Applied Artificial Intelligence

COMP 6721

AI Face Mask Detector

Jay Patel 40197003 - Data Specialist
Krupali Bhatt 40184668 - Training Specialist
Juhi Birju Patel 40190446 - Evaluation Specialist

Index

1. Dataset

The dataset used in this project is a combination of images taken from Kaggle[1] and other references as cited below[2].

The Kaggle dataset contained 4 folders: no mask, cloth mask, N95, and surgical mask. For the fourth category which was N95 with a valve, we were able to collect 169 images from a couple of kaggle[3][4] sources, and manually searching google images[5].

We performed image augmentation on this dataset by writing a python script, which included cropping, flipping, resizing images, changing brightness, saturation and contrast in order to improve the learning process for our model. Additionally, We performed image augmentation on all other images which included resizing and cropping the images towards the centre so that maximum information can be captured.

The dataset is balanced with 5280 RGB images in 5 folders named

- Cloth_mask
- ffp2 mask
- N95_with_valve
- surgical_mask
- and no_mask.

Given below are the statistics of our dataset:

Class	Number of Images
Cloth Mask	1088
Surgical Mask	1029
N95 Mask	1023
N95 with a valve	1069
No mask	1071
Total	5280

The following pie chart shows the distribution of our dataset and we can observe that it is completely balanced.

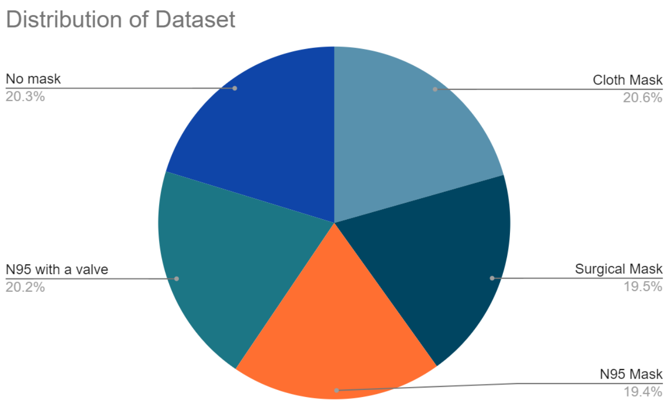


Fig 1. Distribution of face mask dataset

Image Augmentation:

Original Image

Cropped Image

Flipped Image

Rotated Image

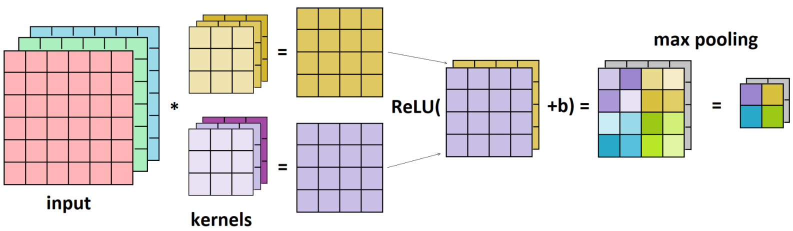
Saturated Image

Contrast Image

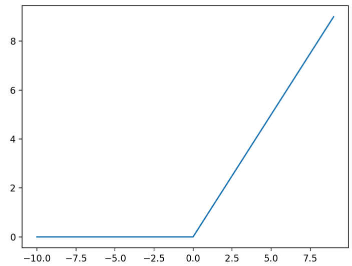
2. CNN Architecture

2.1 Model Design

- Input of [3*100*100] is feed into the first convolution layer, where 3 is the number of channels i.e. RGB in our case and 100 is the height and the width is 100.
- There are 4 convolution layers with a kernel size of (3,3), stride of (1,1) , and a padding of (1,1).
- The convolution layer is followed by a 2 Dimensional Batch Normalisation.
- Followed by the batch normalisation, is the ReLU Activation function.The rectified linear activation function (RELU) is a piecewise linear function that, if the input is positive say x, the output will be x. Otherwise, it outputs zero.
- There are two pooling layers , each after every 2 convolution layers. The pooling used is max pooling with a kernel of (2,2) and a stride of 2.
- We have used a drop out of 0.02 in order to prevent the model from overfitting.
- Finally, the output from the final pooling layer is flattened and fed as an input to the fully connected layer. Neurons in a fully connected layer are connected to all the activations in the previous layer. Therefore, it helps to classify the input into predicted classes.



Convolutional Layer Architecture [1]



ReLU Activation Function [2]

2.2 Data Preparation for the Model

- The balanced dataset is fetched , shuffled and saved into .npy file for faster processing of the images.
- Images in the dataset are transformed to tensor objects, they are normalised and resize to 100 X 100.
- We have used 70% of the dataset for training and 30% for the testing part.
- An equal proportion of all the five classes are taken for the purpose of testing to generate the training and testing confusion matrix.

2.3 Configuring the optimizer

- We have used the Adam optimizer, instead of the classical stochastic gradient descent procedure and the learning rate is set to 0.001, to update network weights iteratively.

---

1 https://www.baeldung.com/cs/deep-cnn-design

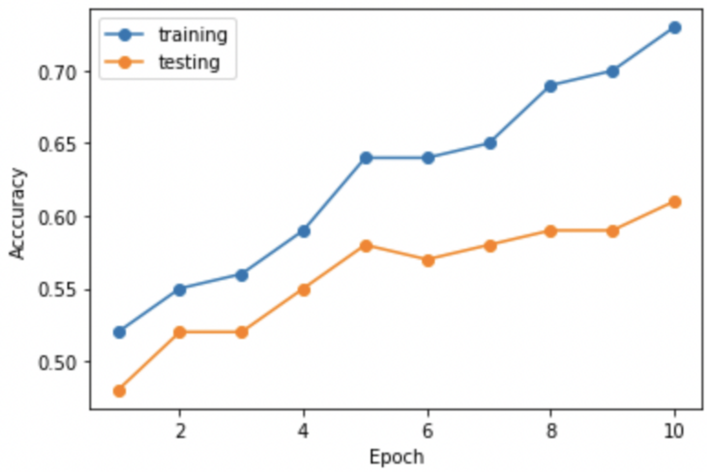
2 https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/

2.4 Model Training

- We trained the model for various epochs. The graph plotted below shows the epoch Vs training and testing accuracy. Based on the graph we decided to move forward with the number of epochs as 13, in order to get the maximum accuracy.
- After each epoch, We print the training accuracy and the cross entropy training loss and we see that the accuracy increases and the loss decreases.
- The optimizer.zero_grad() function sets the gradients to zero.
- The backward() function is called on the loss to calculate back propagation.
- The optimizer.step() iterates over all the parameters and updates them using their internally stored grad values.

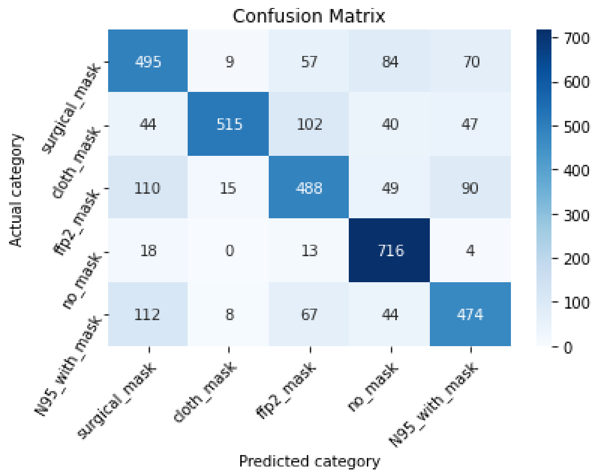
```
Training the model:
Epoch [1/10], Loss: 1.3628, Accuracy: 47.83%
Epoch [2/10], Loss: 1.1961, Accuracy: 39.13%
Epoch [3/10], Loss: 1.0674, Accuracy: 56.52%
Epoch [4/10], Loss: 0.9232, Accuracy: 60.87%
Epoch [5/10], Loss: 1.0227, Accuracy: 56.52%
Epoch [6/10], Loss: 1.0085, Accuracy: 56.52%
Epoch [7/10], Loss: 0.9524, Accuracy: 69.57%
Epoch [8/10], Loss: 0.8849, Accuracy: 65.22%
Epoch [9/10], Loss: 0.7187, Accuracy: 65.22%
Epoch [10/10], Loss: 0.7445, Accuracy: 65.22%
Saving the trained model:
```

Epochs and their Accuracy



Graph of Accuracy Vs Epochs

```
Training Confusion Matrix:
```



Training Confusion Matrix

```
Training Classification Report:
              precision    recall  f1-score   support

           0       0.69      0.64      0.66       779
           1       0.69      0.94      0.80       547
           2       0.65      0.67      0.66       727
           3       0.95      0.77      0.85       933
           4       0.67      0.69      0.68       685

    accuracy                           0.73      3671
   macro avg       0.73      0.74      0.73      3671
weighted avg       0.75      0.73      0.73      3671
```

Training Classification Report

3. Evaluation

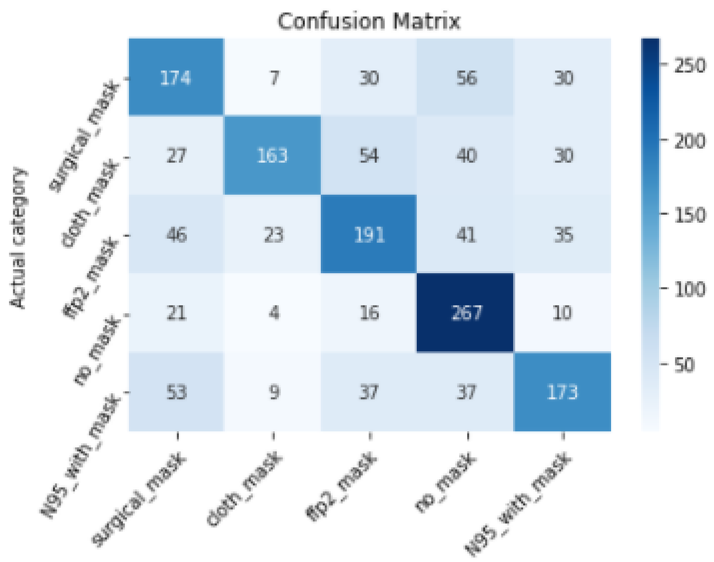
- We are training the model under 13 epochs and a batch size of 32.
- After each epoch the accuracy and cross entropy training loss are printed and we observe that with increasing epochs the accuracy increases and the loss decreases.
- The average accuracy, precision, recall and f1-score for training and testing are calculated to get better analytics for our model
- After successful training of our model, We save the model in a pickle file to reuse it.

Testing Classification Report:

	precision	recall	f1-score	support
0	0.59	0.54	0.56	321
1	0.52	0.79	0.63	206
2	0.57	0.58	0.58	328
3	0.84	0.61	0.70	441
4	0.56	0.62	0.59	278
accuracy			0.61	1574
macro avg	0.61	0.63	0.61	1574
weighted avg	0.64	0.61	0.62	1574

Testing Classification Report

Testing Confusion Matrix:



Testing Confusion Matrix

Classification Results

Predicting the labels from sample images:


cloth_mask


N95_with_valve


no_mask


N95_with_valve


surgical_mask

From the above results and observations, We plan the following for our Build 2.

- Add convolution layers for better feature extraction.
- Add “N95 With Valve” Dataset in order to generate a more balanced dataset.
- Improvise the testing accuracy of our model
- Implement K-fold cross Validation.
- Generate various observation graphs of hyperparameter vs accuracy, for performance tuning and select the best parameters, thereby improving the accuracy.

4. References

Reference Links to the datasets used:

[1] Face Masks Dataset-
https://www.kaggle.com/dataset/a71dfe0333dcabd1827ca3d6dcfd62d43785f83d3c38321b4113339a14f780e9

[2] Other References for Mask Dataset

- https://www.kaggle.com/ashishjangra27/face-mask-12k-images-dataset
- https://www.kaggle.com/andrewmvd/face-mask-detection
- https://www.kaggle.com/prasoonkottarathil/face-mask-lite-dataset
- https://www.kaggle.com/dhruvmak/face-mask-detection
- https://www.kaggle.com/sumansid/facemask-dataset
- https://www.kaggle.com/andrewmvd/face-mask-detection?select=images
- https://www.kaggle.com/omkargurav/face-mask-dataset
- https://www.kaggle.com/prithwirajmitra/covid-face-mask-detection-dataset

[3] Reference for N95 With Valve Images- Kaggle
https://www.kaggle.com/datasets/andrewmvd/face-mask-detection

[4] Reference for N95 With Valve Images - Kaggle
https://www.kaggle.com/datasets/wobotintelligence/face-mask-detection-dataset

[5] Manually Searching Google Images

- https://abc7news.com/n-95-mask-with-valve-kn95-n95-for-smoke/6378125/
- https://hartfordhealthcare.org/about-us/news-press/news-detail?articleid=26322&publicId=395
- https://theconversation.com/high-filtration-masks-only-work-when-they-fit-so-we-created-a-new-way-to-test-if-they-do-155987
- https://www.jracenstein.com/learn/expert-advice/why-you-shouldn%E2%80%99t-wear-a-mask-with-a-valve-for-covid-19/a136

- https://evaculife.com.au/product/adult-kn95-n95-p2-coronavirus-mask-white/
- https://www.sfchronicle.com/bayarea/article/Coronavirus-Bay-Area-officials-warn-some-N95-15208241.php
- https://sf.curbed.com/2018/11/9/18079866/fire-smoke-face-mask-find-oakland-san-francisco
- https://www.popsci.com/story/health/face-mask-valve-covid/
- And many more…

References Links for model design and evaluation

1. https://en.wikipedia.org/wiki/Convolutional_neural_network
2. https://machinelearningmastery.com/k-fold-cross-validation/
3. https://machinelearningmastery.com/confusion-matrix-machine-learning
4. https://www.analyticsvidhya.com/blog/2021/03/image-augmentation-techniques-for-training-deep-learning-models/
5. https://towardsdatascience.com/batch-normalisation-in-deep-neural-network-ce65dd9e8db