CONCORDIA UNIVERSITY

COMP 6721 Applied AI Project by OB_2

Professor: *Dr. René Witte*

Student name: *Maryam Valipour (40224474) - Data Specialist,*
Kary Sutariya (40193909) - Training Specialist,
Kanika Aggarwal (40195981) - Evaluation Specialist

Project Repository of AI

1. DATA SET

Data for this project has been collected mainly from google images and a Kaggle dataset [1] made by author RAHUL MANGALAMPALLI, which are publicly available. IMAGE DOWNLOADER software is used to collect the images from google. The images of class 'Mask worn incorrectly' are mostly gathered from Kaggle [1]. Most of images are taken from google [2] or Kaggle dataset. It is also worth mentioning that efforts have been made to collect data from social media websites, but not more than ten appropriate images were found using these platforms [3].

Moreover, the size of this dataset is 151 MB, which in total contains 2125 images for all the categories. These images are divided into five classes, and the samples of each class are shown in Fig. 1 below.

Figure 1: Samples of each class of dataset

Each of the image comes in different sizes, resolutions, and formats (JPEG and PNG). The size and average resolution for each class are shown in Table 1. After collecting the required data for the project, data is preprocessed in order to be used in the neural network model for training.

Class	No. of images	Size	Resolution
Cloth mask	411	49.9 MB	461x422
Mask worn incorrectly	400	23.1 MB	395x425
N95 mask	423	24.6 MB	328x290
No face mask	461	23.9 MB	240x227
Surgical mask	430	29.8 MB	436x369

Table 1: Common Attributes of the Dataset

1.1. Data Preprocessing.

Firstly, the images and their respective categories are loaded using the PIL library into the program. After that, they are split into train, validation, and test categories using the Scikit-Learn library. Before preprocessing, the number of images in the train, validation, and test dataset are 1431, 159, and 535, respectively.

In this next step, the torchvision.transforms.compose is used to compose several transforms together, which is used on the existing images. All transformation functions accept PIL images, and they can be passed to the transform functions. In this part, some images were removed from the datasets because they only had one channel and could not be resized to (3,224,224). The different transformations [4] used on the images are described below.

1. **Resize**: Since neural networks expect all the image data to be the same size, the images from all datasets should be resized into a particular and shared size. For the purposes of this project, we resize all the images into the size (3, 224, 224), where 3 is the number of channels, and 224 is the width and height of the images.

2. **Randomrotation**: This function rotates images 10 degrees from their angle.

3. **Randomhorizontalflip**: This function flips the images horizontally with a 50 percent probability. The reason for using this type of transformation is that if all the images are from the same angle, the network will be biased to this particular angle, and the network would not generalize well to the unseen data. So, this transformation will cause the network to become less biased, and the training process will be improved.

4. **ToTensor**: This will transform the images into tensors, which has to be done in order for the network to work properly.

5. **Normalization**: This helps to improve the performance of the model. So for this, the data should be normalized using the mean and standard deviation (std) of the images. For the purposes of this project, the mean and standard deviation of the ImageNet dataset is used, which works well enough.

For the training dataset, all the above-mentioned transformations are used. However, for the validation and test dataset, only resize, ToTensor, and Normalization are used. The reason behind this is that for training, overfitting should be avoided, so other kinds of transformations are used to reduce the bias from the dataset, and for the validation and test dataset, the performance of the model on the real dataset should be measured.

After applying these transformations to the images, the number of the images for train, validation, and test datasets were changed to 1368, 147, and 506 images, respectively. For the labels, a number should be assigned to each class of the dataset, so that the network would understand the output classes. This is done using the Labelencoder from the Scikit-Learn library. The categories and their corresponging labels are listed in Table 2. below.

In the next step, the created datasets will be passed to the DataLoader module in the Pytorch library.

Classification	Label
Cloth mask	0
Mask worn incorrectly	1
N95 mask	2
No face mask	3
Surgical mask	4

Table 2: Data Preparation

Then, the datasets are shuffled and divided into batches of 64, so that each batch would contain 64 images. A sample of 64 batches of images are shown in Fig. 2.

Figure 2: Sample of the Preprocessed dataset

In the next section, the architecture of the models are presented.

2. CNN ARCHITECTURE

In this study, approximately 75% of the dataset contributes to the training set and validation set, and the rest to the testing set. The input images are pre-processed and augmented using the steps described in the last section. The model architecture proposed is composed of two phases, data pre-processing and CNN Model training, which is shown in Fig. 3.

The Models are designed in python using Pytorch library [5]. Three different CNN based models are proposed to classify the images into the 5 mentioned categories. Three different model architectures are proposed in this project, and they all share a few hyperparamters, including learning rate, number of epochs for training, batch size, optimizer, and Loss function, which their values are presented in Table 3.

Below, some functions that are used in architecture of the models are described:

1. **Convolution Layers**: Each convolutional layer applies a convolution operation to the input and passes the result to the next layer.

2. **Activation Functions**: They are placed at the end or among the networks, which ultimately decide whether to fire a neuron or not. The choice of activation function at hidden layers as well as at the output layer is significant as it controls the quality of training the models. The LeakyReLu activation function is based on ReLU, and it has a small slope for negative values instead of a flat slope [6].

Therefore, LeakyReLU is used in hidden layers as it can help avoid the vanishing gradient problem and improve computational performance. Additionally, at the end of the network, a SoftMax activation function is used to calculate the probabilities distribution of each class to finally be able to predict the corresponding class of each image.

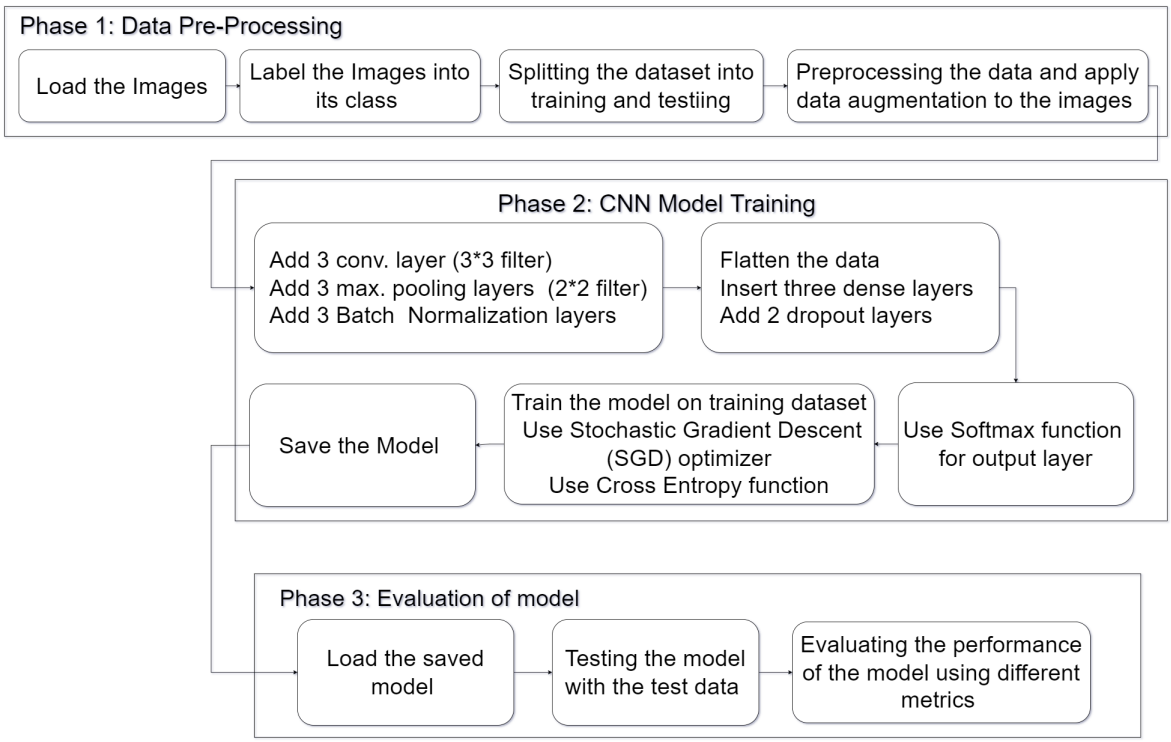


Figure 3: The proposed architecture

3. **Pooling Layers**: Pooling layers decrease the size of the feature map [7]. Thus, the number of trainable parameters is reduced, resulting in rapid calculations without losing the essential features. It also helps to reduce the risk of overfitting in large networks.

Parameter	**Values**
Learning Rate	0.001
Epochs	100
Batch Size	64
Optimizer	SGD
Loss Function	Cross Entropy

Table 3: Hyper Parameters used in Training

4. **Dropout Layers**: The Dropout layer is added to randomly skip some connections to force the network to learn from other parameters. It also helps to avoid overfitting [8].

Layer(type)	Output Shape	Structure
Conv2D	16 * 224 * 224	Filters = 16, Filter Size = 3 * 3, Stride = 1
MaxPooling	16 * 112 * 112	Filters = 16, Filter Size = 2 * 2 , Stride = 2
Conv2D	32 * 112 * 112	Filters = 32 , Filter Size = 3 * 3 , Stride = 1
MaxPooling	32 * 56 * 56	Filters = 32, Filter Size = 2 * 2 , Stride = 2
Conv2D	64 * 56 * 56	Filters = 64 , Filter Size = 3 * 3, Stride = 1
MaxPooling	64 * 28 * 28	Filters = 64, Filter Size = 2 * 2
FC	50176 * 128	-
FC	128 * 64	-
FC	64 * 32	-
FC	32 * 5	-
Classification Layer	5	Softmax

Table 4: Model 1 Summary

Layer(type)	Output Shape	Structure
Conv2D	16 * 224 * 224	Filters = 16, Filter Size = 3 * 3, Stride = 1
MaxPooling	16 * 112 * 112	Filters = 16, Filter Size = 2 * 2 , Stride = 2
Conv2D	32 * 112 * 112	Filters = 32 , Filter Size = 3 * 3 , Stride = 1
MaxPooling	32 * 56 * 56	Filters = 32, Filter Size = 2 * 2 , Stride = 2
FC	100352 * 128	-
FC	128 * 64	-
FC	64 * 32	-
FC	32 *5	-
Classification Layer	5	Softmax

Table 5: Model 2 Summary

5. **Flatten Layer**: The entire filter maps are then flattened to provide features to the classifier.

2.1. Models.

In the first model, 3 Conv blocks and one FC block are used in the model design. Each Conv block shares the same pattern. They are composed of one Conv2D layer, with 3*3 filter and 'same' padding, one LeakyRelu activation function layer, one batch normalization layer, and one Max-pooling layer with filter size 2 x 2 and stride 2. Additionally, the FC block contains four fully connected layers, Relu layers, and two Dropout layers with a probability of 0.5. The final layer of the neural network has only five outputs, which helps to classify the images into 5 classes with their corresponding values. The summary of the first model layers and their output size is presented in Table 4.

The second model includes 2 Conv blocks and one FC block (The blocks used are the same as in the first model). The layers and the respective output sizes of the second model are also presented in Table 5.

In the third model, two Conv block and one FC block are used. The Conv block used is the same as previous models with one difference, which is that max-pooling

Layer(type)	Output Shape	Structure
Conv2D	16 * 224 * 224	Filters = 16, Filter Size = 3 * 3, Stride = 1
Conv2D	32 * 224 * 224	Filters = 32, Filter Size = 3 * 3, Stride = 1
FC	1605632 * 128	-
FC	128 * 64	-
FC	64 * 32	-
FC	32 * 5	-
Classification Layer	5	Softmax

Table 6: Model 3 Summary

layers are omitted from the Conv block. The FC block used is the same as previous models. Similarly, the layers and the respective output sizes of the third model are presented in Table 6.

3. EVALUATION

In this multiclass classification problem, images are classified into five different categories i.e., Cloth mask, No face mask, Surgical mask, N95 mask and mask worn incorrectly. The models are trained using previously described hyper parameters. They are trained and validated for 100 epochs on the train and validation datasets, respectively. The loss and accuracy of both training and validation phases for the three proposed models are summarized in Tables 7, 8, and 9.

Epochs	Train Loss	Train Accuracy	Val Loss	Val Accuracy
10	1.3724	42.94	1.3788	52.27
20	1.1823	54.06	1.1765	57.59
30	1.0360	63.3	1.1040	57.33
40	0.9037	67.68	1.0456	59.75
50	0.7896	71.78	0.9499	65.12
60	0.6490	77.91	0.8001	65.30
70	0.5565	82.03	0.8057	69.29
80	0.4453	86.61	0.8691	71.96
90	0.3688	91.25	0.8144	71.44
100	0.2835	93.04	0.8221	80.101

Table 7: Loss and Accuracy for Train and Validation of Model 1

Epochs	Train Loss	Train Accuracy	Val Loss	Val Accuracy
10	1.3834	40.93	1.4312	42.59
20	1.1047	58.96	1.2771	59.90
30	0.8690	70.64	1.0783	61.69
40	0.7039	76.63	0.9149	66.12
50	0.5441	84.64	0.9528	72.45
60	0.4140	89.99	0.8904	72.56
70	0.3181	91.62	0.7766	69.43
80	0.2549	95.08	0.8359	67.61
90	0.2142	95.65	0.7594	67.35
100	0.1818	95.54	0.7656	71.93

Table 8: Loss and Accuracy for Train and Validation of Model 2

Epochs	Train Loss	Train Accuracy	Val Loss	Val Accuracy
10	0.9316	68.73	1.2111	57.78
20	0.5919	84.02	1.1704	56.06
30	0.3742	90.32	1.1210	62.01
40	0.2577	93.61	1.1452	63.90
50	0.1812	95.31	1.0878	62.04
60	0.1446	96.47	1.0546	59.02
70	0.1172	97.66	1.1973	61.75
80	0.0958	97.68	1.1846	60.22
90	0.0916	97.75	1.1008	60.71
100	0.0943	96.97	1.1543	64.28

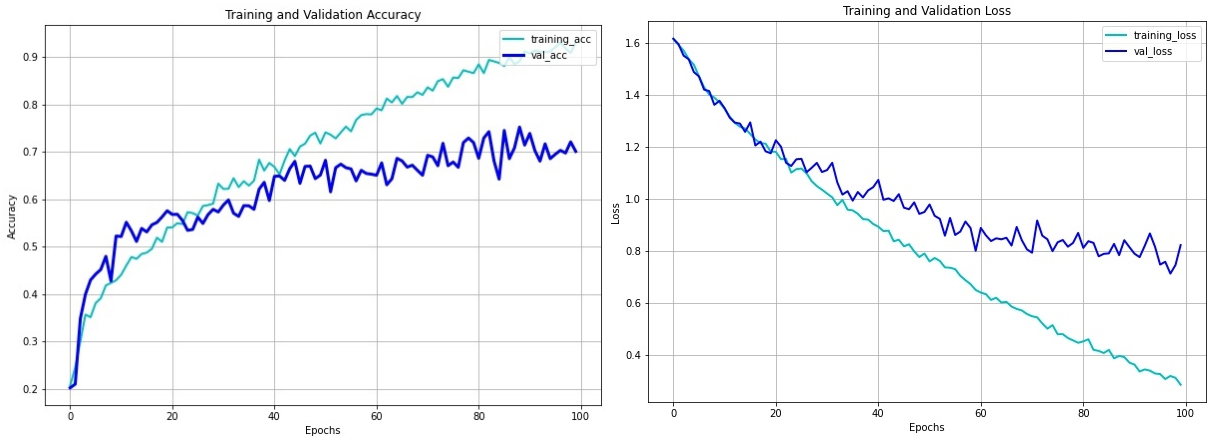
Table 9: Loss and Accuracy for Train and Validation of Model 3

Additionally, the loss and accuracy of the three proposed models in the training and validation process are also shown in Figures 4, 5, and 6.

For testing, 500 different images have been selected which are distributed amongst these five classes. The accuracy of the first, second, and third model for the test dataset is 72.33, 70.75, and 65.91, respectively.

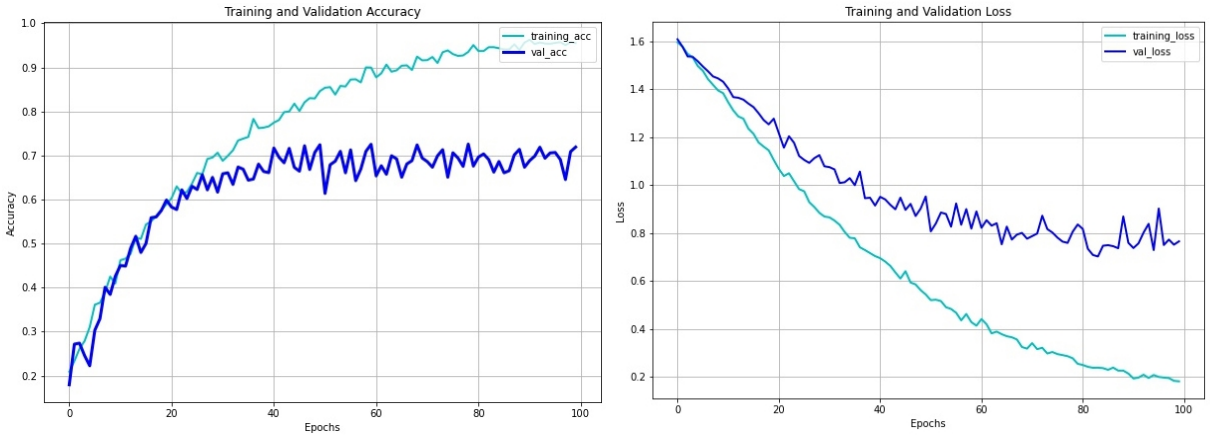
It is known that by adding more convolutional layers and therefore adding more parameters to the network, the performance of the model should improve. However, this has a limit, and from a certain point, adding more convolutional layers will only result in overfitting, and the performance of the model on unseen data would decrease.

The difference between the first and the second model is in the number of convolutional layers used in their architectures, which is three and two convolutional layers, respectively. The first model is doing a little better on the test dataset than the second model; however, the difference is not much noticeable. This means that from this point on, by adding more convolutional layers, the performance might not improve and could result in overfitting.



(a) Training and Validation accuracy (b) Training and Validation loss

Figure 4: Accuracy and Loss test results during Model 1 training



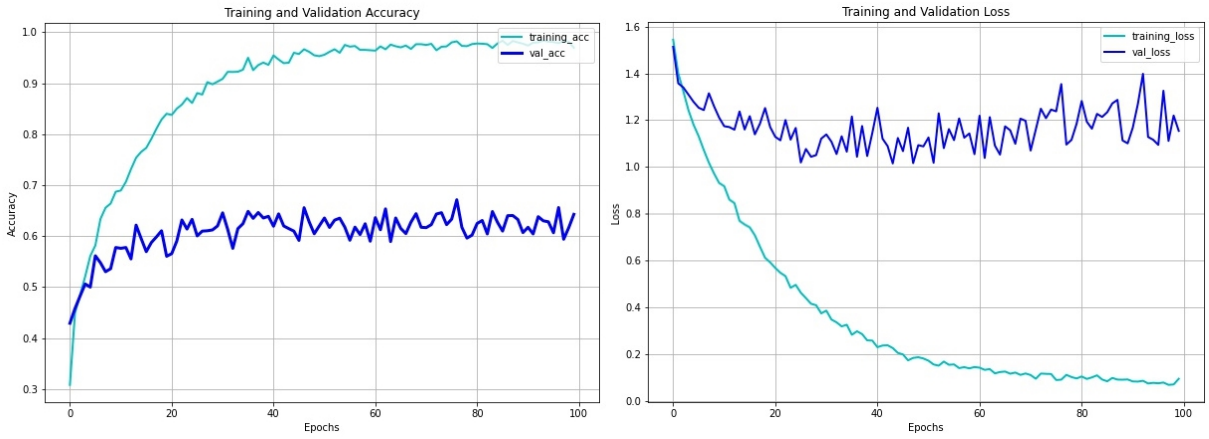
(a) Training and Validation accuracy (b) Training and Validation loss

Figure 5: Accuracy and Loss test results during Model 2 training

As can be seen in the above tables and plots, in the first few epochs, the third model is performing much better than the first and second model. However, at the end, it is performing worse than them. It can be seen that the model is quickly overfitting to the training images and the training accuracy is quickly rising. The reason is that in the third model, there is no max-pooling layer used in the model architecture. Therefore, the output size of the convolutional neural network is very large (1605632) as

described in Table 6. So, it can be seen that the model is overfitting. As was mentioned before, the use of max-pooling layers helps avoid overfitting by reducing the number of parameters.

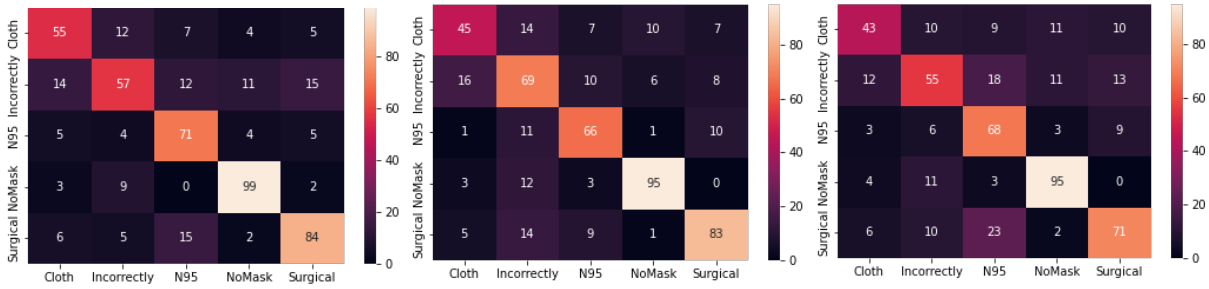
As can be seen, the third model has the lowest accuracy in the test dataset. Therefore, in this project, omitting the max-pooling layers in the third model had greatly impacted the model performance and the test accuracy.



(a) Training and Validation accuracy (b) Training and Validation loss

Figure 6: Accuracy and Loss test results during Model 3 training

The three confusion matrix have been generated in Fig 7. for the three proposed models. The confusion matrix makes easy to show whether the model is performing correctly i.e., which classes the model is predicting correctly and which the model is predicting incorrectly.
The final results of the proposed models are presented in the classification report as shown in Fig 8. Various metrics have been used to evaluate the models i.e., accuracy, precision, recall, and f1 score [9].



(a) Model 1 (b) Model 2 (c) Model 3

Figure 7: Confusion Matrix

$$Accuracy = \frac{Tp + Tn}{Tp + Tn + Fp + Fn} \tag{1}$$

$$Precision = \frac{Tp}{Tp + Fp} \tag{2}$$

$$Recall = \frac{Tp}{Tp + Fn} \tag{3}$$

$$F1 = (\frac{2 * Precision * Recall}{Precision + Recall}) \tag{4}$$

In the above equations, Tp represents: True positive, Tn: True negative, Fp: False positive, and Fn: False negative.

In True positive, model accurately predicted the positive class, whereas in false positive the model incorrectly predicted the positive class. In True negative, model accurately predicted the negative class, whereas in false negatives the model incorrectly predicted the negative class.

	precision	recall	f1-score	support
0	0.66	0.66	0.66	83
1	0.66	0.52	0.58	109
2	0.68	0.80	0.73	89
3	0.82	0.88	0.85	113
4	0.76	0.75	0.75	112
accuracy			0.72	506

(a) Model 1

	precision	recall	f1-score	support
0	0.64	0.54	0.59	83
1	0.57	0.63	0.60	109
2	0.69	0.74	0.72	89
3	0.84	0.84	0.84	113
4	0.77	0.74	0.75	112
accuracy			0.71	506

(b) Model 2

	precision	recall	f1-score	support
0	0.63	0.52	0.57	83
1	0.60	0.50	0.55	109
2	0.56	0.76	0.65	89
3	0.78	0.84	0.81	113
4	0.69	0.63	0.66	112
accuracy			0.66	506

(c) Model 3

Figure 8: Classification Report

In conclusion, the main model has better performance in comparison to the two other models.

3.1. Improvements.

Since the task of classifying these images into the five categories is complex and complicated, a large amount of data is needed to train the model. Then, the model can perform better and yield better accuracy.

Moreover, Bias is another issue. For instance, in our dataset, most of the images are of young males, and is a little biased towards gender, age, and race. A balanced dataset will help to improve the model for real-time applications.

As the number of images are low, data augmentation and several oversampling techniques such as SMOTE (Synthetic Minority Oversampling Technique) and Fair SMOTE should be done to increase the number of training images. Using these techniques can reduce bias as well.

Additionally, different and more complex models such as ResNets, R-CNNs, Fast R-CNNs, Faster R-CNN, YOLO can be used to further improve the overall performance.

REFERENCES

1. https://www.kaggle.com/datasets/rahulmangalampalli/mafa-data
2. https://www.google.com/search?q=worn+mask+incorrectly
3. https://www.facebook.com/search/photos/?q=mask%20worn%20incorrectly
4. https://towardsdatascience.com/improves-cnn-performance-by-applying-data-transformation
5. https://towardsdatascience.com/convolutional-neural-network-for-image-classification-with-implementation-on-python-using-pytorch
6. https://towardsdatascience.com/activation-functions-neural-networks
7. https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer
8. https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks
9. https://machinelearningmastery.com/precision-recall-and-f-measure-for-imbalanced-classification