IIT Project

Face mask Detection

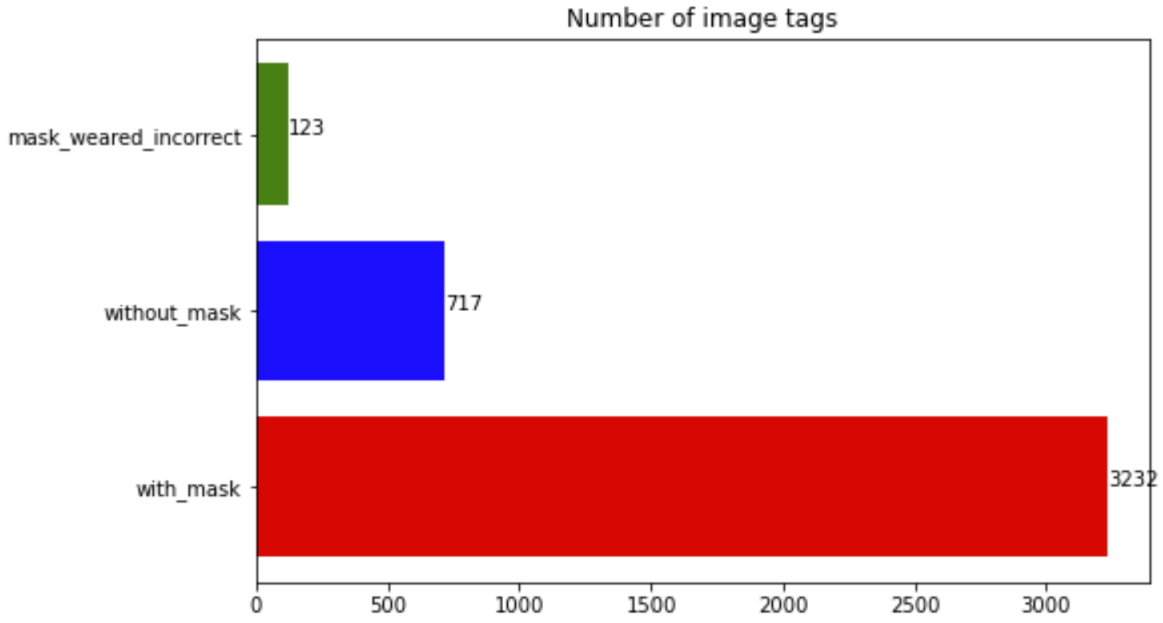
Abijith Pradeep

Objective

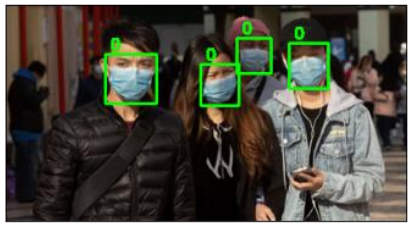
- To detect the faces given a dataset and predict the bounding box of the face and tell which category it belongs to.

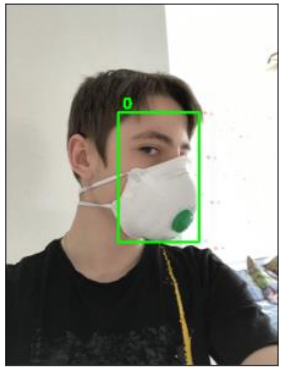
Dataset

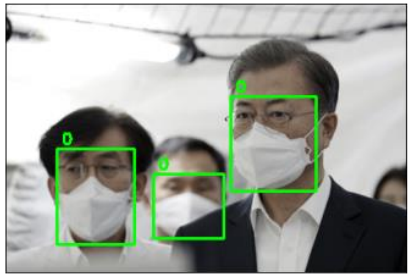
- Face –mask –detection from Kaggle was used.
- Total of images: 4072
- Masked: 3232, No mask: 717, Mask incorrectly worn: 123

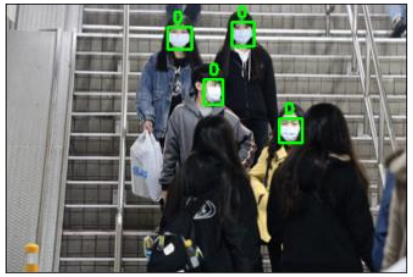


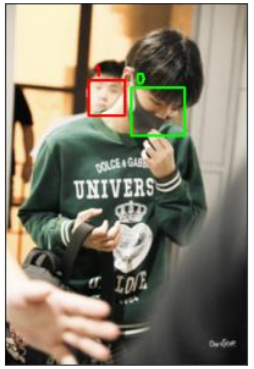
Sample data











Splitting Data

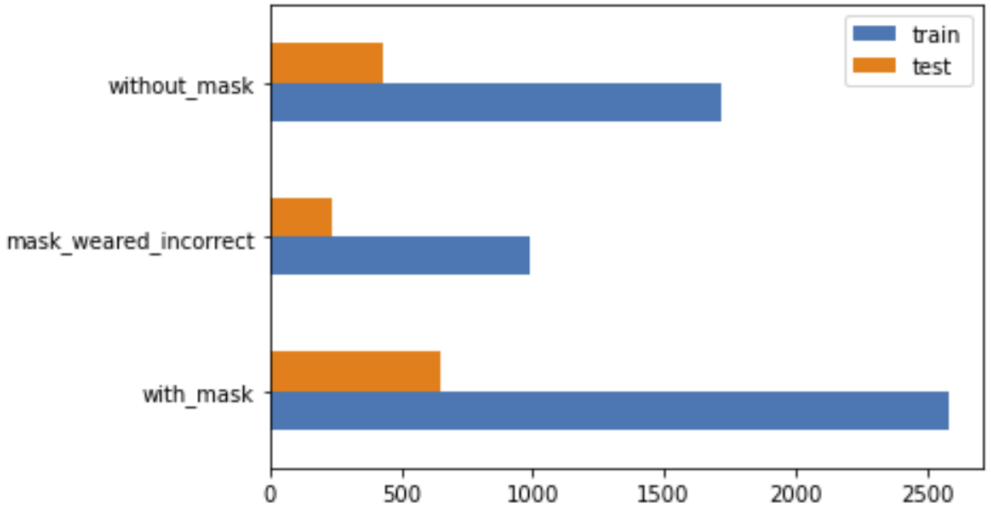
```
x = []
y = []
IMG_SIZE = (64,64)

for _data in data:

  img_path = _data['img_path']
  for (xmin, ymin, xmax, ymax, label) in _data['objs']:
    img = cv2.imread(img_path)
    crop_img = img[ymin : ymax, xmin : xmax]
    re_img = cv2.resize(crop_img, IMG_SIZE)
    re_img = re_img/255
    target = to_categorical(label, num_classes=3)

    if label == 2:
      aug_img = augment_data(re_img, aug_model, iterate=10)
      for aug in aug_img:
        x.append(np.array(aug)); y.append(target)
    elif label == 1:
        aug_img = augment_data(re_img, aug_model, iterate=3)
        for aug in aug_img:
            x.append(np.array(aug)); y.append(target)
    else:
        x.append(re_img); y.append(target)
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=7)
```

Train Test Split

```
x_train shape: (5290, 64, 64, 3)
x_test shape: (1323, 64, 64, 3)
y_train shape: (5290, 3)
y_test shape: (1323, 3)
```

Train vs Test: Category count




```
in train data:
 with mask: 48.81 %, without mask: 32.46 %, mask weared incorrect: 18.73 %
in test data:
 with mask: 49.13 %, without mask: 32.80 %, mask weared incorrect: 18.07 %
```

Model

- Simple Sequential Model used.
- Optimizer: Adam, Loss: Categorical crossentropy, Metrics: Accuracy
- Layers: Conv2D, Maxpooling2D, Dropout, Flatten and Dense

```
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

```
Epoch 1/15
166/166 [==============================] - 215s 1s/step - loss: 1.3989 - accuracy: 0.7800 - val_loss: 0.3910 - val_accuracy: 0.8450
Epoch 2/15
166/166 [==============================] - 209s 1s/step - loss: 0.3581 - accuracy: 0.8769 - val_loss: 0.2815 - val_accuracy: 0.8889
Epoch 3/15
166/166 [==============================] - 211s 1s/step - loss: 0.2643 - accuracy: 0.9053 - val_loss: 0.3527 - val_accuracy: 0.8443
Epoch 4/15
166/166 [==============================] - 209s 1s/step - loss: 0.2053 - accuracy: 0.9267 - val_loss: 0.3817 - val_accuracy: 0.8526
Epoch 5/15
166/166 [==============================] - 211s 1s/step - loss: 0.1695 - accuracy: 0.9452 - val_loss: 0.1768 - val_accuracy: 0.9395
Epoch 6/15
166/166 [==============================] - 210s 1s/step - loss: 0.1332 - accuracy: 0.9599 - val_loss: 0.1480 - val_accuracy: 0.9539
Epoch 7/15
166/166 [==============================] - 211s 1s/step - loss: 0.1176 - accuracy: 0.9620 - val_loss: 0.2184 - val_accuracy: 0.9297
Epoch 8/15
166/166 [==============================] - 209s 1s/step - loss: 0.1065 - accuracy: 0.9681 - val_loss: 0.2190 - val_accuracy: 0.9282
Epoch 9/15
166/166 [==============================] - 210s 1s/step - loss: 0.0882 - accuracy: 0.9720 - val_loss: 0.3404 - val_accuracy: 0.9259
Epoch 10/15
166/166 [==============================] - 211s 1s/step - loss: 0.0863 - accuracy: 0.9726 - val_loss: 0.0928 - val_accuracy: 0.9728
Epoch 11/15
166/166 [==============================] - 211s 1s/step - loss: 0.0503 - accuracy: 0.9853 - val_loss: 0.1235 - val_accuracy: 0.9660
Epoch 12/15
166/166 [==============================] - 212s 1s/step - loss: 0.0676 - accuracy: 0.9769 - val_loss: 0.0874 - val_accuracy: 0.9728
Epoch 13/15
166/166 [==============================] - 211s 1s/step - loss: 0.1303 - accuracy: 0.9620 - val_loss: 0.1549 - val_accuracy: 0.9478
Epoch 14/15
166/166 [==============================] - 212s 1s/step - loss: 0.0565 - accuracy: 0.9822 - val_loss: 0.2381 - val_accuracy: 0.9350
Epoch 15/15
166/166 [==============================] - 210s 1s/step - loss: 0.0388 - accuracy: 0.9896 - val_loss: 0.0433 - val_accuracy: 0.9894
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 62, 62, 32)        896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 62, 62, 32)        0
_________________________________________________________________
dropout (Dropout)            (None, 62, 62, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 60, 60, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 60, 60, 64)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 60, 60, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 58, 58, 128)       73856
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 58, 58, 128)       0
_________________________________________________________________
dropout_2 (Dropout)          (None, 58, 58, 128)       0
_________________________________________________________________
flatten (Flatten)            (None, 430592)            0
_________________________________________________________________
dense (Dense)                (None, 128)               55115904
_________________________________________________________________
dense_1 (Dense)              (None, 64)                8256
_________________________________________________________________
dropout_3 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_2 (Dense)              (None, 3)                 195
=================================================================
Total params: 55,217,603
Trainable params: 55,217,603
Non-trainable params: 0
_________________________________________________________________
```

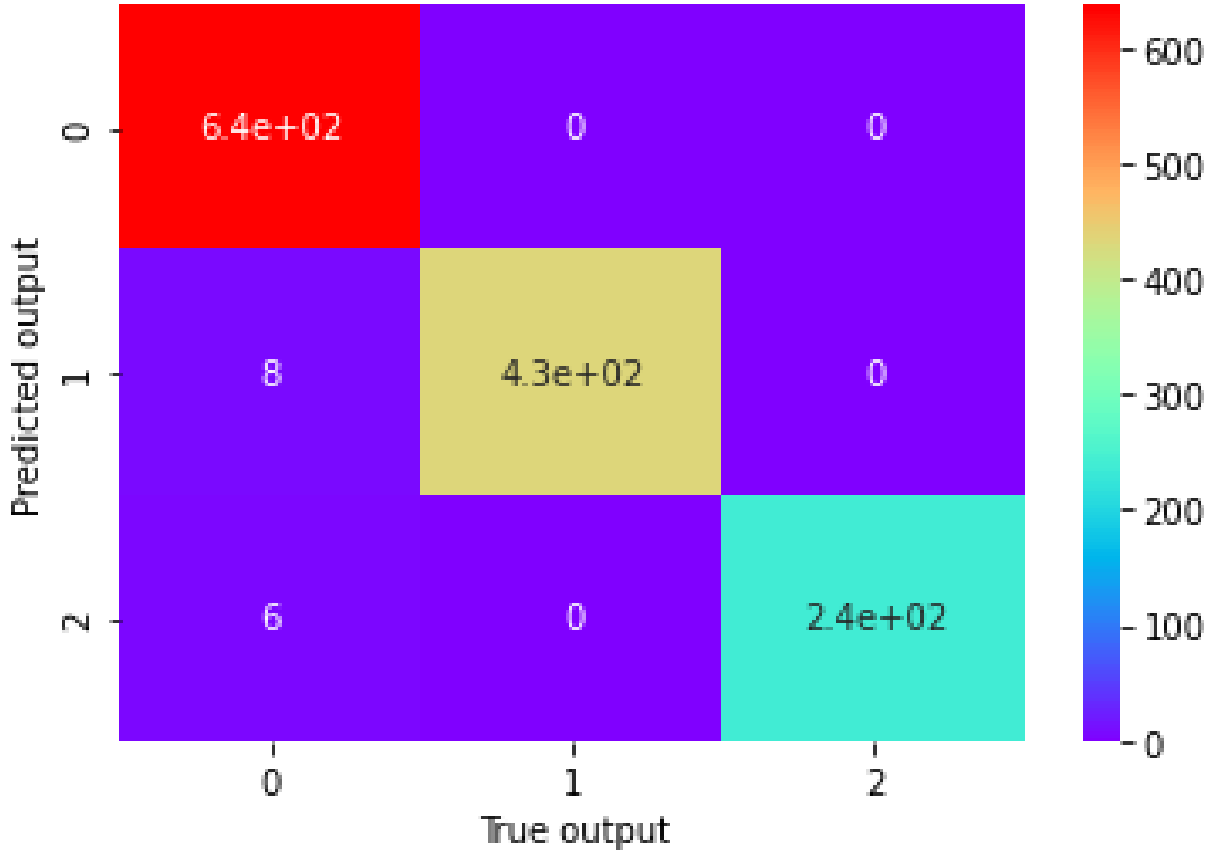
Evaluation on test data

[19]:
```
model.evaluate(x_test, y_test)
```

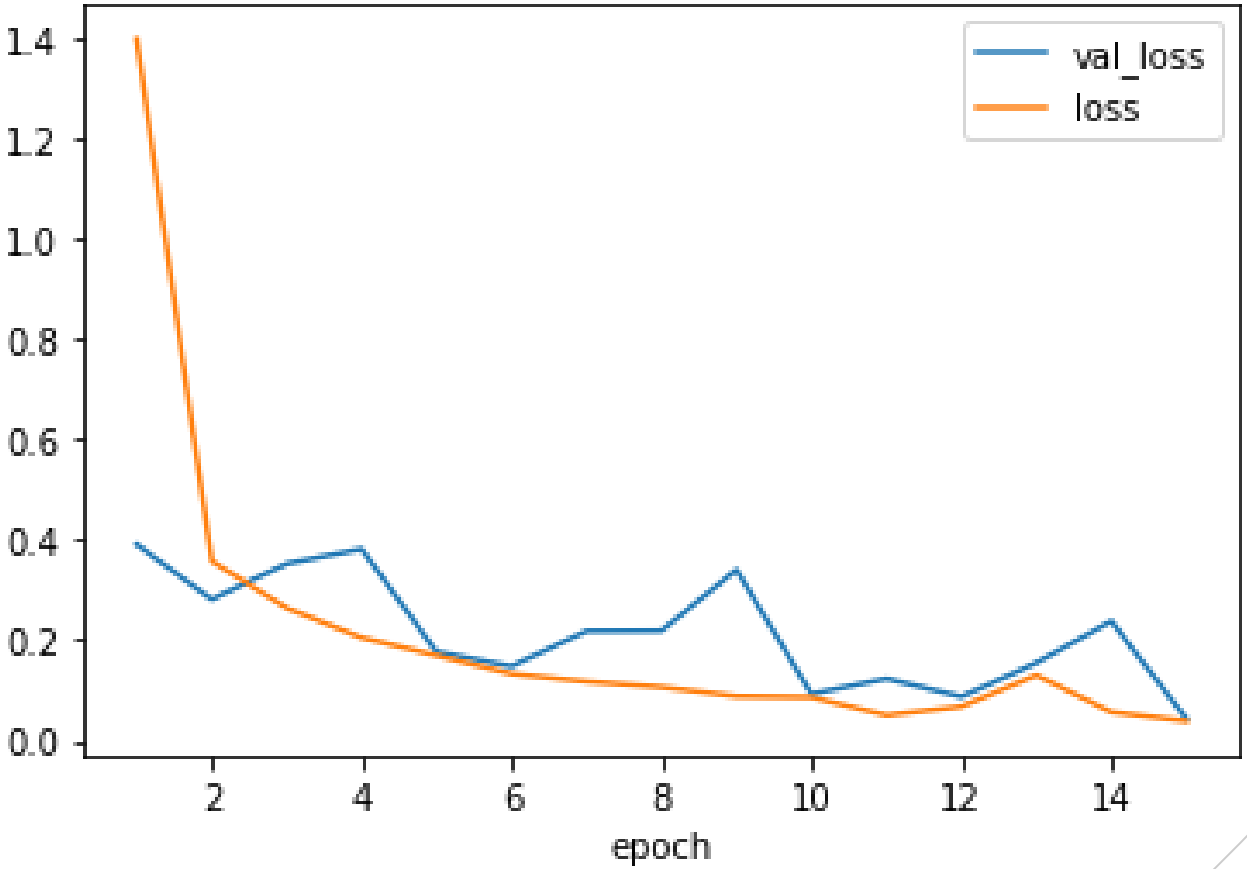
```
42/42 [==============================] - 7s 175ms/step - loss: 0.0433 - accuracy: 0.9894
```

Out[19]:
```
[0.043334100395441055, 0.9894179701805115]
```

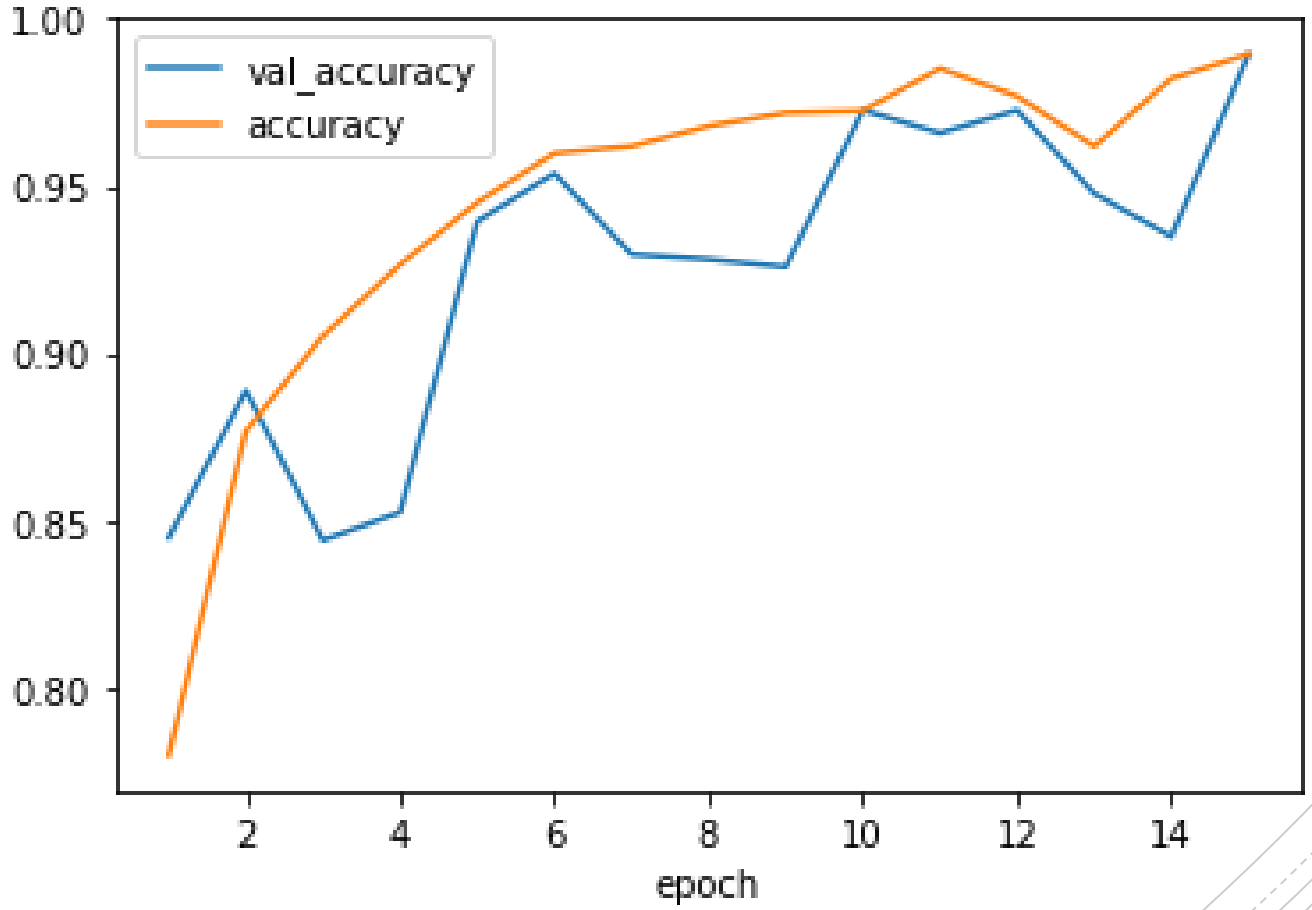
Confusion matrix



Loss Plot



Accuracy plot



Improvising

- Since the detector now predicts and identifies the type of class the person belongs, we can give it an organizational access.
- In other words, get this detector at the entrance of factory/company to check if the employees are coming with correct way, i.e. mask present and covering the mouth and nose.
- It won't allow entry if the person isn't wearing the mask incorrectly.
- Since it takes some time to train the model, we can save this model and use it later on for other purposes.