introduction

Deep learning is a trending field of study and has been used heavily used in computer vision and image processing. With the event of the COVID–19 pandemics, there has been an extensive amount of research in the field with different implementations. It is well–known that CNN is the preferred mechanism for image processing nowadays and face mask detection technologies are increasingly being deployed at public entrances to maintain safe working standards. Conventional machine learning techniques have been proved to be less efficient compared to deep learning and the availability of large datasets have favoured the growth of CNN in this domain. This study aims to demonstrate an end–to–end pipeline implementation of a Convolutional Neural Network (CNN–based) Face Mask detector to classify five categories of face mask detection, namely:

- A person without a face mask.
- A person with a "cloth" face mask.
- A person with a "surgical" mask.
- A person with a "FFP2/N95/KN95"-type mask.
- A person with a mask worn incorrectly.

To this end, we collect training data based on the specifications as elaborated in section 2. We construct the model in section 3, and finally, we present the simulation setup and the evaluation results in section 4.

2 Dataset

The training data set consists of 2000 images collected from internet open sources. We created a balanced dataset consisting of equal number of images for each of the five categories of face mask detection. We, therefore, have 400 images per class collected from various sources as shown in Table 1. The five dataset sources in the table are labeled Source 1 [4], Source 2 [7], Source 3 [6], Source 4 [5], and Source 5 [8, 9].

Table 1: Distribution of training images

Categories	**Source 1**	**Source 2**	**Source 3**	**Source 4**	**Source 5**	**Total**
No Mask	0	0	0	89	311	400
Cloth Mask	68	114	125	36	57	400
Surgical Mask	0	0	85	136	179	400
FFP2/N95/KN95 Mask	74	180	43	50	53	400
Incorrect Mask	0	0	0	0	400	400

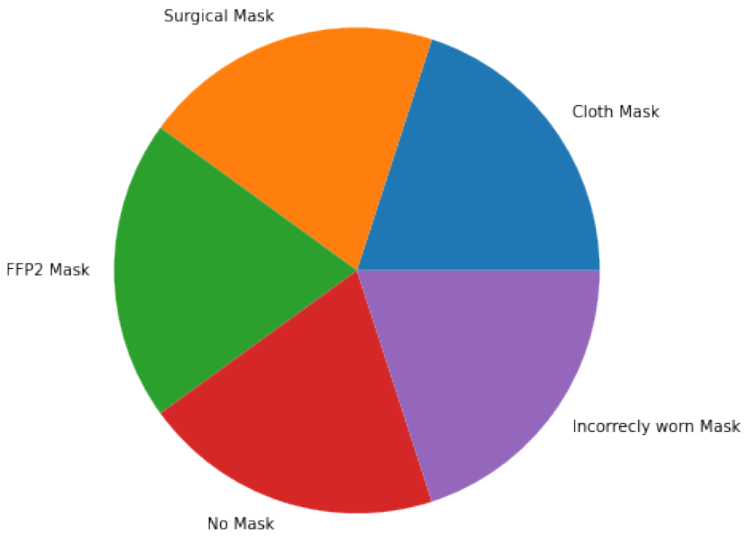

Figure 1: Dataset distribution.

After collecting the images, the dataset was split into a training set (75%) and a testing set (25%). A balanced data set allocation as illustrated in Figure 1 in the CNN–model is done to achieve the best evaluation performance. We remove any corrupt files from the data set folder. During the pre–processing stage, the images are all reshaped to a fixed–size of 64x64. Next, the images are transformed to grey scale which generates higher classification

performance than RGB colour format using a shorter amount of training time. The images are then normalised based on the mean and standard deviation of the data set. This is done by processing all images and calculating the mean and standard deviation manually using a NumPy array.

3 CNN Architecture

The CNN architecture used in this project is a network made up of three convolutional layers followed by three linear layers. We arrived at this architecture choice by experimenting with the various number of layers, testing the accuracy of these network architectures, and selecting the best one. In the set of convolutional layers we apply LeakyRELU activation function to introduce some non-linearity to the network. Also in the linear layers, we apply the ReLU activation function to introduce non-linearity.

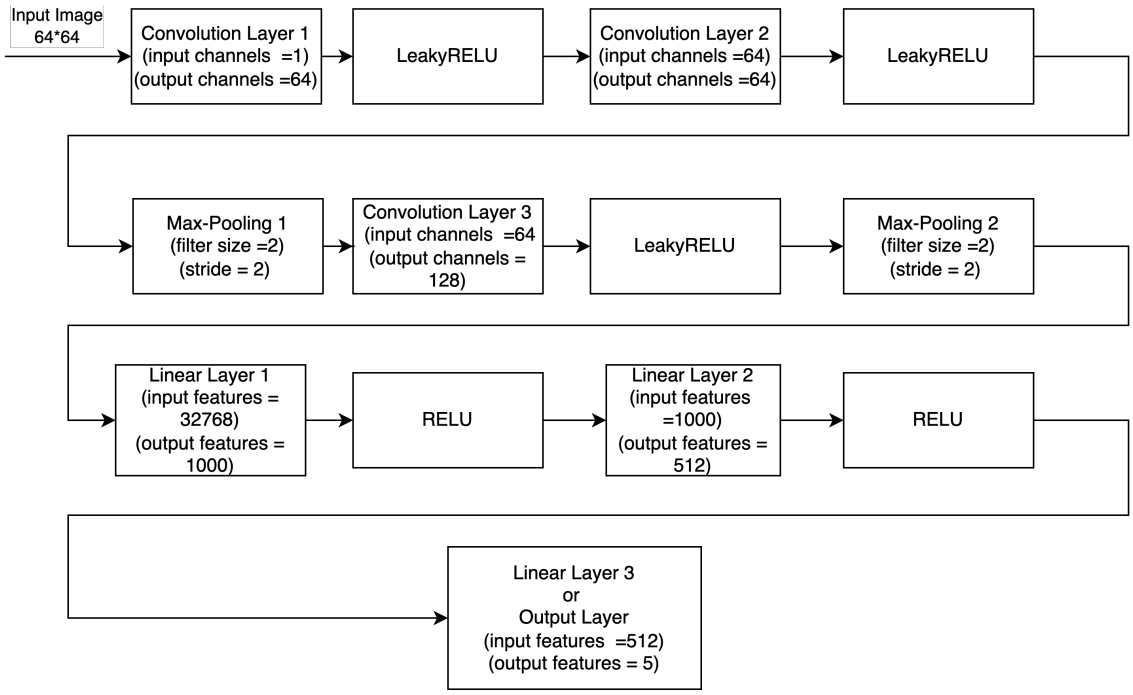

Figure 2: Diagram of CNN Architecture

3.1 Hyper-parameters

There were several hyper-parameters that were experimented with, and tuned to produce our final CNN architectures, namely:

- Dropout Rate: Dropout is a regularization technique for reducing overfitting in neural networks by preventing complex co-adaptations on training data [10]. It does this by turning off some nodes in the network during training at random. We used a dropout rate of 0.1 in this project

- Learning Rate: The learning rate is a hyperparameter that controls how much the model changes in response to the estimated error each time the model weights are updated [1]. Too small a learning rate may mean our model takes too long to train or gets stuck, while too high a learning rate produces an unstable training where the weight updates swing around wildly. We tried the PyTorch SGD optimizer [3] with a learning rate of 0.0001, but later settled on the PyTorch Adam optimizer [2] with the same learning rate because we achieved better accuracy results.

- Kernel Size: The kernel is simply the convolutional filter that is passed over our images during the convolution settled on 8, after which we were not enhancing the accuracy.

- Number of Epochs: We first ran our training process with 6 epochs which were chosen randomly, recorded the test accuracy , and continued to increment the number of epochs till we finally settled on 8, after which we were not enhancing the accuracy

3.2 Comparison With Architecture Variants

Our main network which we henceforth refer to as a "shallow network" consists of 3 convolutional layers and 2 fully connected layers. For the sake of comparison we create two variants of our base model, which should have:

- More convolutional layers in one network

- More max pooling layers in the second network.

We trained both new variant networks with the image data as before and saved the trained models for these networks. We discuss the accuracy achieved by these models in the evaluation section. In order to create the first new model variant model we first add 3 more convolutional layers to our "shallow network" to produce this second model for a total of 6 convolutional layers, the number of fully connected layers and max-pooling layer remain the same. We refer to this network as the "Deep-fewer-maxpooling network". The second network variant we created by adding 3 extra convolutional layers to the base "shallow network". We also add an additional 2 max-pooling layers here for a total of 4 max-pooling layers, the number of fully connected layers remains the same. We refer to this network as the "Deep-many-maxpooling" network.

4 Evaluation

This section presents the experimental metrics and the models' results. Firstly, it gives a brief description of the metrics used to evaluate the proposed models. Secondly, we compare the results generated using our test data on all the models. The three models we trained performed comparatively , with each other.

4.1 Experimental Metrics

The metrics used to evaluate the performances of our proposed models are briefly discussed below:

1. Accuracy: Accuracy is the number of correctly classified sample instances divided by the total number of sample instances. More precisely, it represents the number of true positives and true negatives divided by the number of true positives, true negatives, false positives, and false negatives [11]. It can be depicted as:

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

   where TP and TN are true positive and true negative respectively. While FP and FN are the false positive and false negative respectively.

2. Precision: For ideal good classifiers, precision should be 1. This usually happens when

the false alarm is zero. Mathematically:

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

3. Recall: Recall is otherwise known as sensitivity or the true positive rate and is formulated as follows [11]:

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

4. F1-score: F1–score accounts for both precision and recall. F1–score is the harmonic mean of precision and recall and is a better measure than accuracy [11]. It is defined as follows: F1-score $=2 \times \frac{Precision \times Recall}{Precision + Recall}$.

5. Finally, we also use a confusion matrix to evaluate our models. A confusion matrix is a visual evaluation technique used in machine learning. It depicts the actual class result associated with the prediction class.

4.2 Experimental Set-up and Results

In this subsection, we present the experimental setup and the simulation testing results of the three models. The models were trained using 1500 data samples and tested with 500 samples. For a fair comparison, all models were trained with Adam optimizer, categorical cross-entropy, ReLU, and LeakyReLU activation functions, 0.0001 learning rate, and 8 epochs. Note that we decided to stop the training at 8 epochs because the model starts to overfit after 8 epochs due to the limited training data set. Table 2 shows the Accuracy, Precision, Recall, and the F1 score of the three models

The base model that will initially investigate is tagged "shallow-network." It has an accuracy of 74%. After several architecture modifications, we came up with two better models. We call it "Deep-fewer-maxpooling". It has more convolutional layers, 6 layers in total, but without an additional pooling layer as the base model. This ensure that our limited data sets were efficiently utilized. However, this probably made the model to be more prone to overfitting. We further investigate a network with the same 6 convolutional layers, and tagged it as "Deep–many–maxpooling". Unlike the Deep–fewer–maxpooling architecture, we added max pooling to all the layers except the first two layers. The

intuition behind this is to ensure that the network extracts important features before down-sampling them. Overall, Deep–fewer–maxpooling and Deep–many–maxpooling achieved an accuracy of 77.9% and 79.6% respectively. Thus, Deep-fewer–maxpooling and Deep-many-maxpooling achieved 4.42% and 6.70% improvements over our base model "shallow-network." Deep–many–maxpooling further showed its superiority over shallow networks and Deep–fewer-maxpooling by achieving 0.80 precision, 0.79 shallow network achieved 0.75 precision, 0.75 recall, and 0.74 -score. While Deep-fewer–maxpooling achieved 0.79 precision, 0.78 recall, and 0.78 f1-score. In Overall, the Deep–many–maxpooling model performed better.

Table 2: Table showing the accuracy, precision, recall, and F-1 scores of the proposed models

Models	Accuracy (%)	Precision	Recall	F1-score
Deep-fewer-maxpooling	77.6	0.79	0.78	0.78
Shallow-network	74.6	0.75	0.75	0.74
Deep-many-maxpooling	79.6	0.80	0.80	0.79

We further investigate the class-wise performance of the model to ascertain how our models performed with the 5 classes in our data set. This is shown in 4. It turns out that all the models achieved the highest performance on cloth masks, from either precision, recall, or F1–score perspective. This could be attributed to the fact that the feature maps from the cloth will be relatively different from other mask classes derived from the same material, except for the no mask case. Also, it is notable that model_1 (Deep-fewer-maxpooling) has the lowest precision and recall for FFP2 masks. Model_2(Shallow-Network) and model_3 (Deep-many-maxpooling) achieved lower recalls for the FFP2 masks. On the other hand, the performance of the model in other classes is comparatively competitive.

4.3 Confusion Matrix of the Models

The correct predictions and the mismatches are presented with a confusion matrix. The diagonal values represent the correct predictions, while all the off–diagonal values are mismatched. For example, in Figure 4.3 shows the confusion matrix of the Deep–fewer–maxpooling model. The first cell indicates that 71 "incorrectly worn masks" were classified

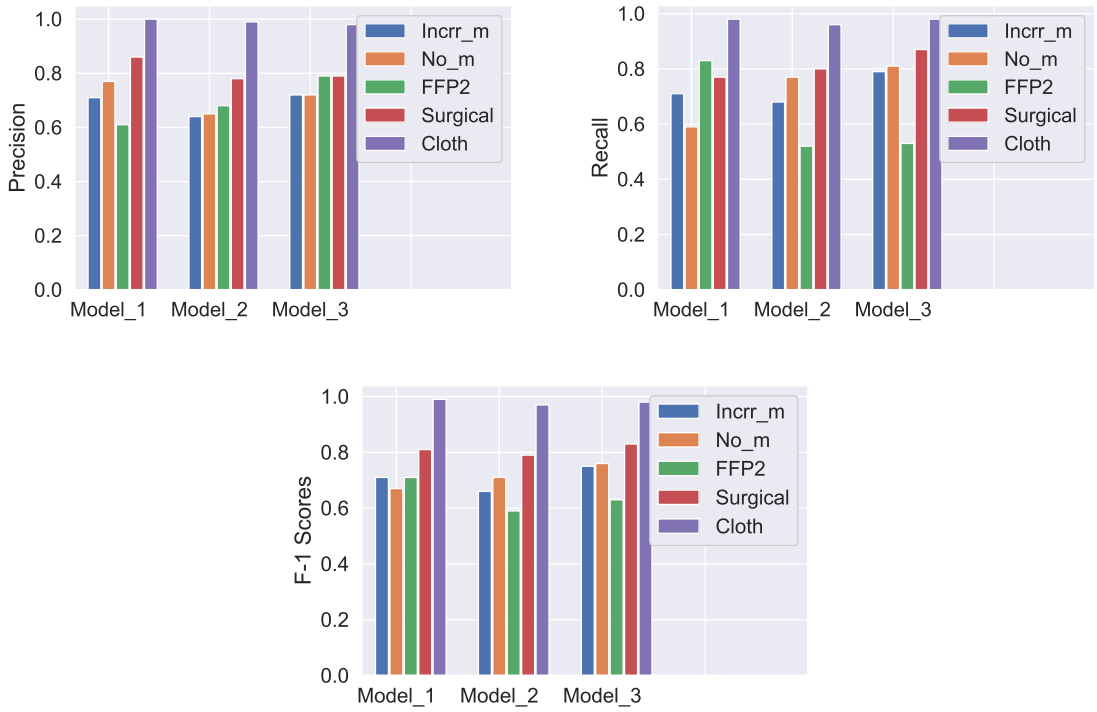

Figure 3: Models' performance-metric-measures on (a)Precision, (b) Recall, and (c) F–1 scores. Test samples is 500, Adam optimizer, ReLU and LeakyReLU activation functions, learning rate=0.0001, and epochs=8. Model_1 represents Deep–fewer–maxpooling, model_2 depicts Shallow-Network, and model_3 represents Deep–many-maxpooling.

correctly. Also, 95 cloth masks were also classified correctly. However, 19 "incorrectly worn masks" were misclassified as FFP2 masks. In the same vein, 96 cloth masks were correctly classified in the "shallow–network–confusion matrix". Lastly, in the confusion matrix of the Deep–many–maxpooling model, 98 cloth masks were correctly classified, while 10 "FFP2" images were incorrectly classified as surgical masks. It is interesting to note that all the models performed very well on cloth masks, which further explains why cloth masks have the highest values of precision, recall, and F1–score in Figure 4.

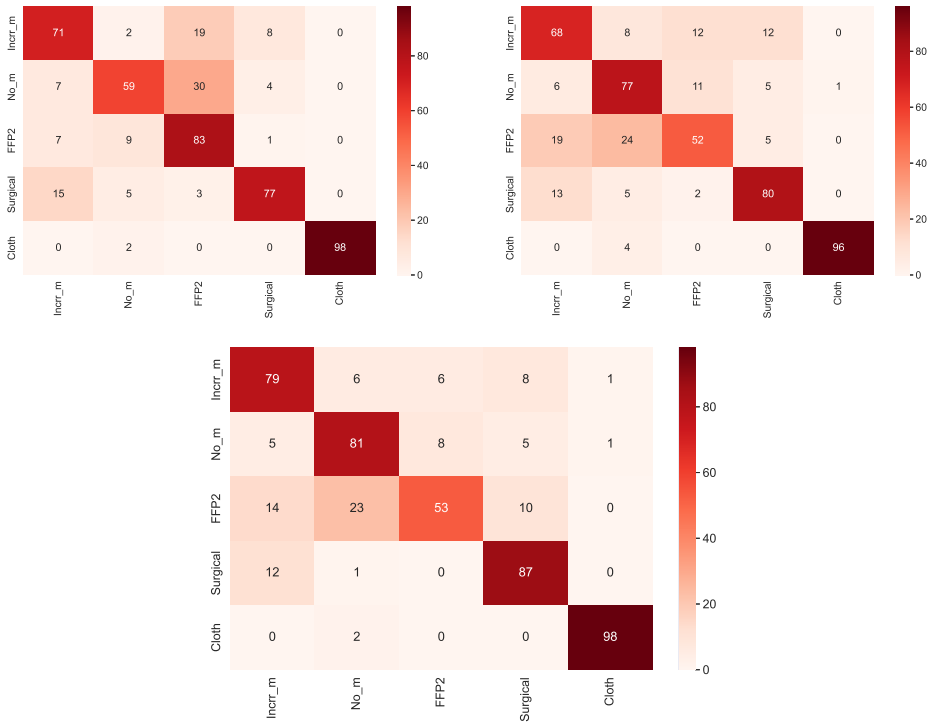

Figure 4: Confusion matrix of (a) Deep–fewer–maxpooling, (b) Shallow–Network, and (c) Deep–many–maxpooling. Test samples is 500, Adam optimizer, ReLU and LeakyReLU activation functions, learning rate=0.0001, and epochs=8. Incrr_m means incorrect mask wearing, No_m means no mask.

5 Conclusion

In this work, we constructed and trained three CNN models using features extracted from 75% of the data set, and tested the models with 25% of the data samples. We compared the test results of our proposed models, and the final model architecture tagged as "Deep–many–maxpooling" showed the best performance in terms of accuracy, precision, recall, and F1–score, achieving 79.6%, 0.80, 0.80, and 0.79 respectively during evaluation. The class–wise performances obtained with respect to precision, recall, and F1–score measures are arguably comparative to other competing models. It is evident that the deeper network took an advantage of learning more feature maps as dropout and max–pooling regularisation techniques were used to avoid overfitting in the network. Our results validate the quality performance of deep Convolution Neural Network (CNN) in image processing.

References

[1] J. Brownlee. Understand the impact of learning rate on neural network performance, Sep 2020.

[2] T. Contributors. Adam - pytorch 1.11.0 documentation. Accessed: June 07, 2022.

[3] T. Contributors. Sgd - pytorch 1.11.0 documentation. Accessed: June 07, 2022.

[4] C. Deb. Face Mask Detection system based on computer vision and deep learning using OpenCV and Tensorflow/Keras. https://github.com/chandrikadeb7/Face-Mask-Detection/tree/master/dataset, 2020. Accessed: June 06, 2022.

[5] P. N. et al. SSDM V2: A real time DNN-based face mask detection system using single shot multibox detector and MobileNetV2. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7775036/, 2021. Accessed: June 06, 2022. Copyright © 2020 Elsevier Ltd. All rights reserved.

[6] W. Intelligence. Face Mask Detection Dataset 20 Categories of Masks. https://www.kaggle.com/datasets/wobotintelligence/face-mask-detection-dataset, 2020. Accessed: June 06, 2022. CC0: Public Domain.

[7] A. JANGRA. Face Mask Detection 12K Images Dataset, Jessica Li. https://www.kaggle.com/datasets/ashishjangra27/face-mask-12k-images-dataset, 2020. Accessed: June 06, 2022. CC0: Public Domain.

[8] V. Kumar. Face Mask Detection building a face mask classifier. https://www.kaggle.com/datasets/vijaykumar1799/face-mask-detection, 2021. Accessed: June 06, 2022. CC0: Public Domain.

[9] S. G. e. a. MAFA dataset. MAFA dataset Face Mask Detection. https://drive.google.com/drive/folders/1nbtM1n0–iZ3VVbNGhocxbnBGhMau_OG, 2021. Accessed: June 06, 2022.

[10] Pavansanagapati. What is dropout regularization?, Jul 2019.

[11] J. Weaver, B. Moore, A. Reith, J. McKee, and D. Lunga. A comparison of machine learning techniques to extract human settlements from high resolution imagery. In *IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium*, pages 6412–6415. IEEE, 2018.