COMP6721 Applied Artificial Intelligence (Fall 2021)
Project Assignment, Part I

AI Face Mask Detector

# Team Information

Team Name:
NS_08

Team Members:
- Rahul Patil(**40166394**) - Training Specialist
- Yash Shah(**40163452**) - Data Specialist
- Rucha Prajapati(**40170922**) - Evaluation Specialist

# Resources

Link to Repository : https://github.com/iRahulP/FaceMask__AI

Link to Dataset :
https://drive.google.com/file/d/1UDvUL7nuzt3JStnN1KOI_onC5h7q6H9O/view?usp=sharing

Link to Colab Notebook :
https://colab.research.google.com/drive/1FQGmTS9ClqgZnWCFMV-7S3me0D2Vshi0?authuser=2

DataSet

Dataset here is used for training and testing purpose of deep learning model for detection of the person wearing mask or not and the type of mask one is wearing

Dataset includes images without mask, with mask, with cloth_masks, with surgical_masks, with ffp2_masks.

Approximately 500 images for each class and 800 images for without masks class.

We have created our Dataset by using multiple sources and combining them to create appropriate dataset.

Current submission has less data as compared to the original dataset, the size for the zip file exceeded 250MB. Original dataset can be found at : https://drive.google.com/file/d/1UDvUL7nuzt3JStnN1KOI_onC5h7q6H9O/view?usp=sharing

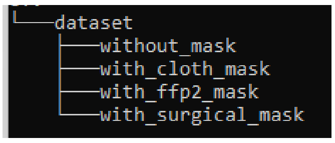
References:

- https://www.kaggle.com/omkargurav/face-mask-dataset just provided all the majority of the images for our dataset.it has people with and without mask and with different categories of mask.
- https://www.google.com/search?q=images+of+people+with+ffp2+masks&tbm=isch&ved=2ahUKEwjg9J7ur4z0AhVugXIEHVNDDOQQ2-cCegQIABAA&oq=images+of+people+with+ffp2+masks&gs_lcp=CgNpbWcQAzoFCAAQgARQ3qqUQ1i6vJRDYM6_lENoAXAAeACAAbIBiAHtB5IBAzIuNpgBAKABAaoBC2d3cy13aXotaW1nwAEB&sclient=img&ei=IvyKYeDbJ-6CytMP04axoA4&bih=692&biw=1600&rlz=1C1CHZN_enIN940IN940
- https://makeml.app/datasets/mask
- https://towardsdatascience.com/masktheface-cv-based-tool-to-mask-face-dataset-1a71d5b68703 - Images were manually created using the tool provided for ffp2 masks dataset using open source face images from unsplash.

# CNN ARCHITECTURE

1) Data Extraction:

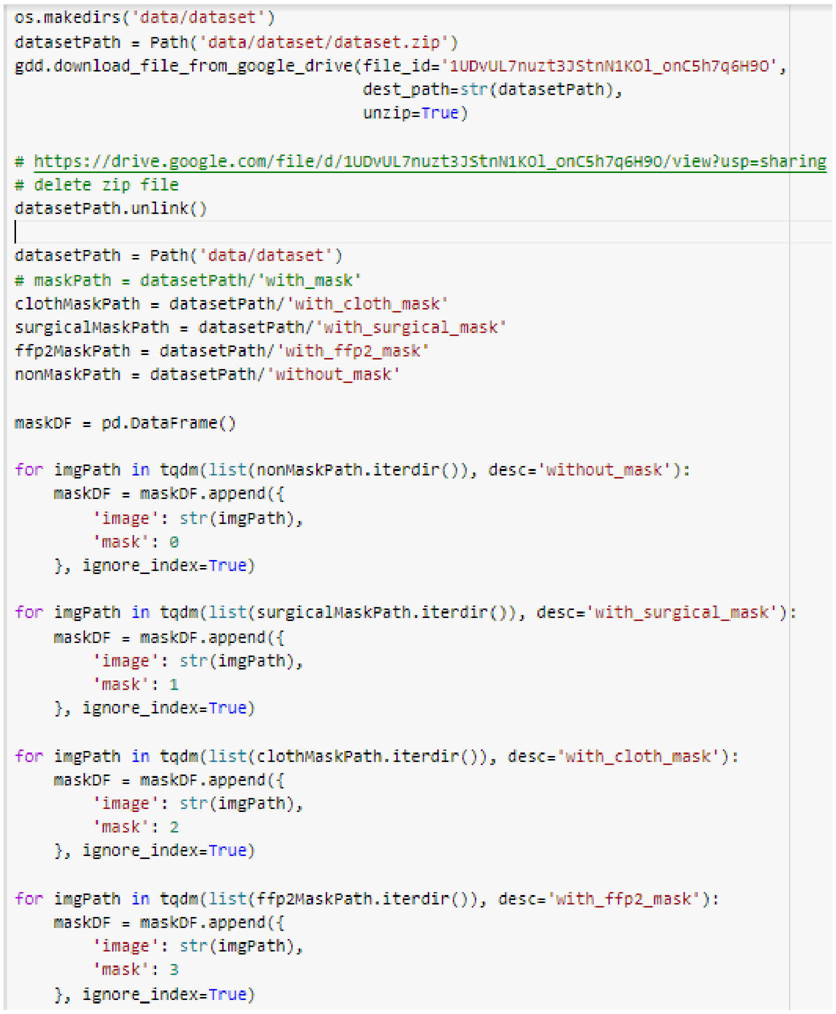
Downloading the dataset from the drive and unzipping dataset will look as follow:



```
└──dataset
    ├──without_mask
    ├──with_cloth_mask
    ├──with_ffp2_mask
    └──with_surgical_mask
```

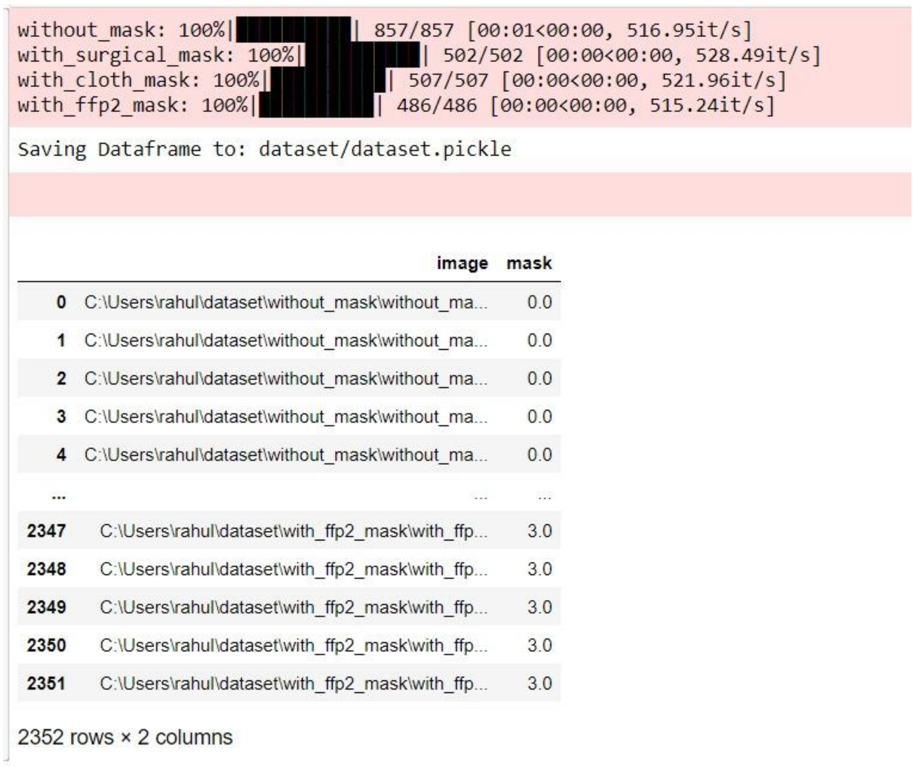
DataFrames is created by iterating all images recursively in every folder and labeling them as follows :

0 if **without mask**,**1** with **surgical mask**, **2** with **cloth mask**, **3** with **ffp2 mask**.



```
os.makedirs('data/dataset')
datasetPath = Path('data/dataset/dataset.zip')
gdd.download_file_from_google_drive(file_id='1UDvUL7nuzt3JStnN1KOl_onC5h7q6H9O',
                                    dest_path=str(datasetPath),
                                    unzip=True)

# https://drive.google.com/file/d/1UDvUL7nuzt3JStnN1KOl_onC5h7q6H9O/view?usp=sharing
# delete zip file
datasetPath.unlink()

datasetPath = Path('data/dataset')
# maskPath = datasetPath/'with_mask'
clothMaskPath = datasetPath/'with_cloth_mask'
surgicalMaskPath = datasetPath/'with_surgical_mask'
ffp2MaskPath = datasetPath/'with_ffp2_mask'
nonMaskPath = datasetPath/'without_mask'

maskDF = pd.DataFrame()

for imgPath in tqdm(list(nonMaskPath.iterdir()), desc='without_mask'):
    maskDF = maskDF.append({
        'image': str(imgPath),
        'mask': 0
    }, ignore_index=True)

for imgPath in tqdm(list(surgicalMaskPath.iterdir()), desc='with_surgical_mask'):
    maskDF = maskDF.append({
        'image': str(imgPath),
        'mask': 1
    }, ignore_index=True)

for imgPath in tqdm(list(clothMaskPath.iterdir()), desc='with_cloth_mask'):
    maskDF = maskDF.append({
        'image': str(imgPath),
        'mask': 2
    }, ignore_index=True)

for imgPath in tqdm(list(ffp2MaskPath.iterdir()), desc='with_ffp2_mask'):
    maskDF = maskDF.append({
        'image': str(imgPath),
        'mask': 3
    }, ignore_index=True)
```

Output DataFrame:



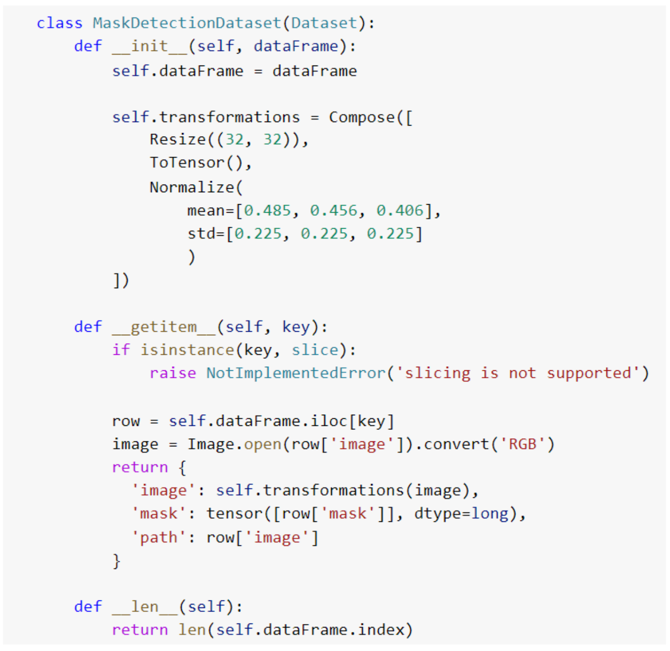
```
without_mask: 100%|██████████| 857/857 [00:01<00:00, 516.95it/s]
with_surgical_mask: 100%|██████████| 502/502 [00:00<00:00, 528.49it/s]
with_cloth_mask: 100%|██████████| 507/507 [00:00<00:00, 521.96it/s]
with_ffp2_mask: 100%|██████████| 486/486 [00:00<00:00, 515.24it/s]
```

```
Saving Dataframe to: dataset/dataset.pickle
```

	image	mask
0	C:\Users\rahul\dataset\without_mask\without_ma...	0.0
1	C:\Users\rahul\dataset\without_mask\without_ma...	0.0
2	C:\Users\rahul\dataset\without_mask\without_ma...	0.0
3	C:\Users\rahul\dataset\without_mask\without_ma...	0.0
4	C:\Users\rahul\dataset\without_mask\without_ma...	0.0
...	...	...
2347	C:\Users\rahul\dataset\with_ffp2_mask\with_ffp...	3.0
2348	C:\Users\rahul\dataset\with_ffp2_mask\with_ffp...	3.0
2349	C:\Users\rahul\dataset\with_ffp2_mask\with_ffp...	3.0
2350	C:\Users\rahul\dataset\with_ffp2_mask\with_ffp...	3.0
2351	C:\Users\rahul\dataset\with_ffp2_mask\with_ffp...	3.0

2352 rows × 2 columns

Dataset class: Having panda dataframe, MaskDetectionDataset class is used to query images by pytorch. All images are transformed using torchvision.transforms resizing them to 32*32, converting to torch Tensor and normalizing them with standard mean and std values which then forces the network to be between 0-1 values.



```python
class MaskDetectionDataset(Dataset):
    def __init__(self, dataFrame):
        self.dataFrame = dataFrame

        self.transformations = Compose([
            Resize((32, 32)),
            ToTensor(),
            Normalize(
                mean=[0.485, 0.456, 0.406],
                std=[0.225, 0.225, 0.225]
                )
        ])

    def __getitem__(self, key):
        if isinstance(key, slice):
            raise NotImplementedError('slicing is not supported')

        row = self.dataFrame.iloc[key]
        image = Image.open(row['image']).convert('RGB')
        return {
          'image': self.transformations(image),
          'mask': tensor([row['mask']], dtype=long),
          'path': row['image']
        }

    def __len__(self):
        return len(self.dataFrame.index)
```

Defining CNN model: In the model, we use nn.module that helps to build neural network models.

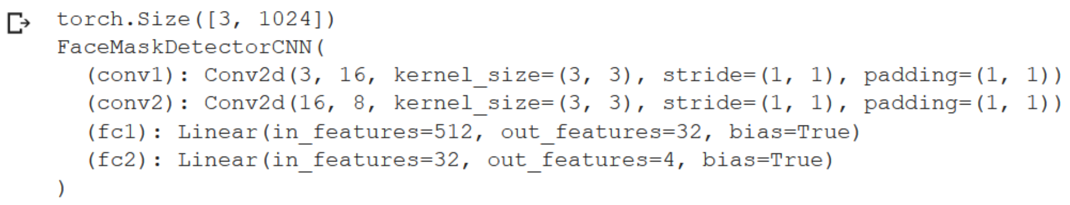
In __init__() we define the layers and subsequent dimensions within them for the CNN as follows :

1) 2 convolution layers (for channels):
    - **Conv1** applies Conv2d with input channels as 3(for RGB image) and output channels as 16.
    - With a kernel size of 3 and padding of 1: the dimensions for the 32*32 images remain the same. `# 32 - 3 + 1 + 2*1`
    - Same goes for **conv2** for input channels of 16 and output channels as 8.

2) 2 fully connected layers (for features):
    - fc1 applies linear with input of (8*8*8) to output features as 32.
    - With fc2 further reducing it to 4 as required..

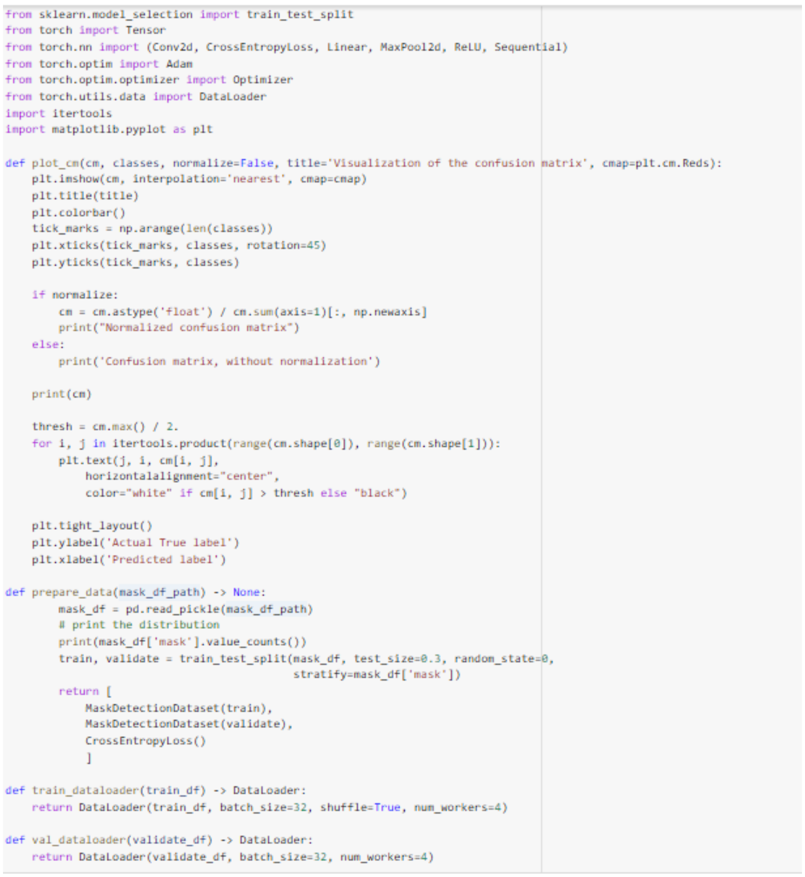
Further we carry **forward pass** in the forward() method, which does actual transformation of the network by passing the data into the computation graph (i.e. neural network) which maps an input tensor to a prediction output tensor as follows in current CNN:

- Initial out calls torch.nn.functional's **max_pool2d** along with the evaluation of conv1 from convolution layer with a **tanh** activation function with a stride of 2, thereby reducing the overall dimensions of the current image to half.
  `# 32 -> 16`
- Further computing output for a relu activation function.
- Action carried forward for conv2, thereby updating dimensions of image:
  `# 16 -> 8`
- As a result we have input dimensions of 8*8 to the fully connected layer at fc1.
- Flatten: reshape the features and pass this to the **fc1.**
- Carry out tanh activation element wise for the above features.
- Finally passing to the output layer and returning the output.



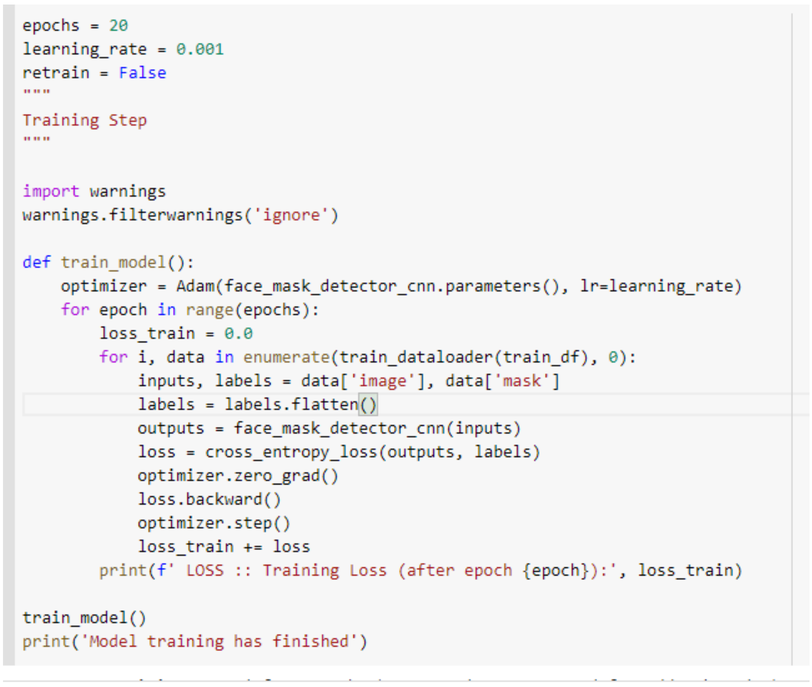
```
torch.Size([3, 1024])
FaceMaskDetectorCNN(
  (conv1): Conv2d(3, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv2): Conv2d(16, 8, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (fc1): Linear(in_features=512, out_features=32, bias=True)
  (fc2): Linear(in_features=32, out_features=4, bias=True)
)
```

Preparation of Dataset by splitting:.Dataset is splitted in 70:30 for training and testing.For this purpose we use the train_test_split() function and pass a dataset label to its parameter. Loading of data is done using Dataloader and uses it as a batch of 32 at the time for training the model.To reduce the time of loading the batch samples, number of workers is set to 4.



```python
from sklearn.model_selection import train_test_split
from torch import Tensor
from torch.nn import (Conv2d, CrossEntropyLoss, Linear, MaxPool2d, ReLU, Sequential)
from torch.optim import Adam
from torch.optim.optimizer import Optimizer
from torch.utils.data import DataLoader
import itertools
import matplotlib.pyplot as plt

def plot_cm(cm, classes, normalize=False, title='Visualization of the confusion matrix', cmap=plt.cm.Reds):
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    print(cm)

    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
            horizontalalignment="center",
            color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('Actual True label')
    plt.xlabel('Predicted label')

def prepare_data(mask_df_path) -> None:
        mask_df = pd.read_pickle(mask_df_path)
        # print the distribution
        print(mask_df['mask'].value_counts())
        train, validate = train_test_split(mask_df, test_size=0.3, random_state=0,
                                           stratify=mask_df['mask'])
        return [
            MaskDetectionDataset(train),
            MaskDetectionDataset(validate),
            CrossEntropyLoss()
            ]

def train_dataloader(train_df) -> DataLoader:
    return DataLoader(train_df, batch_size=32, shuffle=True, num_workers=4)

def val_dataloader(validate_df) -> DataLoader:
    return DataLoader(validate_df, batch_size=32, num_workers=4)
```

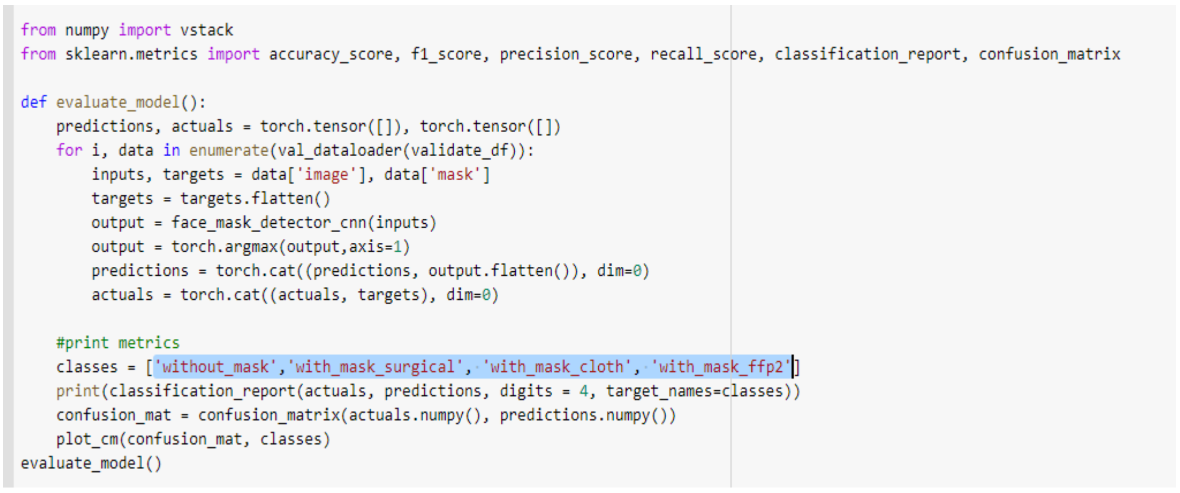
Training the Model: We use Adam to optimize its parameters and the learning rate is set to 0.001. Then, we iterate through the training dataset and retrieve batches to get images and labels, pass it to the model and compute the loss of that batch. The Number of epochs is 20 for this project which can be changed.



```python
epochs = 20
learning_rate = 0.001
retrain = False
"""
Training Step
"""

import warnings
warnings.filterwarnings('ignore')

def train_model():
    optimizer = Adam(face_mask_detector_cnn.parameters(), lr=learning_rate)
    for epoch in range(epochs):
        loss_train = 0.0
        for i, data in enumerate(train_dataloader(train_df), 0):
            inputs, labels = data['image'], data['mask']
            labels = labels.flatten()
            outputs = face_mask_detector_cnn(inputs)
            loss = cross_entropy_loss(outputs, labels)
            optimizer.zero_grad()
            loss.backward()
            optimizer.step()
            loss_train += loss
        print(f' LOSS :: Training Loss (after epoch {epoch}):', loss_train)

train_model()
print('Model training has finished')
```

Evaluate the model: We use matplotlib which predicts labels on X-axis and actual labels on Y-axis.First create empty tensors each for prediction and actual data, iterate through validate dataset and retrieve batches to get images and labels to evaluate the accuracy on unseen images.Then, we concatenate the sequence of tensors in the given dimension.

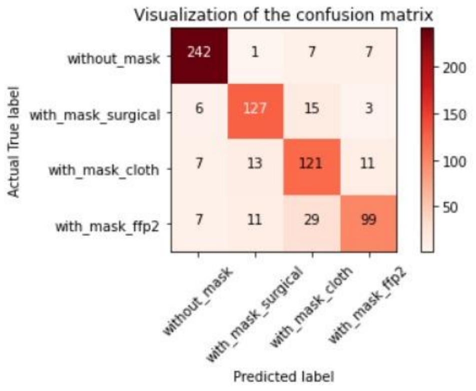
Afterwards we define four classes (without_mask, with_mask_cloth, with_mask_surgical and with_mask_ffp2) and plot confusion matrix having actuals and predictions on its axis.



```
from numpy import vstack
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score, classification_report, confusion_matrix

def evaluate_model():
    predictions, actuals = torch.tensor([]), torch.tensor([])
    for i, data in enumerate(val_dataloader(validate_df)):
        inputs, targets = data['image'], data['mask']
        targets = targets.flatten()
        output = face_mask_detector_cnn(inputs)
        output = torch.argmax(output,axis=1)
        predictions = torch.cat((predictions, output.flatten()), dim=0)
        actuals = torch.cat((actuals, targets), dim=0)

    #print metrics
    classes = ['without_mask','with_mask_surgical', 'with_mask_cloth', 'with_mask_ffp2']
    print(classification_report(actuals, predictions, digits = 4, target_names=classes))
    confusion_mat = confusion_matrix(actuals.numpy(), predictions.numpy())
    plot_cm(confusion_mat, classes)
evaluate_model()
```

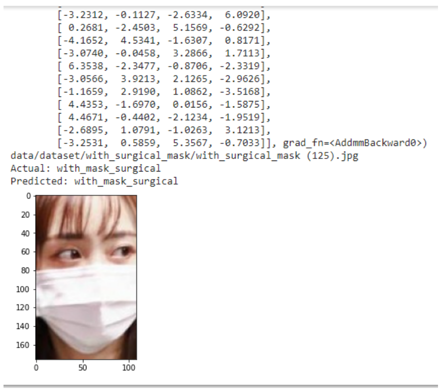
Output of Confusion Matrix:

```
                  precision    recall  f1-score   support

    without_mask       0.92      0.94      0.93       257
with_mask_surgical     0.84      0.84      0.84       151
   with_mask_cloth     0.70      0.80      0.75       152
    with_mask_ffp2     0.82      0.68      0.74       146

        accuracy                           0.83       706
       macro avg       0.82      0.81      0.82       706
    weighted avg       0.84      0.83      0.83       706

Confusion matrix, without normalization
[[242   1   7   7]
 [  6 127  15   3]
 [  7  13 121  11]
 [  7  11  29  99]]
```



Prediction of random image using trained Model:

Iteration over one batch is done of 32 images of the test data frame(validate_df) and after loading and running one image is selected at random from it using Random.choice() and it is plotted in a graph and its actual and predicted values are obtained.



```
        [-3.2312, -0.1127, -2.6334,  6.0920],
        [ 0.2681, -2.4503,  5.1569, -0.6292],
        [-4.1652,  4.5341, -1.6307,  0.8171],
        [-3.0740, -0.0458,  3.2866,  1.7113],
        [ 6.3538, -2.3477, -0.8706, -2.3319],
        [-3.0566,  3.9213,  2.1265, -2.9626],
        [-1.1659,  2.9190,  1.0862, -3.5168],
        [ 4.4353, -1.6970,  0.0156, -1.5875],
        [ 4.4671, -0.4402, -2.1234, -1.9519],
        [-2.6895,  1.0791, -1.0263,  3.1213],
        [-3.2531,  0.5859,  5.3567, -0.7033]], grad_fn=<AddmmBackward0>)
data/dataset/with_surgical_mask/with_surgical_mask (125).jpg
Actual: with_mask_surgical
Predicted: with_mask_surgical
```

Cloud Model vs Local Model :The colab version of the cnn model is specific to computation power of cpu, whereas the local model had ambiguities for the same as to usage of cpu and gpu devices resulting in runtime error of : Input type (torch.cuda.FloatTensor) and weight type (torch.FloatTensor) should be the same. Thus the local python notebook has been configured to use cudo:0 device for all computations and num_workers were also kept 0 in case of running the instance on windows pc.

Future Prospects:

We will be increasing our training set and increase the hidden layers, and possibly update mapping of activation functions which will possibly further increase the performance and checking if the system is bias to specific age,race,gender or any other factor and to improve it by adding images to our training dataset.

# References:

Following are all the resources being referred and used for implementation as well as error resolution for the project:

- https://stackoverflow.com/questions/50480689/pytorch-torchvision-brokenpipeerror-errno-32-broken-pipe
- https://discuss.pytorch.org/t/runtimeerror-input-type-torch-cuda-floattensor-and-weight-type-torch-floattensor-should-be-the-same/21782
- https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html
- https://towardsdatascience.com/how-i-built-a-face-mask-detector-for-covid-19-using-pytorch-lightning-67eb3752fd61
- https://tqdm.github.io/
- https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html
- https://pytorch.org/docs/stable/nn.html#convolution-layers
- https://towardsdatascience.com/pytorch-layer-dimensions-what-sizes-should-they-be-and-why-4265a41e01fd
- https://stats.stackexchange.com/questions/296679/what-does-kernel-size-mean/339265
- https://www.analyticsvidhya.com/blog/2019/10/building-image-classification-models-cnn-pytorch/
- https://www.analyticsvidhya.com/blog/2020/10/what-is-the-convolutional-neural-network-architecture/
- https://towardsdatascience.com/pytorch-layer-dimensions-what-sizes-should-they-be-and-why-4265a41e01fd