Project on Computer Vision

Yogini Sanmugarajah

Overview

What is computer vision?

Computer vision is part of Artificial Intelligence, which enables computers to mimic the human brain and teach computers to learn from the given input.

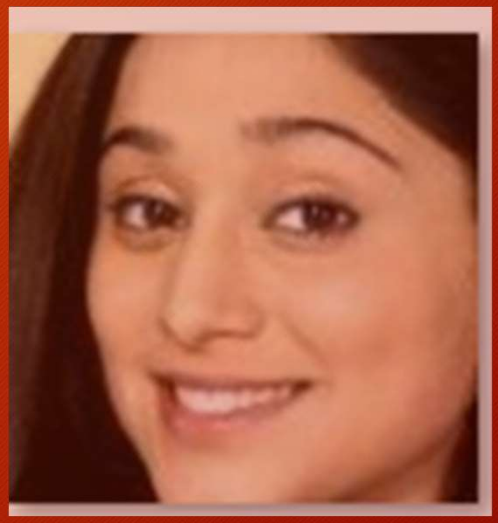
- Computer Vision(CV) has vastly improved in the current era
- It is been used in many ways across many industries around the world
- CV has made many complicated activities to be performed with ease
- For example:
  - In the medical field tumor detection, cancer detection, etc
  - In manufacturing product assembly, defect detection, etc

Objective

Today I am going to present how computer vision is used to detect a human face

with the mask

without the mask



Import Libraries

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, Activation,Dense,Flatten, BatchNormalization, MaxPool2D
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
import glob
import numpy as np
from sklearn.metrics import classification_report
```

Import the Data

- Import image using ImageDataGenerator
- Set the parameters
- Split train set and validation set

```
path ='/content/drive/MyDrive/Train'

img_datagenerator = ImageDataGenerator(validation_split= 0.2,
                                       rescale = 1./255,
                                       rotation_range = 40,
                                       width_shift_range= 0.2,
                                       height_shift_range= 0.2,
                                       zoom_range = 0.2,
                                       horizontal_flip = True,
                                       fill_mode= 'nearest')
```

```
train_generator = img_datagenerator.flow_from_directory(path,
                                                        target_size= (70,70),
                                                        batch_size =batch_size,
                                                        color_mode = 'rgb',
                                                        class_mode = 'binary',
                                                        shuffle = True,
                                                        seed = 42,
                                                        subset = 'training')
```

```
Found 8015 images belonging to 2 classes.
```

```
valid_generator = img_datagenerator.flow_from_directory(path,
                                                        target_size= (70,70),
                                                        batch_size =batch_size,
                                                        color_mode = 'rgb',
                                                        class_mode = 'binary',
                                                        shuffle = True,
                                                        seed = 42,
                                                        subset = 'validation')
```

```
Found 2003 images belonging to 2 classes.
```

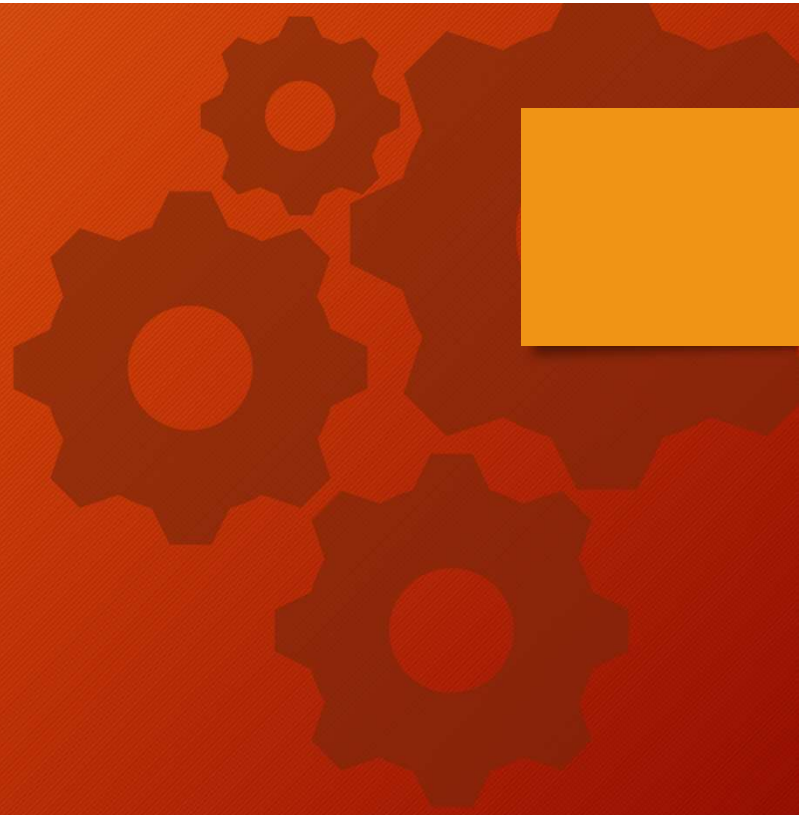
- Parameters have been set for the training dataset and validation dataset

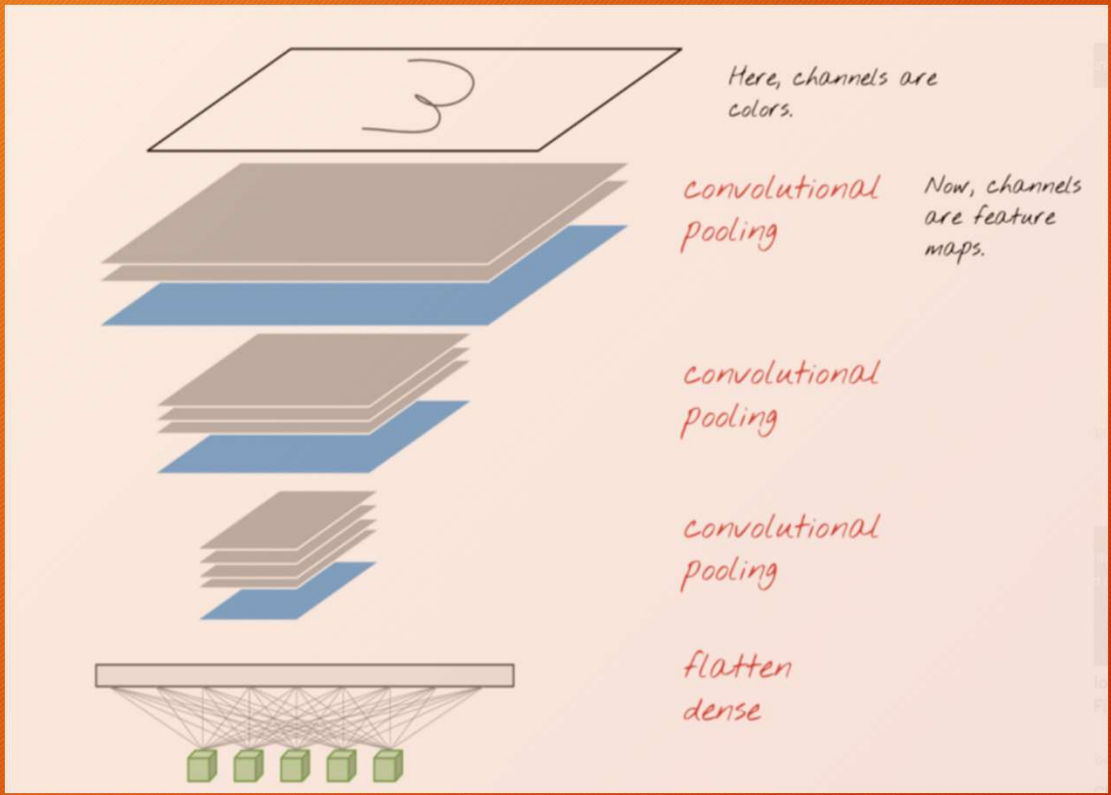
Defining Model-Sequential Neural Network

```
model = Sequential([Conv2D(filters = 32, kernel_size = (3,3), activation = 'relu', padding= 'same',input_shape=(70,70,3)),
                    MaxPool2D(pool_size = (2,2), strides=2),
                    Conv2D(filters = 32, kernel_size = (3,3), activation = 'relu', padding= 'same'),
                    MaxPool2D(pool_size = (2,2), strides=2),
                    Conv2D(filters = 64, kernel_size = (3,3), activation = 'relu', padding= 'same'),
                    MaxPool2D(pool_size = (2,2), strides=2),
                    Flatten(),
                    Dense(units=64,activation='relu'),
                    Dense(units=1,activation='sigmoid')
])
```

- Convolutional neural network models are good in image recognition.
- They have strong ability in abstracting spatial information from multiple levels
- Convolutional connections help to reduce the computational cost, number of parameters, and overfitting risk
- In SCNN multilayered neural networks stacked and hidden layers are formed on top of each other in a sequence
- This helps the convolutional neural networks to learn the hierarchical features

```
model.summary()

Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_3 (Conv2D)           (None, 70, 70, 32)        896

 max_pooling2d_3 (MaxPooling  (None, 35, 35, 32)       0
 2D)

 conv2d_4 (Conv2D)           (None, 35, 35, 32)        9248

 max_pooling2d_4 (MaxPooling  (None, 17, 17, 32)       0
 2D)

 conv2d_5 (Conv2D)           (None, 17, 17, 64)        18496

 max_pooling2d_5 (MaxPooling  (None, 8, 8, 64)         0
 2D)

 flatten_1 (Flatten)         (None, 4096)              0

 dense_2 (Dense)             (None, 64)                262208

 dense_3 (Dense)             (None, 1)                 65

=================================================================
Total params: 290,913
Trainable params: 290,913
Non-trainable params: 0
_________________________________________________________________
```





- Conv2d identifies the feature of the image by looking at the color channels, dark edges, shapes, and textures
- Maxpooling reduces the spatial dimension, which means reducing the height and width of the image by reducing the pixels in the image
- Flatten will unstack the output and make it into an array
- Dense layer is a layer of neurons where each neuron receives input from all the neurons from the previous layer

Source: https://towardsdatascience.com/a-beginners-guide-to-convolutional-neural-networks-cnns-14649dbddce8

```
model.compile(optimizer = Adam(learning_rate = 0.0001),loss = 'binary_crossentropy', metrics = ['accuracy'])
```

- Optimizer helps to optimize the input weights
- Loss function enables the model to find errors or deviations in the learning process
- Metrics helps to evaluate the model performance

Fit the Model

```
history = model.fit(train_generator, epochs = epoch, validation_data = valid_generator,batch_size = batch_size )

Epoch 1/30
1002/1002 [==============================] - 1367s 1s/step - loss: 0.3117 - accuracy: 0.8710 - val_loss: 0.2192 - val_accuracy: 0.9146
Epoch 2/30
1002/1002 [==============================] - 95s 95ms/step - loss: 0.1975 - accuracy: 0.9258 - val_loss: 0.1818 - val_accuracy: 0.9261
Epoch 3/30
1002/1002 [==============================] - 94s 94ms/step - loss: 0.1747 - accuracy: 0.9352 - val_loss: 0.1622 - val_accuracy: 0.9371
Epoch 4/30
1002/1002 [==============================] - 97s 97ms/step - loss: 0.1701 - accuracy: 0.9361 - val_loss: 0.1480 - val_accuracy: 0.9481
Epoch 5/30
1002/1002 [==============================] - 94s 94ms/step - loss: 0.1588 - accuracy: 0.9407 - val_loss: 0.1321 - val_accuracy: 0.9506
Epoch 6/30
1002/1002 [==============================] - 96s 95ms/step - loss: 0.1481 - accuracy: 0.9475 - val_loss: 0.1279 - val_accuracy: 0.9536
Epoch 7/30
1002/1002 [==============================] - 94s 94ms/step - loss: 0.1480 - accuracy: 0.9431 - val_loss: 0.1135 - val_accuracy: 0.9571
Epoch 8/30
```

The accuracy of trainset and validation set are 97 percent

```
Epoch 19/30
1002/1002 [==============================] - 95s 95ms/step - loss: 0.0970 - accuracy: 0.9659 - val_loss: 0.0761 - val_accuracy: 0.9725
Epoch 20/30
1002/1002 [==============================] - 96s 96ms/step - loss: 0.0913 - accuracy: 0.9686 - val_loss: 0.0843 - val_accuracy: 0.9666
Epoch 21/30
1002/1002 [==============================] - 96s 96ms/step - loss: 0.0950 - accuracy: 0.9657 - val_loss: 0.0676 - val_accuracy: 0.9760
Epoch 22/30
1002/1002 [==============================] - 97s 96ms/step - loss: 0.0885 - accuracy: 0.9709 - val_loss: 0.1286 - val_accuracy: 0.9521
Epoch 23/30
1002/1002 [==============================] - 97s 97ms/step - loss: 0.0888 - accuracy: 0.9708 - val_loss: 0.0661 - val_accuracy: 0.9770
Epoch 24/30
1002/1002 [==============================] - 97s 96ms/step - loss: 0.0841 - accuracy: 0.9702 - val_loss: 0.0838 - val_accuracy: 0.9675
Epoch 25/30
1002/1002 [==============================] - 98s 98ms/step - loss: 0.0827 - accuracy: 0.9713 - val_loss: 0.0696 - val_accuracy: 0.9725
Epoch 26/30
1002/1002 [==============================] - 97s 96ms/step - loss: 0.0816 - accuracy: 0.9717 - val_loss: 0.1027 - val_accuracy: 0.9626
Epoch 27/30
1002/1002 [==============================] - 96s 96ms/step - loss: 0.0738 - accuracy: 0.9731 - val_loss: 0.0876 - val_accuracy: 0.9710
Epoch 28/30
1002/1002 [==============================] - 94s 94ms/step - loss: 0.0719 - accuracy: 0.9748 - val_loss: 0.0869 - val_accuracy: 0.9675
Epoch 29/30
1002/1002 [==============================] - 95s 94ms/step - loss: 0.0706 - accuracy: 0.9757 - val_loss: 0.0558 - val_accuracy: 0.9815
Epoch 30/30
1002/1002 [==============================] - 95s 94ms/step - loss: 0.0711 - accuracy: 0.9744 - val_loss: 0.0816 - val_accuracy: 0.9700
```

```
loss_train = history.history['loss']
loss_val = history.history['val_loss']
epocplot = range(1, epoch+1)
plt.plot(epocplot,loss_train,'g',label = 'Traning loss')
plt.plot(epocplot,loss_val,'b',label = 'Validation loss')
plt.title('Training vs valid loss')

plt.show()
```

The loss closer to zero better the model performance

Test the Model

```
from IPython.display import Image,display
TGREEN = '\033[1;37;42m'
TRED = '\033[1;37;41m'

for i in vari_data:
  img_directory = i
  img_data = image.load_img(img_directory, target_size= (70,70))
  img_data = image.img_to_array(img_data)
  img_data = np.expand_dims(img_data, axis = 0)

  classify = model.predict(img_data)
  display(Image(img_directory,width = 150,height = 150))

  print("\n")

  if(int(classify[0][0]) == 0):
    print(TGREEN + 'MASK ON. \n')
  else:
    print(TRED + 'MASK OFF. \n')
```

MASK ON.

MASK ON.

MASK ON.

MASK OFF.

MASK OFF.

MASK OFF.

The model is able to detect the faces with mask and without mask

MASK ON.

MASK ON.

MASK ON.

MASK ON.

MASK OFF.

MASK ON.

Some of the wrong detections

Real Life Applications

- During outbreak of respiratory diseases- monitor and control in malls and other enclosed public places
- Use in hospitals to monitor the staff entering the operation theatre
- Care center staff dealing with patients who have low immunity

Dataset: https://www.kaggle.com/datasets/ashishjangra27/face-mask-12k-images-dataset

Github : https://github.com/y2131

THANK YOU