
# COMP 6721

## Applied Artificial Intelligence

Project Report

Submitted To: Prof. Dr. René Witte

Team Id: NS_06

Vishanth Surresh (40181942) - Training Specialist, Evaluation Specialist

Tejaswini Devireddy (40186127) - Data Specialist, Evaluation Specialist

**Index**

## 1. DataSet:

Dataset in the field of AI is a collection of data units. Although it is a collection of data units, system will consider it as a single unit for the sake of analytics and predictions. Only way for the AI to learn is from the data set. Without dataset or proper dataset, it is impossible to get proper results. AI first needs to be trained with a proper dataset and can test with another dataset. Training and testing are two main phases involved in any AI projects. Does not matter how good the implementation is, only a proper appropriate dataset can produce accurate results with high probability of correctness. During the process of this project, we first spent time on collecting appropriate data set according to our requirements and when once we are good with data set, we started writing algorithms for the AI to learn from the dataset [Train Phase] and also tested with few images [Test phase].

For our project, below are details of datasets:

We used open-source datasets for our project. The dataset we used in this project is a consolidation of multiple datasets taken from Kaggle [1] and Google Images. Dataset is an unbalanced one which consists of 1588 images in total in 5 different folders named as cloth, N95, N95_with_valve, no (without mask) and surgical.

| Mask Type | Number of Images |
|---|---|
| Cloth Mask | 400 |
| N95 Mask | 400 |
| Surgical Mask | 400 |
| No Mask | 400 |
| N95_with_valve | 388 |
| Total | 1588 |

## 2. CNN Architecture:

## 2.1 CNN Implementation:

A convolution neural network consists of input layer, hidden layer and output layer. Generally in feed forward network middle layers are referenced as hidden layers as their inputs and outputs are masked by activation function and final convolution.

In this project we used 4 convolution layers.

Input – [3 * 32 * 32]

The input is feed forwarded into the first convolution layer, where 3 is the number of channels i.e. RGB in our case and 32, 32 are the height and width of the image.

Each of these 4 convolution layers has the kernel size of (3,3), stride of (1,1) and padding of (1,1). The convolution layer is followed by a Batch Norm Layer which is used to distribute data uniformly across a mean that is best to the network. The batch normalization is followed by the Activation function. Here we used ReLU as the activation function. There are two pooling layers, each after every 2 convolution layers. Max pooling with a filter of (2,2) and a stride of 2 was used. Padding help to retain more information at the border of the image. The Dropout function is used to avoid overfitting of the model. Finally output from final pooling layer is flattened and fed as an input to the fully connect layer. Neurons in fully connected layer was connected to activations in previous layer which helps classify data into multiple classes.

## 2.2 Data Preparation

Pickle file is used to store the dataset created. Transformations is achieved using the torchvision.transforms library in pytorch. Followed by transformations resize of (32, 32), normalization and totensor functions is applied on the data. K Fold is used for the training and testing split. Dataset is divided into k folds using K fold cross – validator which is used to train the model with different possibilities of data.

## 2.3 Data Loaders

Data loader function is used for automatic batching. It is used for training and validation. Neural Network model is trained with batch size of 32. Shuffle attribute is set true which trains the model better as it receives data which are not in a repetitive trend.

## 2.4 Optimizer Configuration

Optimizer is a function that modifies the attributes of the neural network, such as weights and learning rate. This helps in reducing the overall loss and improve the accuracy. Adam optimizer is used and learning rate is set to 0.001

## 2.5 Model Training

Followed by optimizer configuration we train the data. The number of epochs is set to 10 and for each epoch the data is loaded using Dataloader. The model is trained using face_mask_detector model as mentioned in 2.1. Entropy loss is used to calculate the loss value. Optimizer.zero_grad() function is used to set gradients to zero. Back propagation loss is calculated using backward() function. A screenshot is attached below which gives an idea about how CNN model is trained.

```
report = pd.DataFrame(fold_outputs, columns=['accuracy', 'precision', 'recall', 'f1 - score'])
print()
print("Classification Report")
print(report.mean())
print()
```

```
Training Loss after epoch 6 : 15.649200439453125 Accuracy: 89.31%
Training Loss after epoch 7 : 12.68525218963623 Accuracy: 91.01%
Training Loss after epoch 8 : 11.973682403564453 Accuracy: 91.07%
Training Loss after epoch 9 : 7.540719509124756 Accuracy: 95.29%
Running Fold : 5
Training Loss after epoch 0 : 72.93213653564453 Accuracy: 41.90%
Training Loss after epoch 1 : 41.20624542236328 Accuracy: 67.09%
Training Loss after epoch 2 : 30.886798858642578 Accuracy: 77.14%
Training Loss after epoch 3 : 26.192737579345703 Accuracy: 79.77%
Training Loss after epoch 4 : 19.5633487701416 Accuracy: 86.06%
Training Loss after epoch 5 : 17.526750564575195 Accuracy: 85.93%
Training Loss after epoch 6 : 16.10250473022461 Accuracy: 88.07%
Training Loss after epoch 7 : 14.222383499145508 Accuracy: 88.94%
Training Loss after epoch 8 : 8.573171615600586 Accuracy: 94.16%
Training Loss after epoch 9 : 9.478973388671875 Accuracy: 94.22%

Classification Report
accuracy      0.728519
precision     0.737803
recall        0.729052
f1 - score    0.727841
dtype: float64
```

***Figure 1 Training results***

## 3. Evaluation

CNN model is evaluated using a K fold strategy on training and testing dataset. The model is iterated over 5 folds with 10 epochs each and a batch size of 32. The average accuracy, precision, recall and f1 – score of these folds are calculated. And finally the confusion matrix is visualized using plot_cm() function.

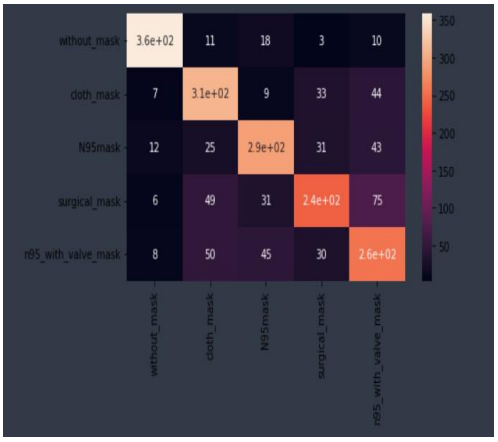


***Figure 2. confusion matrix***

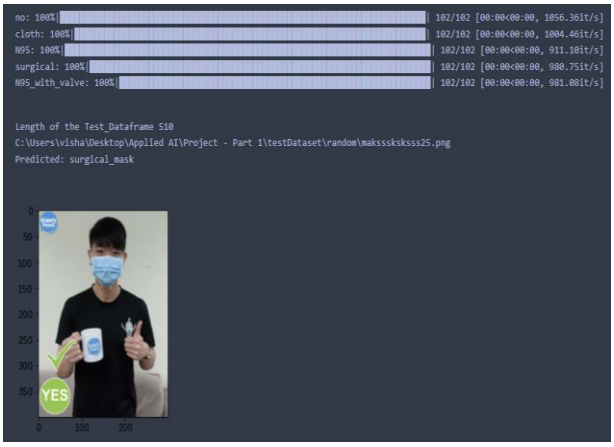


***Figure 3. Test Results***

| Metric | Score |
|---|---|
| **Accuracy** | 0.728519 |
| **Precision** | 0.737803 |
| **Recall** | 0.729052 |
| **F1-Score** | 0.727841 |

From confusion matrix and metric scores we observed the following information

1. Adding more convolution layers will extract more features
2. The model doesn't predict well for Surgical mask

## 4. References


**1.** https://www.kaggle.com/datasets/wobotintelligence/face-mask-detection-dataset

**2.** https://www.kaggle.com/datasets/prithwirajmitra/covid-face-mask-detection-dataset

**3.** https://www.kaggle.com/datasets/dhruvmak/face-mask-detection

**4.** https://humansintheloop.org/resources/datasets/mask-dataset-download/?submissionGuid=168b3ab4-a263-4c08-9c34-8ad672aa6e14

**5.** https://www.kaggle.com/omkargurav/face-mask-dataset

**6.** https://www.kaggle.com/sumansid/facemask-dataset

**7.** https://www.kaggle.com/ashishjangra27/face-mask-12k-images-dataset