FACE MASK DETECTION USING CONVOLUTIONAL NEURAL NETWORKS

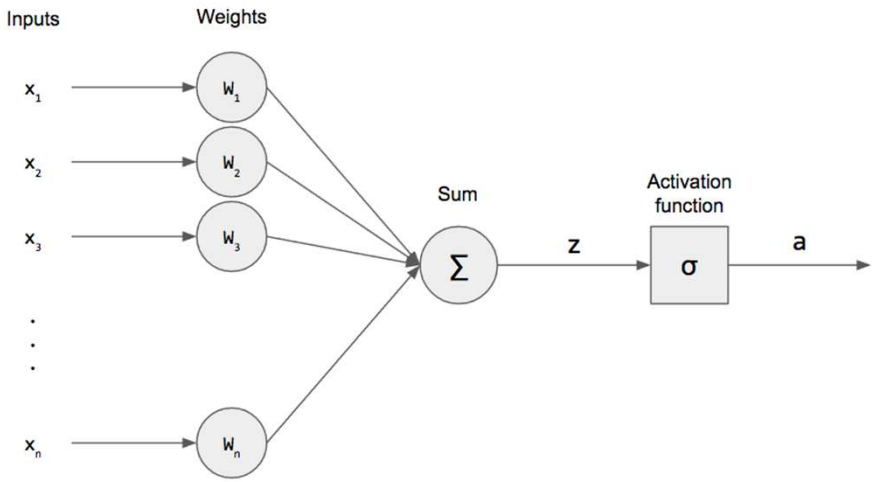
2023ITB018 SWASTIKA JAS
2023ITB019 BURA KULASEKHAR
2023ITB021 GAURVI DOHARE
UNDER THE SUPERVISION OF
PROFESSOR ARINDAM BISWAS

Table of contents

1. FUNDAMENTALS :

CONCEPTUAL UNDERSTANDING

PERCEPTRON



NEURON

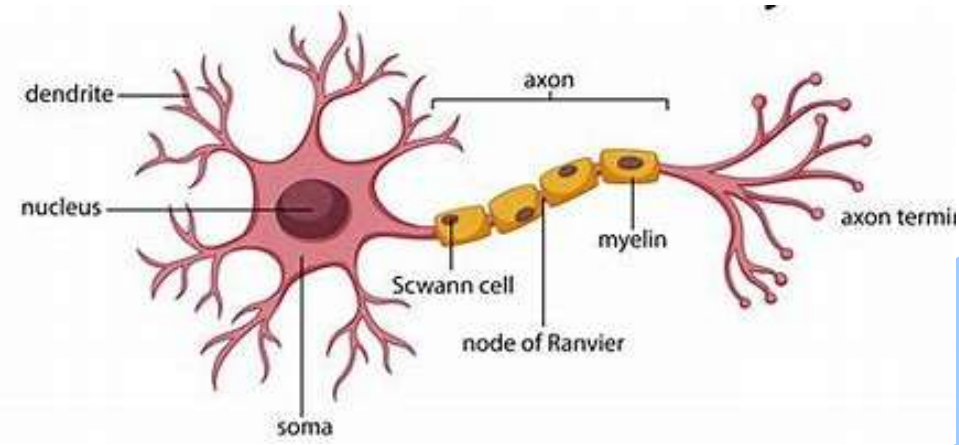
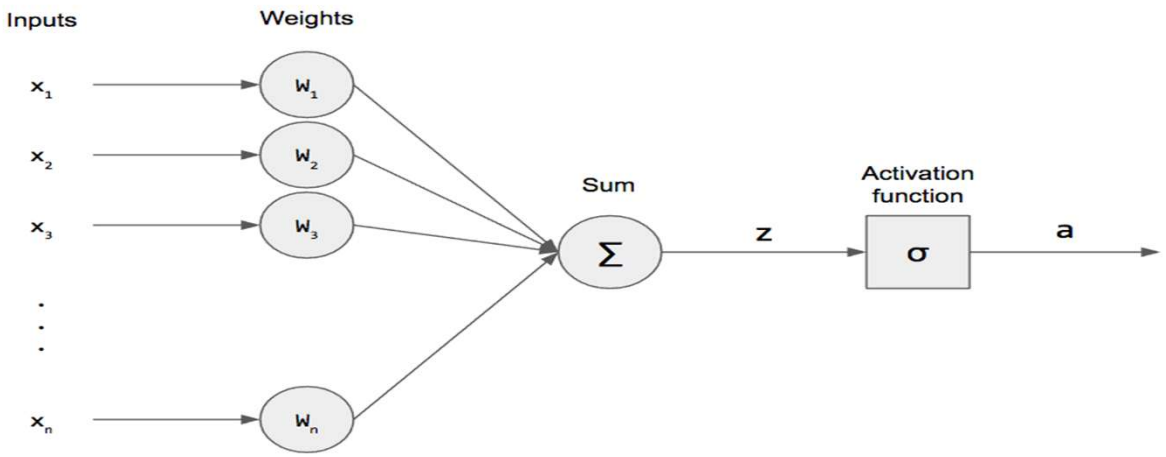


Image source: https://gamedevacademy.org/wp-content/uploads/2017/09/Single-Perceptron.png

Image source: https://static.vecteezy.com/system/resources/previews/000/358/962/large_2x/vector-diagram-of-neuron-anatomy.jpg

PERCEPTRON

❖ What is Perceptron?



❖ **Structure of a Perceptron** :

1. Inputs(x_1,x_2,...,x_n)
2. Weights(w_1,w_2,...,w_n)
3. Summation Function(∑): Computes the weighted sum of inputs:

$$z = \sum_{i=1}^{n} w_i \cdot x_i + b$$

4. where b is the bias term

ACTIVATION and ACTIVATION FUNCTIONS

❖ **WHAT IS ACTIVATION?**

❖ **WHAT IS ACTIVATION FUNCTION?**

Properties of Activation Functions:

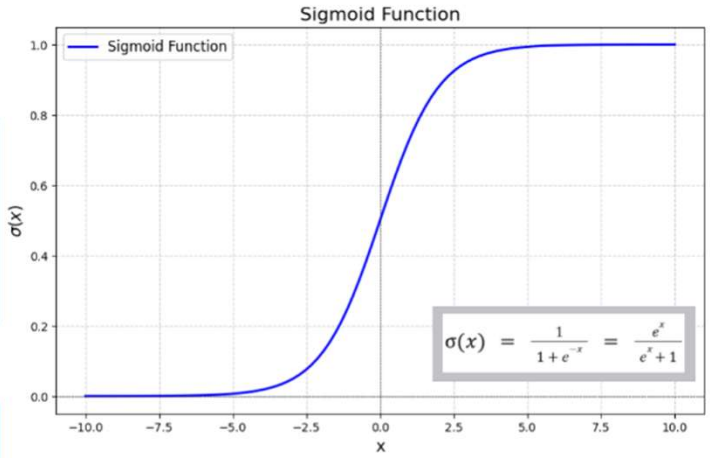
- Non–Linearity
- Differentiability
- Monotonicity
- Non-Saturation
- Computational Efficiency
- Output range

TYPES OF ACTIVATION FUNCTIONS

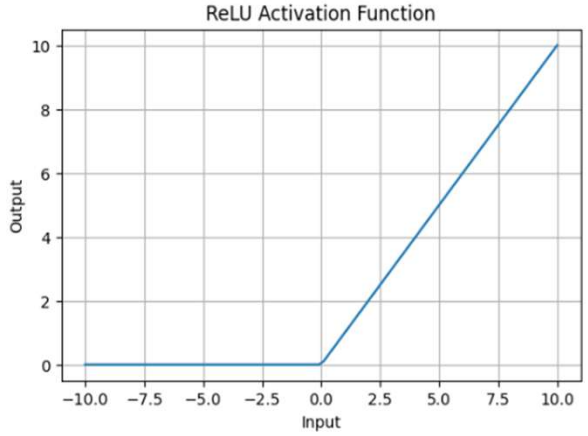
➢ **Linear and Non-linear**

Examples of Non-linear activation functions:

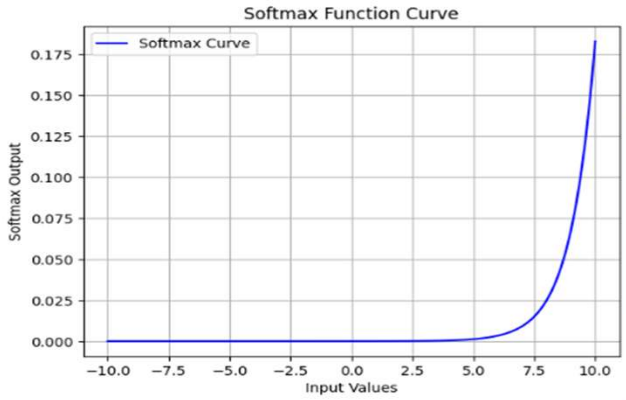
(***img src***: *GEEKS FOR GEEKS)*



Graph of Sigmoid Activation Function



ReLU Activation Function



Softmax Activation Function

BACKPROPAGATION

- WHAT IS BACKPROPAGATION ?
- HOW IT WORKS ?
- Key Concepts in Backpropagation:
  - Loss Function
  - Gradient Descent
  - Chain Rule
  - Learning Rate
  - Epochs

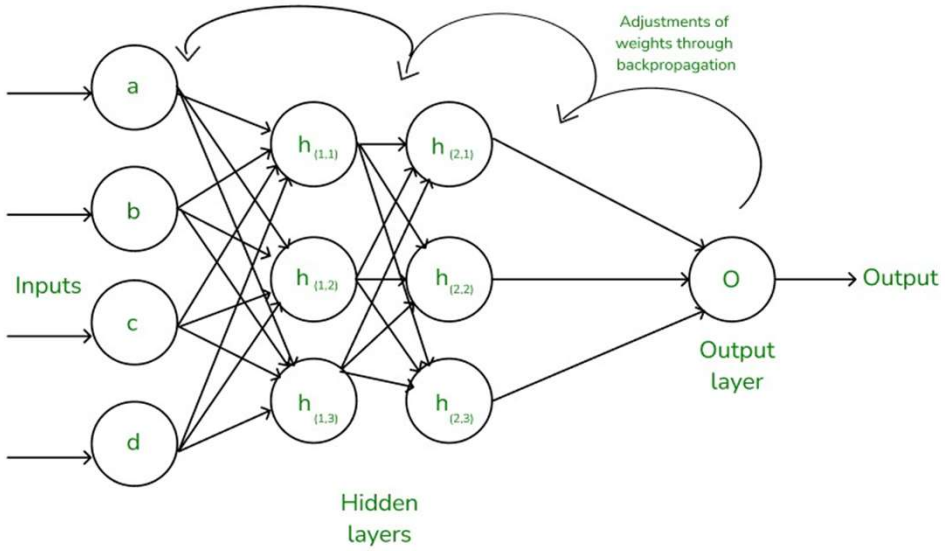


Image source: https://media.geeksforgeeks.org/wp-content/uploads/20240217152156/Frame-13.png

2. CONVOLUTIONAL NEURAL NETWORKS

Conceptual Understanding

What is CNN?

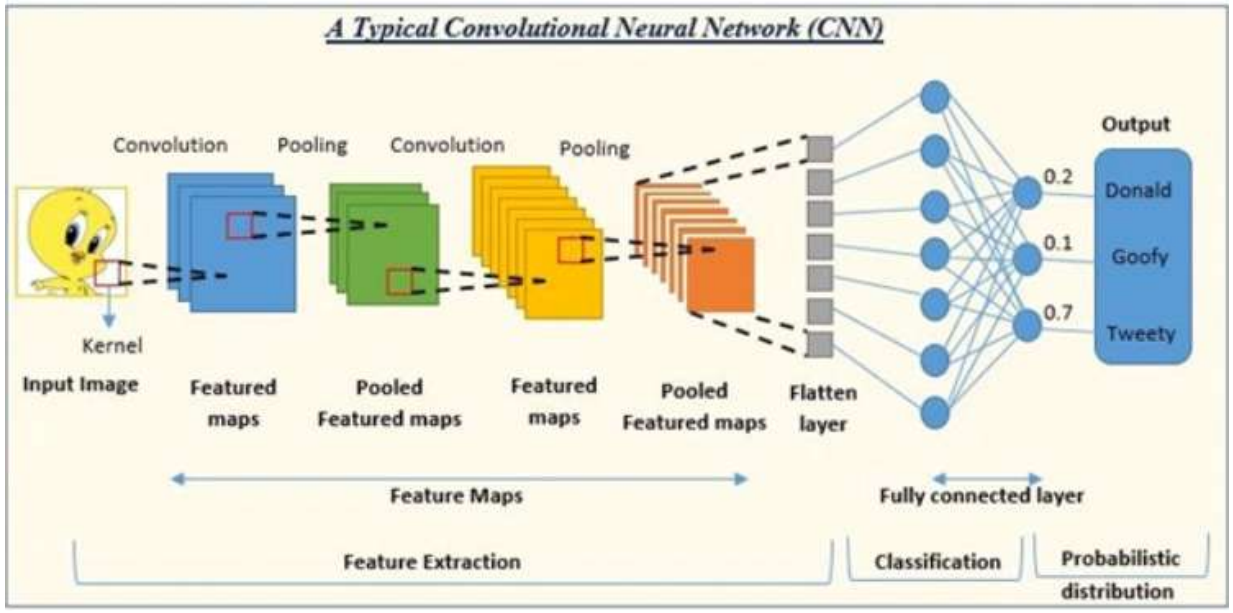


Image source:
https://backend.aionlinecourse.com/uploads/tutorials/2023/06/aionlinecourse_convolutional_neural_networks.png

LAYERS OF CNN

- Input Layer

Generally, the input will be an image or a sequence of images.

- Convolution Layer

This layer is used to extract the features from the input dataset

- Activation Layer

This layer applies a non-linear function to the output of each neuron

- Pooling Operations

This layer is used to reduce spatial dimensions of feature maps while retaining important information

- Flattening

Resulting feature maps are flattened into a one-dimensional vector.

- Fully Connected Layer (Dense Layer)

The flattened feature map is passed through one or more fully connected layers, where every neuron is connected to every neuron in the previous layer

- Output Layer

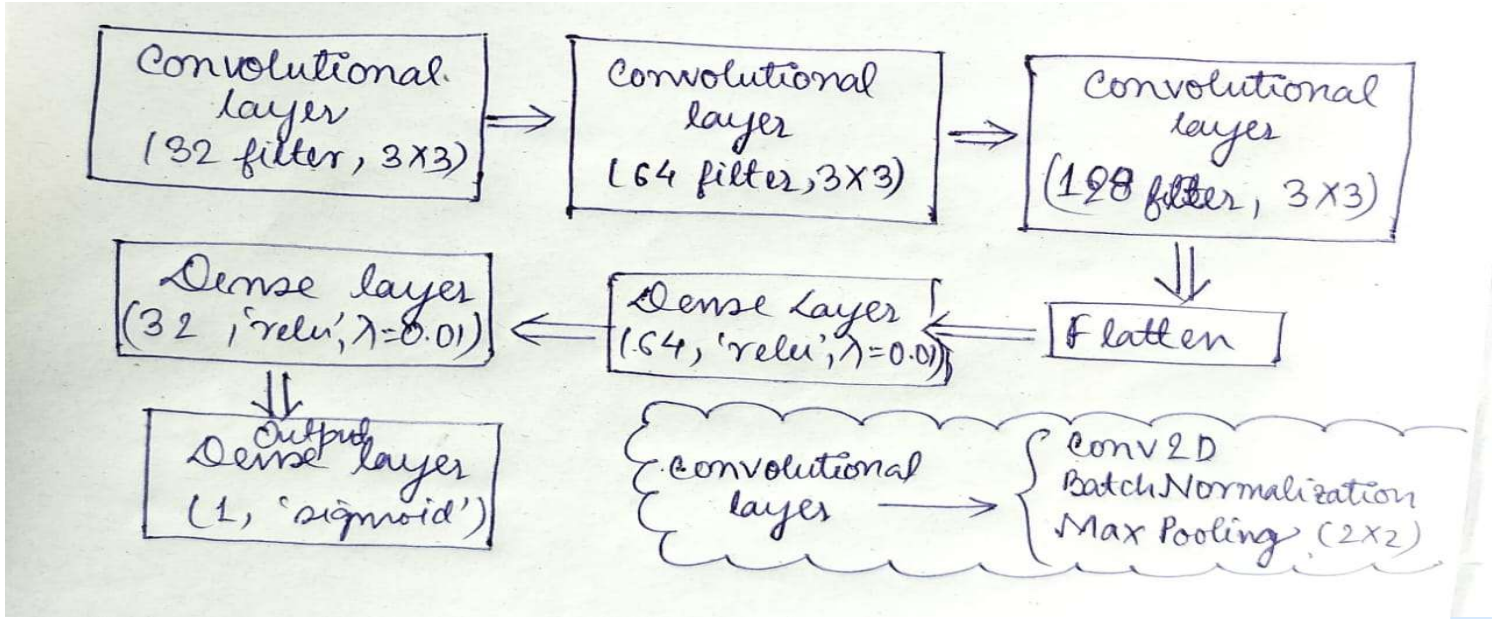
The output layer gives the decision.

3. BINARY PET CLASSIFIER CNN MODEL

CAT-DOG CLASSIFIER

Introduction

- The model aims for a binary classification ,it identifies a given image as cat(0) or dog(1) image.
- MODEL STRUCTURE:

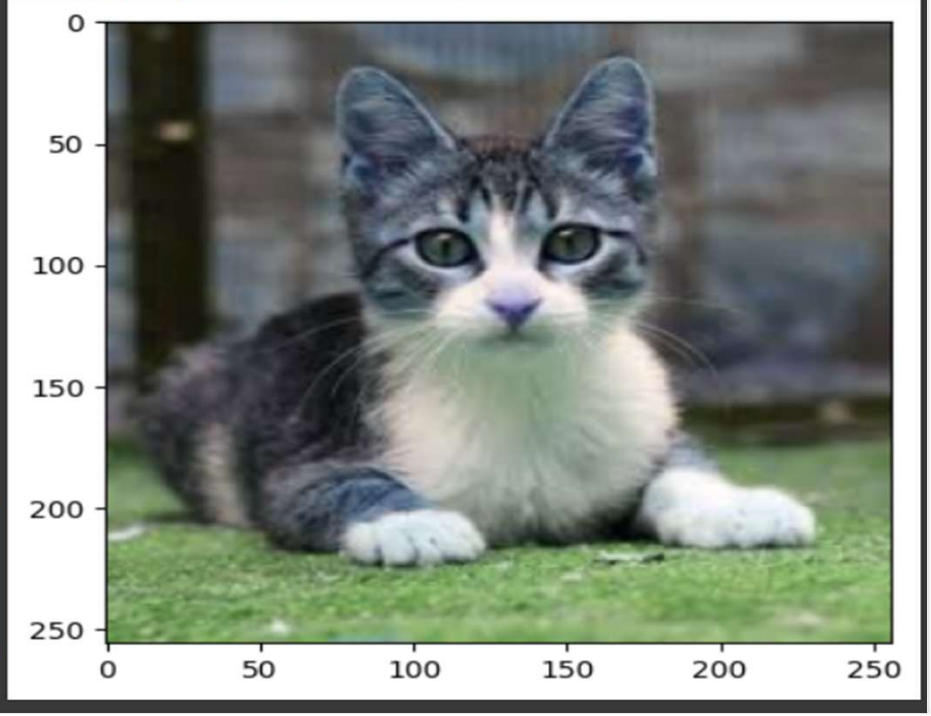


Code snippet (for model):

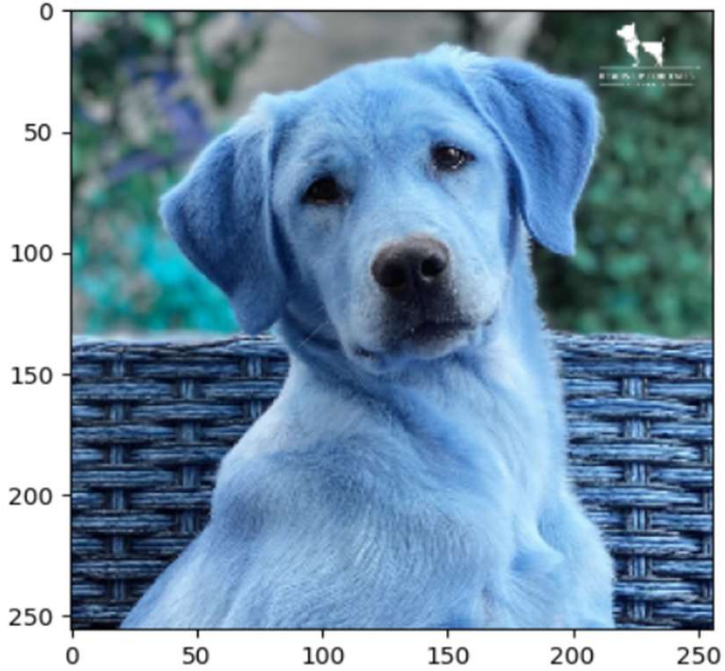
```
#create cnn 3 CONV LAYERS
model = Sequential()

model.add(Conv2D(32,kernel_size=(3,3),padding='valid',activation='relu',input_shape=(256,256,3)))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2),strides=2,padding='valid'))

model.add(Conv2D(64,kernel_size=(3,3),padding='valid',activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2),strides=2,padding='valid'))

model.add(Conv2D(128,kernel_size=(3,3),padding='valid',activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2),strides=2,padding='valid'))

model.add(Flatten())

model.add(Dense(64, activation='relu', input_dim=20,
                kernel_regularizer=l2(0.01)))  # L2 regularization with lambda=0.01

# Add a second Dense layer with L2 regularization
model.add(Dense(32, activation='relu',
                kernel_regularizer=l2(0.01)))

# Output layer
model.add(Dense(1, activation='sigmoid'))  # Binary classification example
```

RESULT

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
array([[0.]], dtype=float32)
```

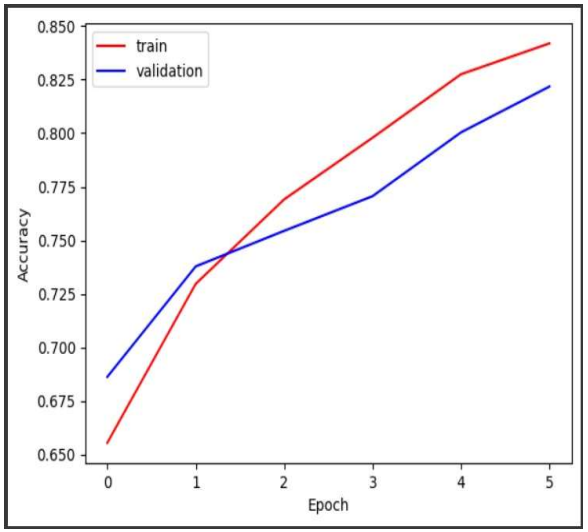



```
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 797ms/step
array([[1.]], dtype=float32)
```

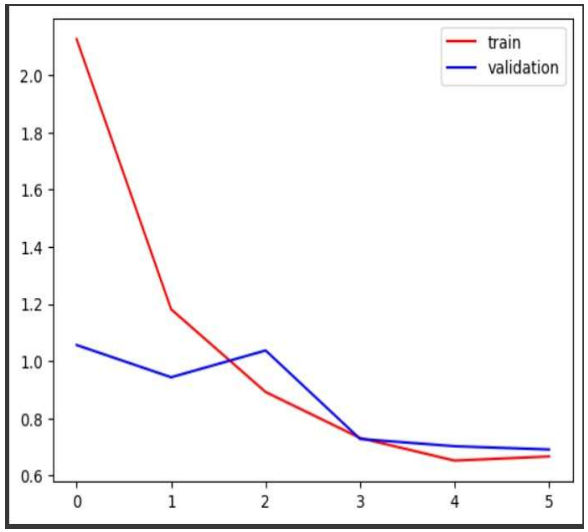


Data Visualization

- The graph between the train and test data over several epochs are noted below for this model .
- The graphs depict accuracy (left) and loss(right) VS epochs graphs respectively.



ACCURACY VS EPOCHS



LOSS VS EPOCHS

- Inference:

- Accuracy increases and loss decreases for validation data as the model is trained over greater epochs.
- Performance is refined and utilized in the ultimate model.

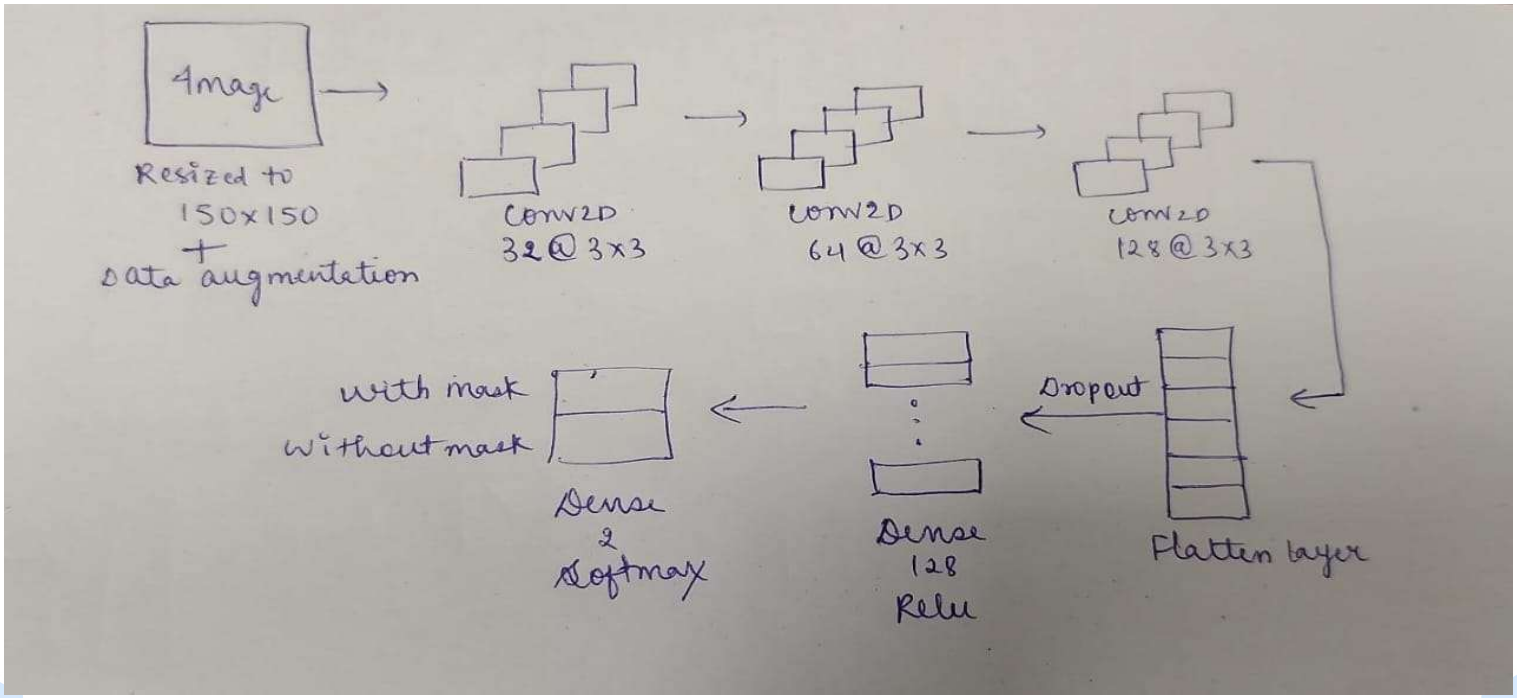
4. FACE MASK DETECTION CNN MODEL

Detects whether image of a person is masked or unmasked

Introduction

- The model aims to classify whether given image of any person is masked or unmasked .

MODEL STRUCTURE:



MODEL STRUCTURE

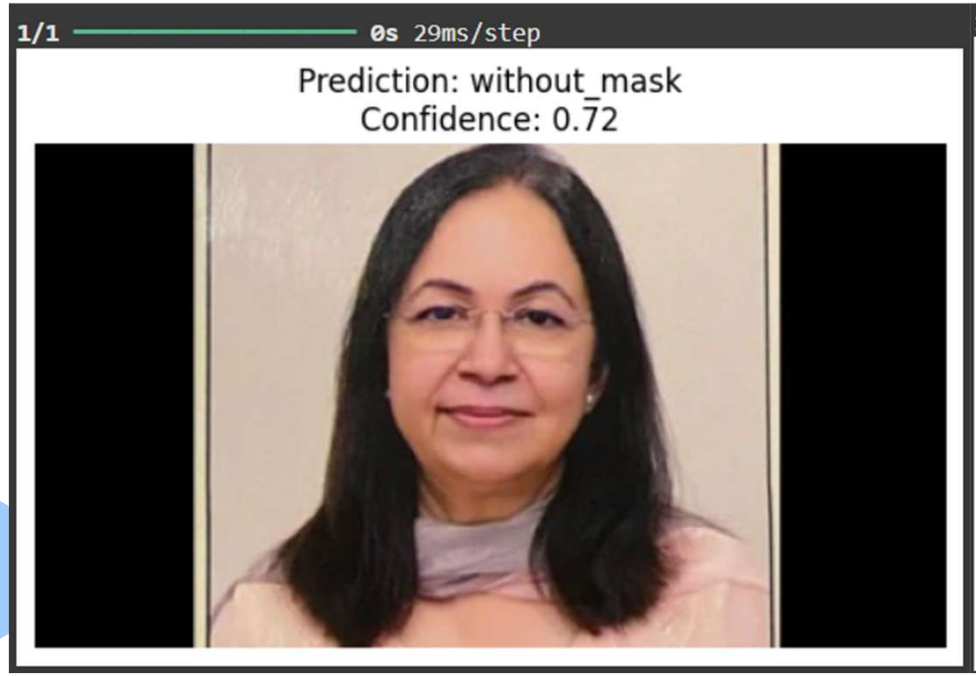
Code Snippet :

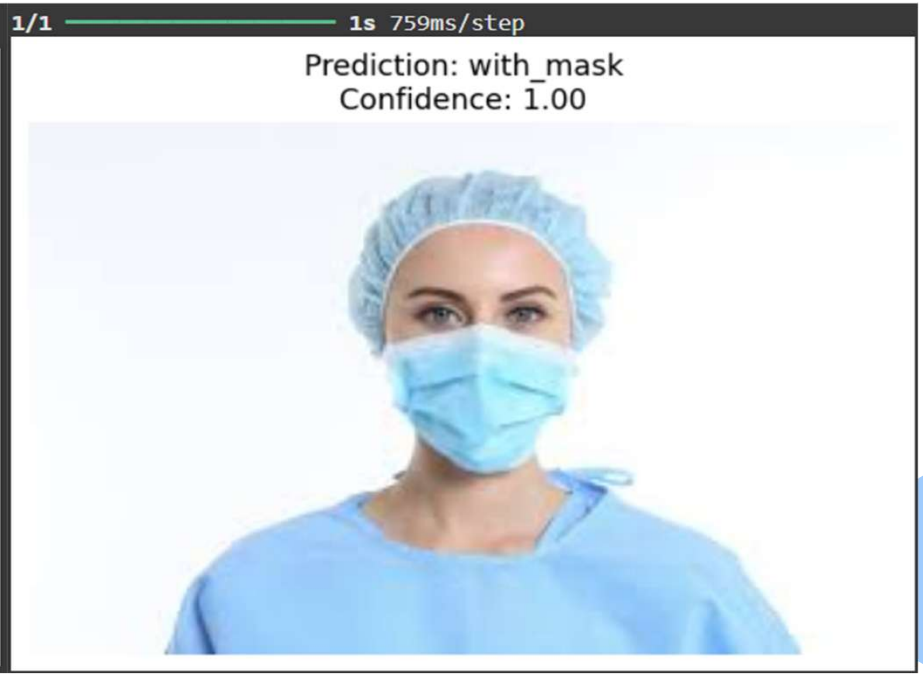
```
model = Sequential()

model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(img_size, img_size, 3)))
model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))

model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(2, activation='softmax'))
```

Model Summary:

Layer (type)	Output Shape
conv2d (Conv2D)	(None, 148, 148, 32)
max_pooling2d (MaxPooling2D)	(None, 74, 74, 32)
conv2d_1 (Conv2D)	(None, 72, 72, 64)
max_pooling2d_1 (MaxPooling2D)	(None, 36, 36, 64)
conv2d_2 (Conv2D)	(None, 34, 34, 128)
max_pooling2d_2 (MaxPooling2D)	(None, 17, 17, 128)
flatten (Flatten)	(None, 36992)
dense (Dense)	(None, 128)
dropout (Dropout)	(None, 128)
dense_1 (Dense)	(None, 2)

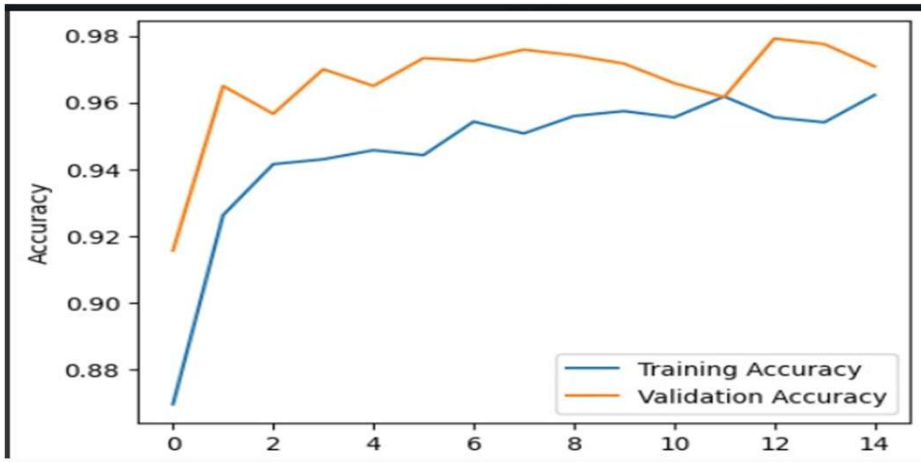
Face Mask Detection CNN Model Results :



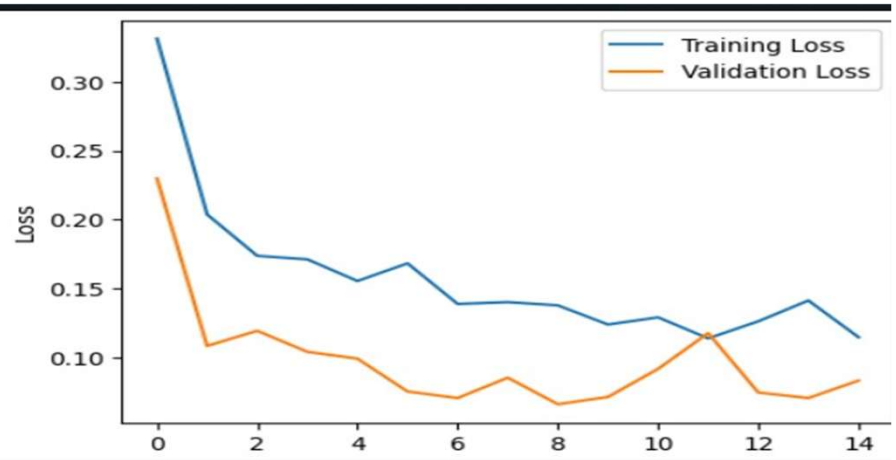


Data Visualization

- The graph between the train and test data over several epochs are noted below for this model .



ACCURACY VS EPOCHS



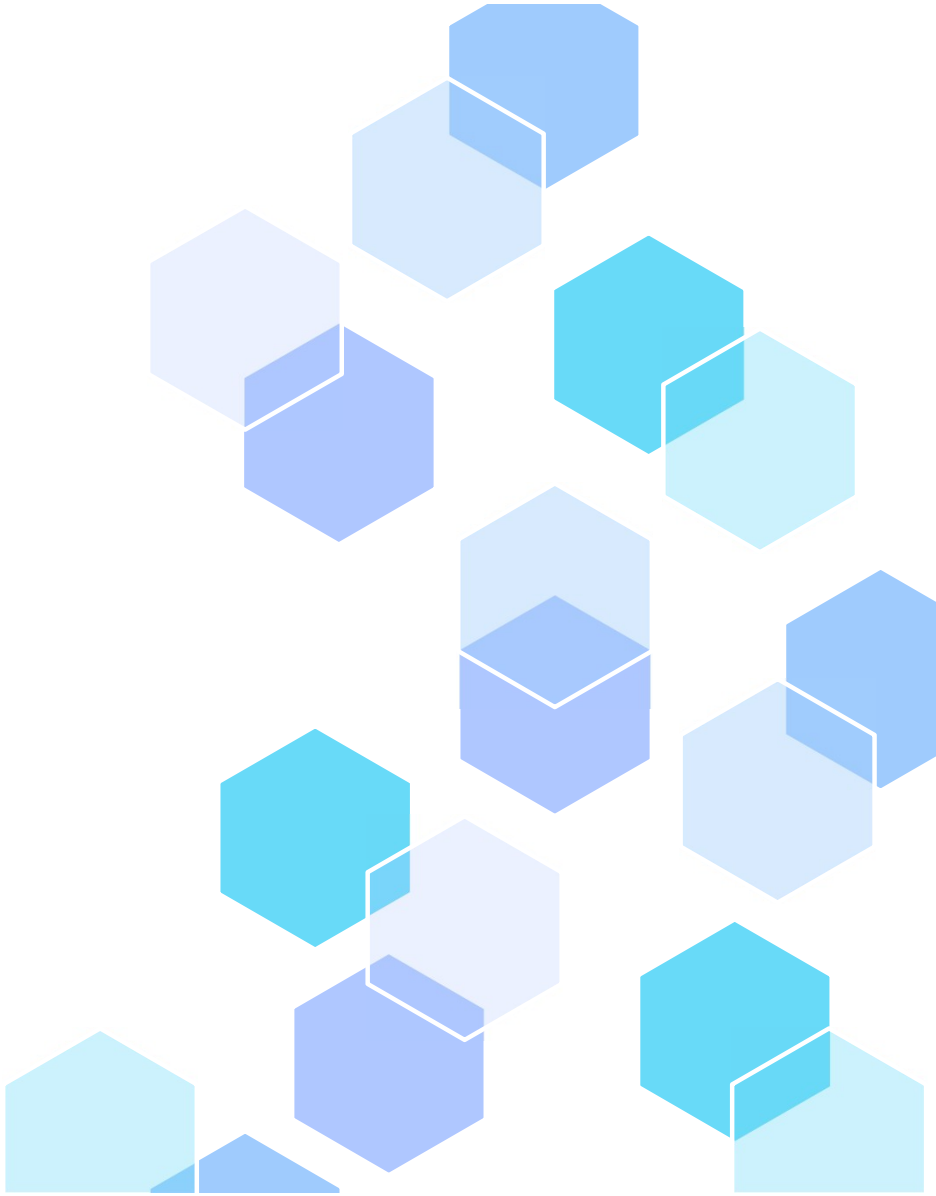
LOSS VS EPOCHS

- Inference:
  - Accuracy increases and loss decreases for validation data as the model is trained over greater epochs.
  - Prediction of masked and unmasked images is close to accurate.

05

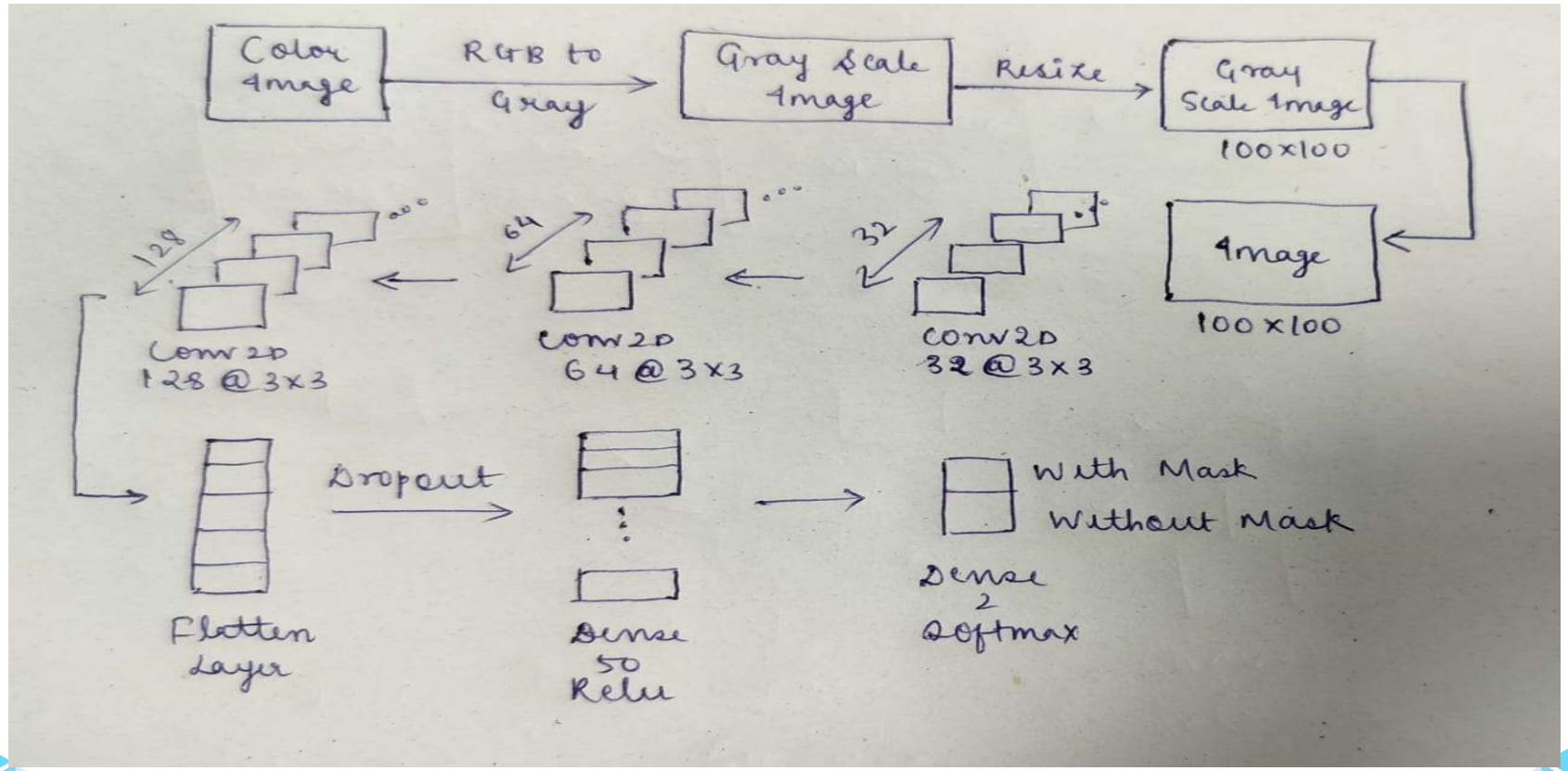
REAL-TIME FACE MASK DETECTION SYSTEM

Real-Time Multi-User Detection and Localization of Masked and Unmasked Faces.



Introduction

The model integrates camera, detected video via OpenCV , passes through Haar-cascade classifier and use CNN model to detect and localize masked and unmasked faces in real –time.



Data preprocessing code snippet:

[1]:
```
import cv2,os

data_path='dataset'
categories=os.listdir(data_path)
labels=[i for i in range(len(categories))]

label_dict=dict(zip(categories,labels)) #empty dictionary

print(label_dict)
print(categories)
print(labels)
```

```
{'with mask': 0, 'without mask': 1}
['with mask', 'without mask']
[0, 1]
```

[2]:
```
img_size=100
data=[]
target=[]


for category in categories:
    folder_path=os.path.join(data_path,category)
    img_names=os.listdir(folder_path)

    for img_name in img_names:
        img_path=os.path.join(folder_path,img_name)
        img=cv2.imread(img_path)

        try:
            gray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
            #Coverting the image into gray scale
            resized=cv2.resize(gray,(img_size,img_size))
            #resizing the gray scale into 50x50, since we need a fixed common size for all the images in the dataset
            data.append(resized)
            target.append(label_dict[category])
            #appending the image and the label(categorized) into the list (dataset)

        except Exception as e:
            print('Exception:',e)
            #if any exception rasied, the exception will be printed here. And pass to the next image
```

[3]:
```
import numpy as np

data = np.array(data) / 255.0
data = np.reshape(data, (data.shape[0], img_size, img_size, 1))
target = np.array(target)

# Use TensorFlow's keras utility for 'to_categorical'
from tensorflow.keras.utils import to_categorical

new_target = to_categorical(target)
```

[4]:
```
np.save('data',data)
np.save('target',new_target)
```

[ ]:
```

```

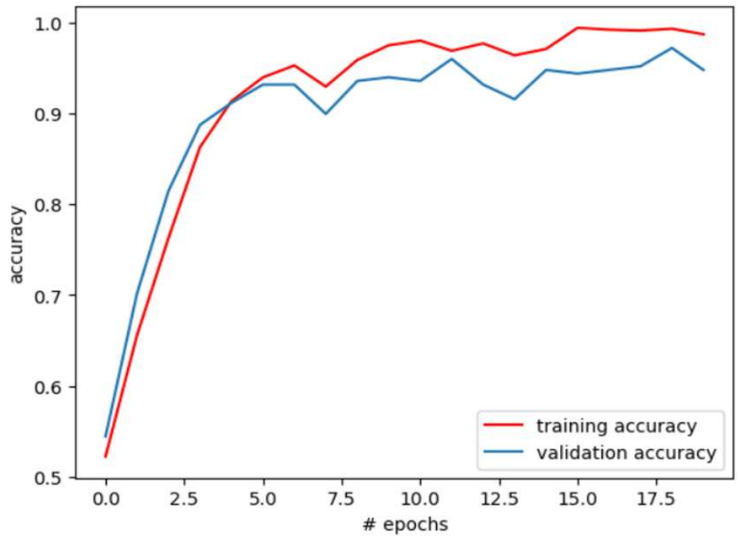
CNN MODEL STRUCTURE

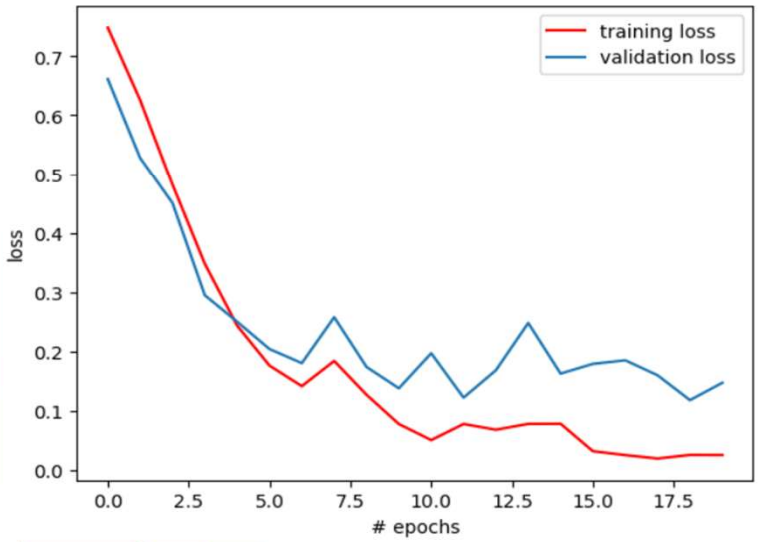
Code Snippet :

```
# Add Input layer
model.add(Input(shape=data.shape[1:]))

# First CNN layer
model.add(Conv2D(32, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

# Second CNN layer
model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

# Third CNN layer
model.add(Conv2D(128, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

# Fully connected layers
model.add(Flatten())
model.add(Dropout(0.5))
model.add(Dense(50, activation='relu'))
model.add(Dense(2, activation='softmax'))
```

Model Summary:

Layer (type)	Output Shape
conv2d (Conv2D)	(None, 98, 98, 32)
activation (Activation)	(None, 98, 98, 32)
max_pooling2d (MaxPooling2D)	(None, 49, 49, 32)
conv2d_1 (Conv2D)	(None, 47, 47, 64)
activation_1 (Activation)	(None, 47, 47, 64)
max_pooling2d_1 (MaxPooling2D)	(None, 23, 23, 64)
conv2d_2 (Conv2D)	(None, 21, 21, 128)
activation_2 (Activation)	(None, 21, 21, 128)
max_pooling2d_2 (MaxPooling2D)	(None, 10, 10, 128)
flatten (Flatten)	(None, 12800)
dropout (Dropout)	(None, 12800)
dense (Dense)	(None, 50)
dense_1 (Dense)	(None, 2)





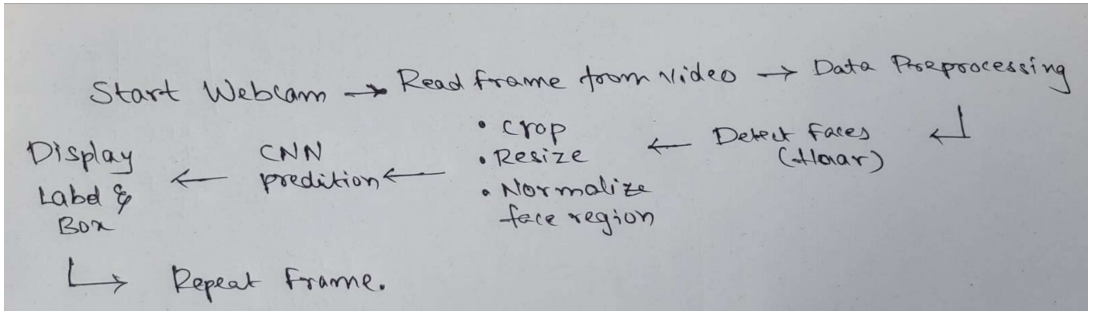
Mask Detection code snippet:

```
from keras.models import load_model
import cv2
import numpy as np
```

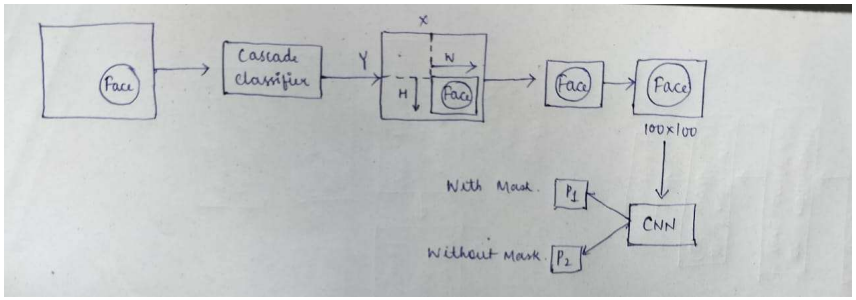
```
model = load_model('model-020.keras')

face_clsfr=cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

source=cv2.VideoCapture(0)

labels_dict={0:'MASK',1:'NO MASK'}
color_dict={0:(0,255,0),1:(0,0,255)}
```

```
while(True):

    ret,img=source.read()
    gray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    faces=face_clsfr.detectMultiScale(gray,1.3,5)

    for (x,y,w,h) in faces:

        face_img=gray[y:y+h,x:x+w]
        resized=cv2.resize(face_img,(100,100))
        normalized=resized/255.0
        reshaped=np.reshape(normalized,(1,100,100,1))
        result=model.predict(reshaped)

        label=np.argmax(result,axis=1)[0]

        cv2.rectangle(img,(x,y),(x+w,y+h),color_dict[label],2)
        cv2.rectangle(img,(x,y-40),(x+w,y),color_dict[label],-1)
        cv2.putText(img, labels_dict[label], (x, y-10),cv2.FONT_HERSHEY_SIMPLEX,0.8,(255,255,255),


    cv2.imshow('LIVE',img)
    key=cv2.waitKey(1)

    if(key==27):
        break

cv2.destroyAllWindows()
source.release()
```

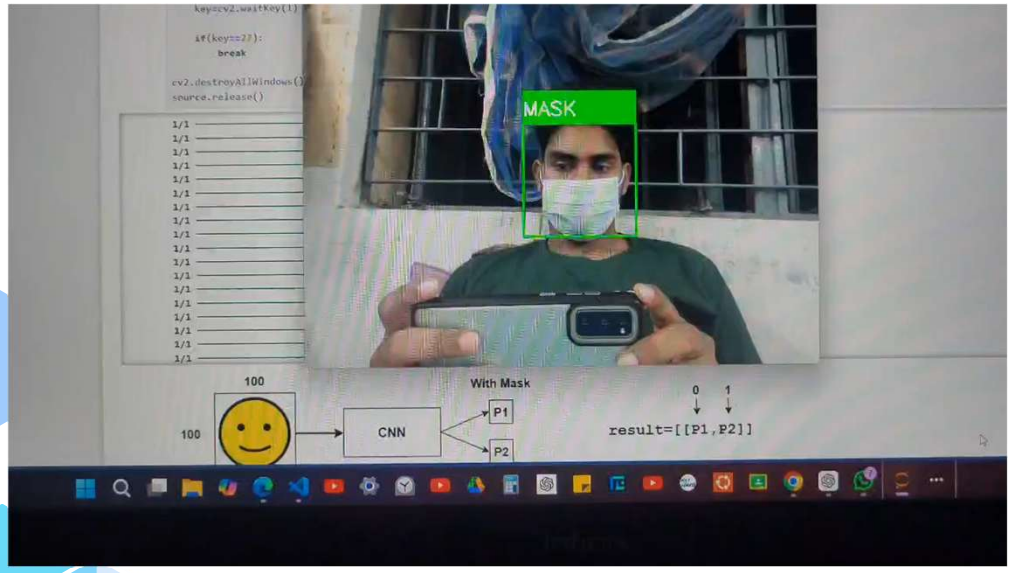
Workflow of the model:

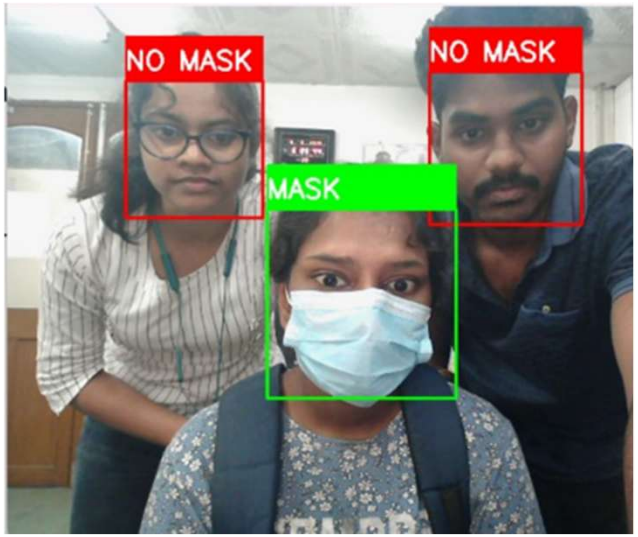


Detecting faces:



RESULT





REFERENCES

For Datasets:

- observations/experiements/data at master · prajnasb/observations · GitHub
- Find Open Datasets and Machine Learning Projects | Kaggle

Others:

- GeeksforGeeks | Your All-in-One Learning Portal
- Wikipedia
- Shodhganga : a reservoir of Indian theses @ INFLIBNET
- Google Colab
- Dataaspirant
- https://gamedevacademy.org/
- COVID-19: Face Mask Detector with OpenCV, Keras/TensorFlow, and Deep Learning - PyImageSearch

THANK YOU!