A screenshot of a video game

Description automatically generated with medium confidence

**GINA CODY SCHOOL OF ENGINEERING AND**

**COMPUTER SCIENCE**

COMP 6721 – Applied Artificial Intelligence

**AI Face Mask Detector**

**Authors:**

Brijesh Rameshbhai Lakkad (40164018)

Charit Pareshbhai Patel (40160658)

Vatsal Chauhan (………)

**Team Id:** RN\_04

**Presented to:**

Dr. René Witte

Nov 10, 2021

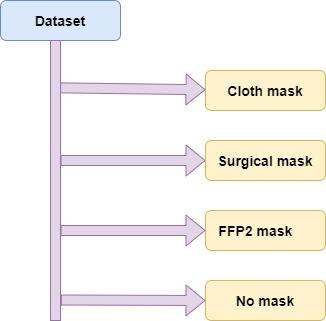
# Dataset

The working of applications in the field of Artificial intelligence is mainly dependent on the datasets. To obtain the best outcomes, we must have a good and balanced dataset. Therefore, we create a good model by training it with a large amount of preprocessed data, and ultimately it can classify new data correctly.

To collect a proper dataset for this project, these two resources have been used:

Dataset 1: Face Mask Detection Dataset - 20 Categories of Masks [1]

Dataset 2: Face Mask Detection - 853 images belonging to 3 classes [2]

We have divided our dataset with 1333 images into four different classes depending upon the type of face mask on the person. Those classes are the person wearing a cloth mask, a surgical mask, an FFP2 mask and no mask.

Both datasets have been used, though Dataset 1 has contributed to the final dataset more than Dataset 2. Dataset 1 included many different categories of data. However, it didn't include the data with the FFP2 mask. That led us to find Dataset 2. And this dataset helped us to collect more data for the face with the FFP2 mask.

### Image Pre-processing

Our Data scientist has processed the data by writing many Python scripts. These scripts mainly included the below tasks:

1. Select the balanced number of images per class.

2. Index each selected image from Dataset 1 and Dataset 2 with storing its name and class to the CSV file.

3. Standardizing the pixel values as raw image data will not give the desired results.

4. Resize each image to create tensors of the same dimensions.

5. Transform raw images into tensors to let the algorithm use them.

We have automated the shuffling and splitting of data into training, validation, and testing sets with the ratio of 80-10-10 percent, respectively.

Our dataset is composed of images of different sizes and colours. Therefore, we are resizing each image in (250,250) dimensions and converting it into an RGB image. The data are also normalized using the pre-defined means and standard variations.

To read the image from the dataset folder, we are taking the help of a CSV file. This file stores the information about the images with their name and target class. So, for each row in the CSV file.

x

The below is a glimpse of dataset.

> train\_df.head()

Out[5]:

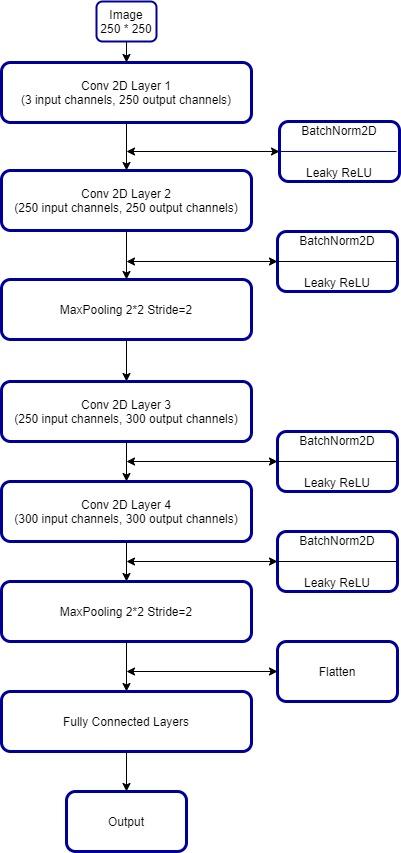
|  | **filename** | **classname** |
| --- | --- | --- |
| **0** | 1001.png | mask\_colorful |
| **1** | 1002.jpg | face\_no\_mask |
| **2** | 1003.png | mask\_surgical |
| **3** | 1004.png | mask\_colorful |
| **4** | 1005.jpg | face\_no\_mask |

Graphical user interface, text, application

Description automatically generated

*Figure 1: Number of images files per class*

# CNN Architecture

In this model, there are 2 CNN blocks, and each block consists of 2 convolution layers and 1 max-pooling layer. As there are no negative values for any pixel value, the project uses the Relu activation function to remove negative values from the feature map and introduce *non-linearity transformation.* [3]

After applying convolution and extracting features from the image, a Dropout zeroes some of the elements of the input tensor with probability p using samples from a Bernoulli distribution. [4] Then 3 linear is added to reduce the size of the tensor and learn the features.

This project uses Adam, a gradient-based stochastic optimization.[5] As we know, by changing the model parameters, like weights, and adding bias, the model can be optimized. Loss function checks whether the model is moving in the correct direction and making progress, whereas optimization improves the model to deliver accurate results. And the learning rate will decide how big the steps should be to change the parameters.

Below are the pre-set hyperparameter values:

*Learning rate*: 0.001

*Epochs*: 50

*Batch size*: 25

*Figure 2: CNN Architecture*

Figures 3 and 4 show the plots of accuracy versus the number of epochs and loss versus the number of epochs, respectively. From figure 3, we can conclude that at 50 epochs, the model has good accuracy than having less than 25 epochs. And we can infer from figure 4 that as the epochs pass, the training and validation loss decreases.

**Chart

Description automatically generated with medium confidence**

*Figure 3: Accuracy vs. No. of epochs*

**A picture containing shape

Description automatically generated**

*Figure 4: Training and validation loss vs. No. of epochs*

**Evaluation:**

After preprocessing datasets and designing CNN architecture, the next task is to train the model using the training dataset and evaluate its performance with a testing dataset. Here we have trained our model with four epochs. For every epoch, the learning rate is 0.01 which is used to set the weights for each epoch. We evaluated the performance of our model using recall, precision, accuracy, and f1-score for all the phases. Here, it is clearly evident that the accuracy of our model is (.........) for the training phase, and for the testing, the accuracy is (........). Confusion matrix shows that some of the images are misclassified, this is because ……………………

**References:**

* [3] <https://medium.com/thecyphy/train-cnn-model-with-pytorch-21dafb918f48>
* [1] <https://www.kaggle.com/wobotintelligence/face-mask-detection-dataset>
* [4] <https://pytorch.org/docs/stable/generated/torch.nn.Dropout.html>
* [5] <https://arxiv.org/abs/1412.6980>
* [2] <https://www.kaggle.com/andrewmvd/face-mask-detection>