1 INTRODUCTION

Object detection is a computer vision approach for detecting and locating objects in images and video. Object detection can detect instances of visual objects of specific classes such as persons, animals, carsor building in digital pictures like photos or video frames. When an image is shown to the eyes, the brain immediately recognizes the objects. On the other hand, a machine needs long time and lot of trainingdata to recognize the images. However, due to the development of recent technology in machine learn-ing, deep learning and the field of computer vision has become a lot easier and more intuitive. Objectdetection technology has exploded in popularity across a wide range of sectors. It enables self-drivingcars in safely navigating traffic, detecting violent behaviour in crowded areas, monitoring object through video surveillance, recognising face mask through object detection module, ensuring adequate qualitycontrol of pars in production, among many other things. And this is only the tip of the iceberg in termsof what object detection technology can achieve! In modern video surveillance systems, detecting facein video streams is critical task. Deep learning algorithms have recently been developed that deliver face identifying results.

Artificial intelligence enables image and video-based detection algorithms that can accurately detect an object and determine whether the human is wearing or not wearing a mask. Face mask identification can be done with diverse datasets utilizing deep learning and machine learning approaches such as supportvector machines and decision trees. The main purpose of this thesis is to develop a Face mask detection model. The model was trained using the MobileNetV2 architecture. Using two separate image datasets, the model was trained and tested. A pretrained model of the MobileNetV2 architecture was used to distinguish faces from video streams. In addition to the OpenCV framework, a variety of packages ofmachine learning, deep learning approaches and image processing techniques were used. To provideeffective monitoring and enable proactively, the following processes will be used: data augmentation, loading the classifier, establishing the fully connected layer, pre-processing, and loading the picture data, applying the classifier, training phase, validation and testing phase.

2 DEEP LEARNING

Deep learning is a subfield of machine learning in artificial intelligence that works with algorithms that are inspired by the biological structure and function of the brain to help computers with intelligence. A computer model learns to execute categorization tasks directly from images, text, or sound using deeplearning. Deep learning models can attain cutting-edge accuracy, sometimes even surpassing human performance. Models are trained utilizing a vast quantity of labeled data and multi-layer neural network architectures. Data science, which covers statistics and predictive modelling, contains deep learning as a component. Deep learning improves the process of gathering, analysing, and interpreting massive amounts of data much faster and easier for data scientists. Deep learning is a vital component of driver-less cars, allowing them to discriminate between a stop sign and a lamppost. Voice control in consumer products such as phones, tablets, tvs, and hands-free speakers is enabled by this feature.

2.1 Artificial neural networks and biological brains

Artificial neural networks have been around since the 1940s, and deep learning emerges from them. Artificial neurons from neural networks, which are interconnected networks of processing units that resemble axons in a biological brain. Dendrites receive input signals from multiple nearby neurons in abiological neuron, which can number in the thousands. These changed signals are subsequently sent to the neuron's cell body, or soma, where they are combined and sent to the axon. The axon will send a signal to the adjacent dendrites of other neurons if the received input signal exceeds a certain threshold.For comparison, Figure 1 shows the structure of a biological neuron. (Pattanayak 2017.)

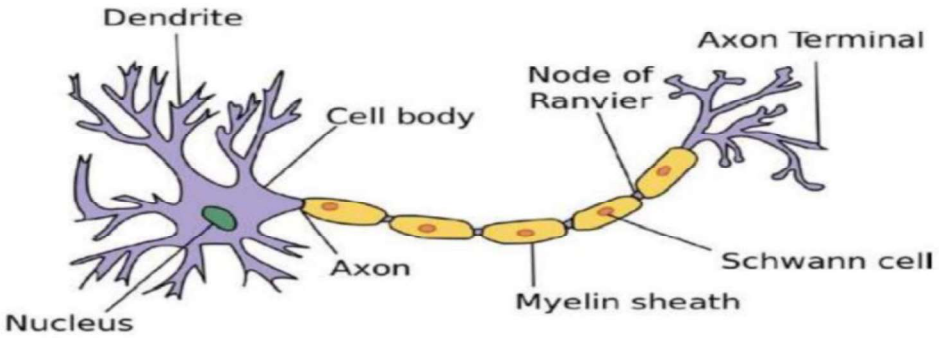


FIGURE 1. Structure of a biological neuron (Pro Deep Learning with TensorFlow 2017.)

Artificial neuron units are based on biological neurons, with a few tweaks for ease of use. The input connection to the neuron, like the dendrites, transmit attenuated or amplified input impulses from neigh- bouring neurons. The signals are transmitted on to the neuron, which sums up the incoming signals before deciding what to output based on the overall input received. When the entire input crosses a pre- defined threshold, for example, a binary threshold neuron produces an output value of 1; otherwise, the output remains at 0. Artificial neural networks use a variety of different types of neurons, with the acti-vation function on the entire input to produce the neuron output being the only difference. For simpleanalogy and comprehension, the different biological analogues are marked in the artificial neuron in Figure 2. (Pattanayak 2017.)

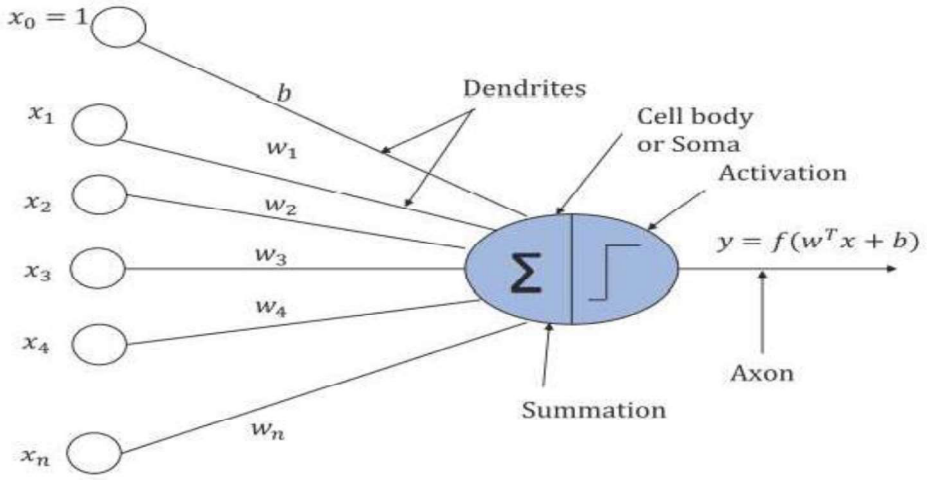


FIGURE 2. Structure of an artificial neuron (Pro Deep Learning with TensorFlow 2017.)

2.2 The process of deep learning

Deep learning models are often referred to as deep neural networks because most deep learning ap- proaches use neural network designs. The number of hidden layers in a neural network is commonly referred to as "deep". Deep learning refers to the numerous layers that a neural network collects over time, with performance improving as the network becomes more complex. Each layer of the networkprocesses the data it receives in a unique way, which then informs the following layer. Deep neural networks can have up to 150 hidden layers, whereas traditional neural networks only have 2-3. (Math-Works, 2). Deep learning models are taught utilizing enormous quantities of labeled data and neural

network architectures that learn features directly from the data instead of requiring manual feature ex-traction. The structure of a neural network is shown in Figure 3.

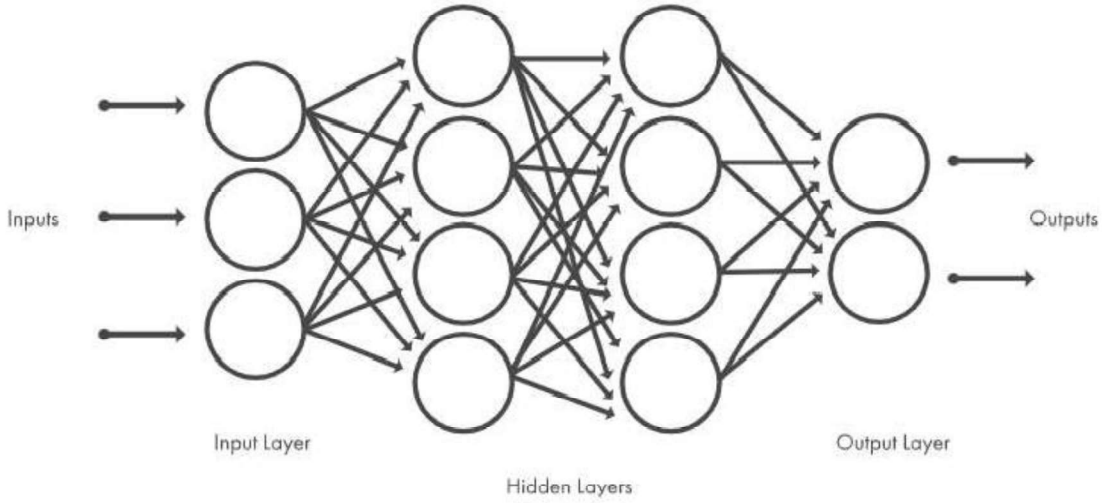


FIGURE 3. Layered neural network are made up of a series of interconnected nodes (MathWorks2022.)

2.3 Deep learning methods

To create powerful deep learning models, a variety of approaches can be used. The loss of learning rateis one of them. The learning rate is a hyperparameter that determines how much the model changes when the weights are updated in response to the expected error. It is a factor that defines the system or providesoperational conditions prior to the learning process. An unstable training process or the acquisition of a faulty set of weights might arise when learning rats are overly high. A slow learning rate could result ina lengthy and ineffective training process. Another strategy is transfer learning, which requires gaining access to a network's internals and fine-tuning a previously trained model. To begin, users add fresh data to an established network that already contains previously unknown classifications. After the net-work has been updated, new tasks can be completed with more specific classification skills. This method has the advantage of requiring significantly less data that other methods, reducing computation time tominutes or hours. (Burns 2021.)

applications and those with a high variety of output types. However, because it requires a big amount of data and might take days or weeks to train, it is a less used technique in general. The dropout methodhas been shown to increase neuronal performance on supervised learning tasks in domains including asspeech recognition, document categorizations, and computation-al biology. By randomly deleting units and their connections from the neural network during training, this method seeks to alleviate the problem of overfitting in networks with many parameters. (Burns 2021.)

2.4 Examples of deep learning at work

Deep learning applications can be found in a wide range of fields. Automotive experts are employing deep learning to detect elements such as stop signs and traffic signals automatically. In addition, deeplearning is used to recognize pedestrians, which helps to reduce accidents. It is used in the aerospace and defence industry to distinguish items from satellites that locate regions of interest and determine whether troops are in safe or dangerous zones. Cancer researchers are using it to detect cancer cells automatically in the medical field. Researchers improved a microscope that generates high-dimensional data that canbe used to train a deep learning system to consistently identify cancer cells. By automatically recognizing if people or objects are within a harmful distance of heavy machinery, this technology aids in increasing worker safety near heavy machinery. Deep learning is also used in automatic hearing and voice transla-tion. (MathWorks 2022).

3 CONVOLUTIONAL NEURAL NETWORK (CNN)

Artificial neural networks are becoming increasingly used for processing unstructured data, such as im- ages, text, audio, and speech. For such unstructured input, convolutional neural networks (CNNs) oper- ate well. Convolutional neural networks find essential features from data when there is a topology con-nected with it. CNNs are inspired by multi- layer perceptron's in terms of architecture. CNN takes use of local spatial correlation by imposing local connection limitations between neurons in adjacent layers. The processing of data via the convolution operation is the heart of convolutional neural networks. When any signal is convolution with another signal, a third signal is produced that may provide more infor-mation about the signal than the original signal. (Pattanayak 2017.)

3.1 CNN architectures

CNN architectures occur in a variety of shapes and sizes, but they all have convolutional and pooling(or subsampling) layers that are organised into modules. These modules are followed by one or more fully connected layers, like a normal feedforward neural network. To create a deep model, modules are frequently stacked on top of one another. The network receives an image directly, which is then pro-cessed through various rounds of convolution and pooling. The results of these processes are then fed into one or more fully connected layers. The output layer receives the inputs from the layers above it, executes the calculations using its neurons, and then computes the output. Even though this is the most used base architecture in the literature, various architecture modifications have been proposed in recent years with the goal of boosting image classification accuracy or lowering computation costs. The rela-tionship from input layer to output layer is shown in Figure 4. (Geron 2017.)

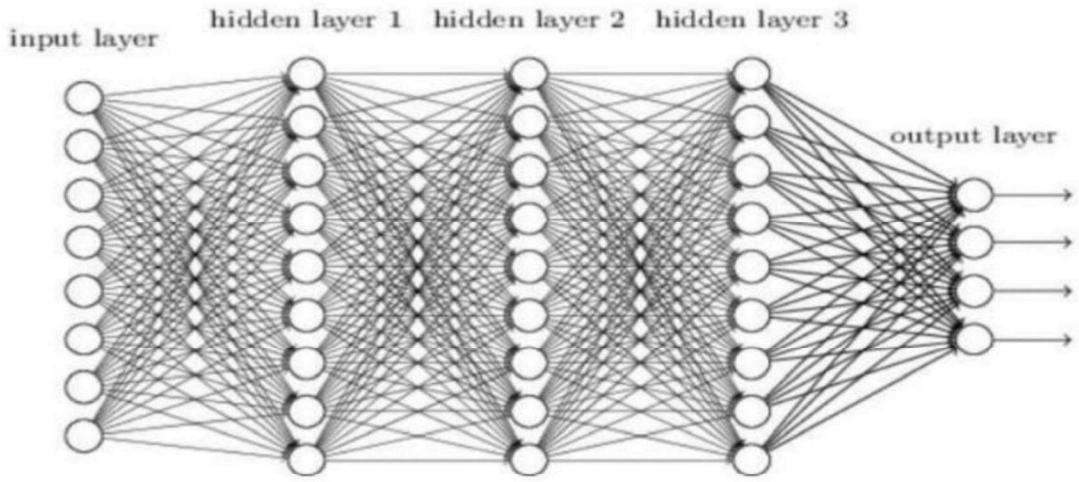


FIGURE 4. CNN architecture (Hands-On Machine Learning with Scikit-Learn and TensorFlow 2017.)

3.2 Workflow of CNN

The higher performance of convolutional neural networks with picture, speech, or audio

layer are the main three basic types of layers available. A convolutional network's first layer is the signals inputsdistinguishes them from other neural networks. Convolutional layer, pooling layer, and fully connected convolutional layer. While further convolutional layers or pooling layers can be added after convolu-tional layers, the fully connected layer is the last layer. The CNN becomes more complicated with eachlayer, detecting larger areas of the image. Earlier layers concentrate on basic elements like colors andborders. As the visual data travels through the cnn layers, it begins to distinguish larger elements or features of the item, eventually identifying the target object. (IBM Cloud Education 2020.)

3.2.1 Convolutional layer

The convolutional layer is the most important component of a CNN because it is where most of the computation takes place. It requires input data, a filter, and a feature map,

among other things. For ex-ample, a color image made up of a 3D matrix of pixels is sent to the input layer. This means the inputwill have three dimensions: height, width, and depth, which match to the rgb color space of a picture.

Convolution is the term for this procedure. The feature detector is a two-dimensional (2-D) weighted array that represents a portion of the image. The filter size, whichcan vary in size, is usually a 3x3 matrix, which also affects the size of the receptive field.

After that, thefilter is applied to a portion of the image, and a dot product between the input pixels and the filter thenshifts by a stride, and the procedure is repeated until the kernel has swept across the entire image. A feature map, activation map, or convolved feature is the ultimate output of a series of dot products from the input and the filter. (IBM Cloud Education 2020.)

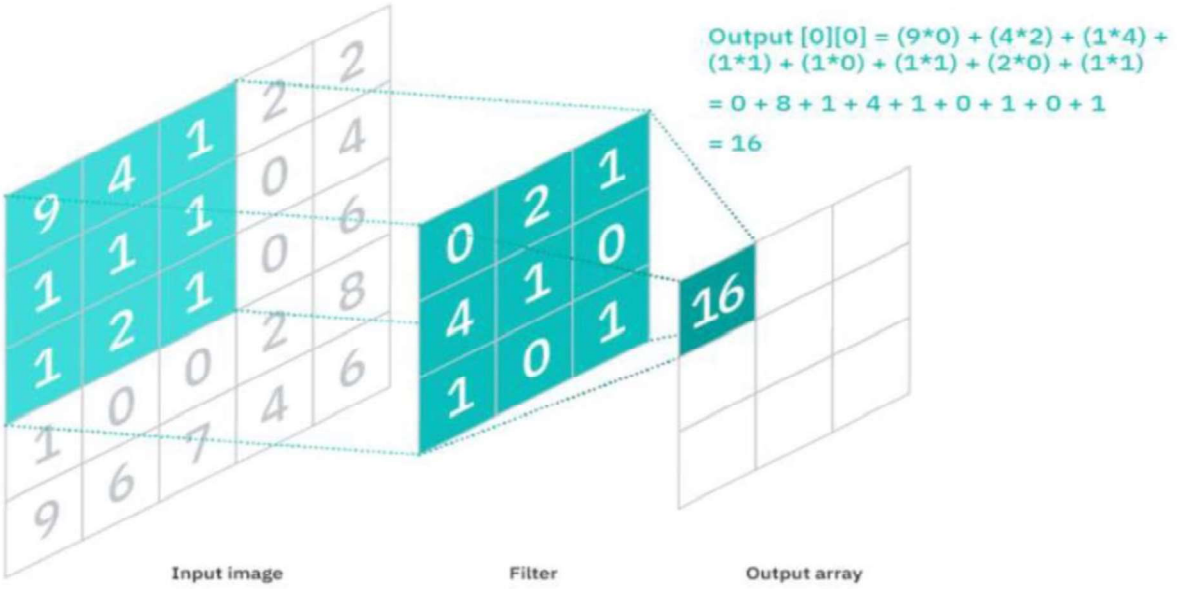


FIGURE 5. Visual representation of a convolutional layers (IBM Cloud Education 2020.)

Each output value in the feature map does not have to relate to each pixel value in the input image, asseen in the Figure 5. It simply must be connected to the receptive field, which is where the filter is applied. Convolutional and p
the output array does not have to map directly to each input value. The feature detector's weights remain
ooling layers are often referred to as "partially connected" layers because

constant as it advances over the image, a technique known as parameter sharing. Through the process of backpropagation and gradient descent, some parameters, such as weight values, adjust during training.Before the neural network can be trained, three hyperparameters that determine the output volume sizemust be established. Number of filters, stride, and zero padding are among them. The depth of the output
is influenced by the number of filters used. The kernel's stride is the number of pixels it moves over the

input matrix. Although stride values of two or more are uncommon, a larger stride produces a smalleroutput. When the filters do not fit the input image, zero-padding is utilized. All members outside of theinput matrix are set to zero, resulting in a larger or similarly sized output. (IBM Cloud Education 2020.)

3.2.2 Pooling layer

Down sampling, often known as pooling layers, is a dimensionality reduction technique that reduces the number of factors in an input. The pooling process, like the convolutional layer, sweeps a filter acrossthe entire input, but this filter has no weights. Instead, the kernel populates the output array by applyingan aggregation function to the values in the receptive field. Pooling is divided into two categories: Max pooling and Average pooling. Max Pooling is a convolutional procedure in which the filter or Kerneltakes the pixel with the highest value from the input and sends it to the output array. This approach isused more commonly as compared to average pooling. Average pooling, on the other hand, is a pooling

technique that employs the average value for patches of a feature map to build a downsampled featuremap. (IBM Cloud Education 2020.)

3.2.3 Fully connected layer

The fully connected layer's name is self-explanatory. In a neural network, fully connected layers arethose where all the inputs from one layer are connected to each activation unit of the next layer. In partially connected layers, the pixel values of the input image are not directly connected to the outputlayer, as previously stated. Each node in the output layer is connected directly to a node in the previouslayer in the fully connected layer. This layer performs categorization based on the features extracted by the preceding layers and the filters applied to them. While convolutional and pooling layers typicallyutilize relu functions to classify inputs, fully connected layers typically use a softmax activation function to produce a probability ranging from 0 to 1. (IBM Cloud Education 2020.)

4 RELATED TECHNOLOGY

Machine learning and deep learning have gained in popularity because of the recent push in the AI industry, and early adopters of this technology are starting to see benefits. Machine learning and deeplearning can be implemented using a variety of programming languages, each of which focuses on a different aspect of the problem. Python is the most common language for machine learning and deeplearning. For a long time, python has been preferred programming language for machine learning anddeep learning researchers. Python provides developers with some of the most flexible and feature-rich tools that improve not only their productivity but also the quality of their code. It is one of the most developer-friendly programming languages, with a diverse set of libraries to serve every use case or projects. (Kumar 2021.)

4.1 Python

Python is a high-level and general-purpose programming language. Python is a programming language that may be used to create desktop gui apps, websites, web applications, mathematics, system scripting and so many. It has an autonomous memory management system and a dynamic type of system. It con- tains a large comprehensive library and supports different programming paradigms such as object-ori- ented, imperative, functional, and procedural. Python is predominantly a dynamic typed programming language that was released in 1991. Guido van Rossum invented it and after that the Python Software Foundation developed it. It has simple syntax that lets programmers to express concepts in fewer lines of code. Primarily it was built with code readability in mind. (GreeksforGeeks 2020.)

Python has a lot of advantages as a programming language. Python is easy to learn, which implies that anyone can learn to write code in a relatively short period of time. Python is one of the easiest object-oriented programming languages when compared to java, c, c++, and c#. It is an open-source program-ming language, which implies that anyone can develop it and contribute to it. Python has an online forum where thousands of programmers come together every day to enhance the language. Python is also free to download and use on any operating system, including windows, mac, and linux. Python has a large range of graphical user interfaces that can be readily integrated into the interpreter, making in one of the

most popular languages among programmers. It understands the concept of class and object encapsula-tion, allowing applications to be more efficient over time. Python comes with many inbuilt libraries that may be imported at any time and utilized in a specific program right out of the box. (Edureka 2021.)

4.2 Tensor Flow

TensorFlow is an open-source machine learning software that focuses on deep neural networks from start to finish. TensorFlow is a collection of libraries, tools, and community resources that are diverse and comprehensive. It enables programmers to construct and deploy cutting-edge machine learning- based applications. The Google Brain team first designed the TensorFlow python deep-learning library for internal usage. The open-

creased since then. There are a few key principles in TensorFlow. Tensors are TensorFlow's fundamental building blocks. In the TensorFlow python deep-learning framework, a tensor is an array

that repre-sents 1 orms of data. Unlike a one-dimensional vector or array or a two-dimensional matrix, a tensorcan have n dimensions. The shape represents dimensionality. A one-dimensional tensor is a vector; atwo-dimensional tensor is matrix; and a zero-dimensional tensor is a scalar. Figure 10 shows varioustensor dimensions. (Sharma 2021.)

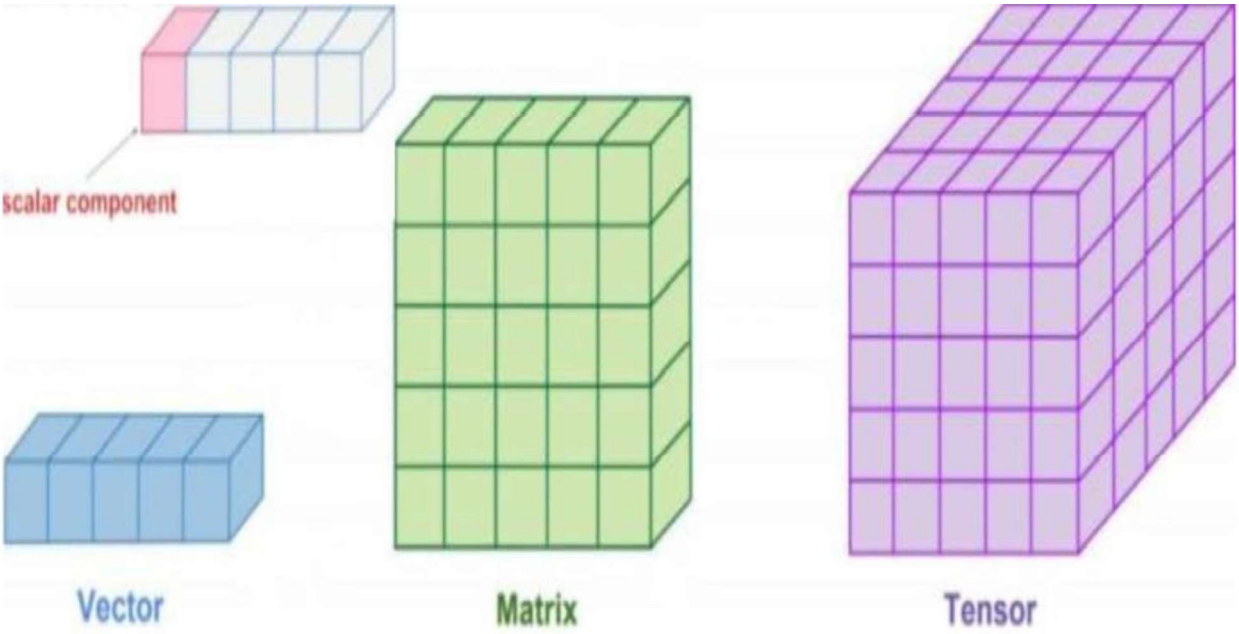


FIGURE 10. Tenso architecture (Analyticsvidhya 2021.)

4.3 Keras

Google developed Keras, a high-level deep learning API for building neural networks. It is written inpython and helps with neural network development. It is modular, quick, and simple to use. It was cre-ated by Google developer Francois Chollet. Low-level computation makes use of a library known as the "Backend". Keras provides the ability to swap between different is not handled by Keras. Instead, it

back ends. TensorFlow, Theano, PlaidML, MXNet, and CNTK (Microsoft Cognitive Toolkit)

are among the frameworks support by Keras. TensorFlow is the only one of these five frameworks that has accepted Keras as its official high-level API. Keras is a deep learning framework that is built on top of TensorFlowand has built-in modules for all neural network computations. Simultaneously, the TensorFlow core API may be used to build custom computations with tensors, computation graphs, sessions, and so on. It gives users complete control and flexibility over their applications, as well as the ability to quickly implement ideas. (Simplilearn 2021.)

4.4 OpenCV-Python

OpenCV is a large open-source library for image processing, machine learning, and computer vision. python, c++, java, and other programming languages are among the languages supported by OpenCV. It can recognize objects, faces, and even human writing by analysing efficient library for numerical operations. One of OpenCV's goals is to provide a simple-to-use computer vision infrastructure that allows individuals to quickly create rather complex vision applications. Over 500 functions in the OpenCV library cover a wide range of vision topics, including factory product inspection, medical im- aging, security, user interface, camera calibration, stereo vision, and robotics. Because computer vision and machine learning are frequently used together, OpenCV also includes a comprehensive machine learning library. (GeeksforGeeks 2021.)

4.5 NumPy and SciPy

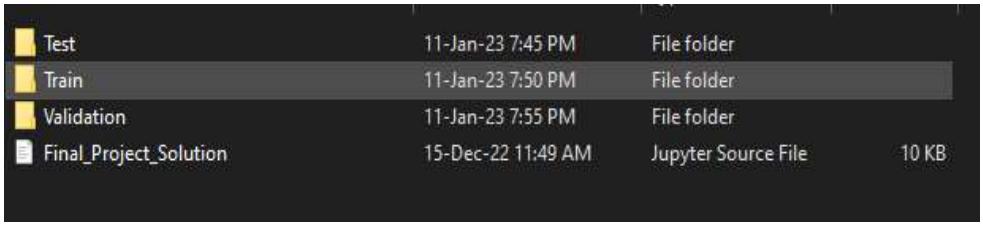
NumPy is the most important python package for scientific computing. It is a library that includes a multidimensional array object, derived objects (such as masked arrays and matrices), and a variety ofroutines for performing fast array operations, such as mathematical, logical, shape manipulation, sorting, selecting, basic linear algebra, basic statistical operations, random simulation, and more. Many other popular python packages, like as pandas and matplotlib, are compatible with NumPy. (NumPy 2022.).On the other hand, SciPy is another important python library for scientific and technical computing thatis free and open source. It is a set of mathematical algorithms and utility functions based on the python numpy extension. It gives the user a lot of power by providing high-level

commands and classes for manipulating and displaying data in an interactive python session. SciPy builds on NumPy, developers do not need to import NumPy if SciPy has already been imported. (Great Learning Team 2020.)

4.6 Imutils and Matplotlib

Imutils are a set of convenience functions for OpenCV and python 2.7 and python 3 that make basic image processing functions including translation, rotation, scaling, skeletonization, and presenting mat- plotlib pictures easier. Matplotlib is an important python visualization library for 2D array plots. Mat- plotlib is a multi-platform data visualization package based on numpy arrays and intended to operate with scipy stack. . It was first introduced in 2022 by John Hunter. One of the most important advantages of visualization is that it provides visual access to large volumes of data in simple images. . Matplotlib has a variety of plots such as line, bar, scatter, histogram, and so on. It is a cross-platform package that provides a variety of tools for creating in python from data stored in lists or arrays, and it is one of the most capable libraries available in python.

1. Splitting Data into Test, Train and Validation set



2. Again Splitting Test, Train and Validation set into two groups WithMask and Without Mask.

3. WithMask and WithoutMask

Final Project - Face Mask Detection ¶

(Solution)

In [1]:
```
import numpy as np
import pandas as pd
%matplotlib inline
import matplotlib as mpl
import matplotlib.pyplot as plt
import os
```

In [2]:
```
import tensorflow as tf
from tensorflow import keras
```

Please enter the correct file path

In [3]:
```
train_dir = r'C:\Deep Learning by Internshala\finalPro\Train'
validation_dir = r'C:\Deep Learning by Internshala\finalPro\Validation'
test_dir = r'C:\Deep Learning by Internshala\finalPro\Test'
```

In [4]:
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

In [5]:
```
train_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)
```

In [5]:
```
train_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
        train_dir,
        target_size=(128, 128),
        batch_size=20,
        class_mode='binary')

validation_generator = test_datagen.flow_from_directory(
        validation_dir,
        target_size=(128, 128),
        batch_size=20,
        class_mode='binary')
```

```
Found 10000 images belonging to 2 classes.
Found 800 images belonging to 2 classes.
```

In [6]:
```
from tensorflow.keras import layers
from tensorflow.keras import models
```

In [7]:
```
model = models.Sequential()

model.add(layers.Conv2D(32, (3, 3), activation='relu',
                        input_shape=(128, 128, 3)))
model.add(layers.MaxPooling2D((2, 2)))

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

model.add(layers.Conv2D(128, (3, 3), activation='relu'))
```

In [7]:
```
model = models.Sequential()

model.add(layers.Conv2D(32, (3, 3), activation='relu',
                        input_shape=(128, 128, 3)))
model.add(layers.MaxPooling2D((2, 2)))

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

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

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

model.add(layers.Flatten())

model.add(layers.Dense(512, activation='relu'))

model.add(layers.Dense(1, activation='sigmoid'))
```

In [8]:
```
from tensorflow.keras import optimizers

model.compile(loss='binary_crossentropy',
              optimizer=optimizers.RMSprop(lr=1e-4),
              metrics=['acc'])
```

```
C:\Users\User\anaconda3\envs\Tensor_Flow\lib\site-packages\keras\optimizers\optimizer_v2\rmsprop.py:135:
argument is deprecated, use `learning_rate` instead.
  super(RMSprop, self).__init__(name, **kwargs)
```

In [9]:
```
history = model.fit_generator(
      train_generator,
      steps_per_epoch=500,
      epochs=20,
      validation_data=validation_generator,
      validation_steps=40)
```

```
C:\Users\User\AppData\Local\Temp\ipykernel_9160\2741540135.py:1: UserWarning: `Model.fit_generator` is deprecated and will be
removed in a future version. Please use `Model.fit`, which supports generators.
  history = model.fit_generator(

Epoch 1/20
500/500 [==============================] - 620s 1s/step - loss: 0.1566 - acc: 0.9392 - val_loss: 0.0476 - val_acc: 0.9850
Epoch 2/20
500/500 [==============================] - 573s 1s/step - loss: 0.0544 - acc: 0.9810 - val_loss: 0.0507 - val_acc: 0.9825
Epoch 3/20
500/500 [==============================] - 570s 1s/step - loss: 0.0340 - acc: 0.9878 - val_loss: 0.0159 - val_acc: 0.9962
Epoch 4/20
500/500 [==============================] - 572s 1s/step - loss: 0.0270 - acc: 0.9915 - val_loss: 0.0397 - val_acc: 0.9825
Epoch 5/20
500/500 [==============================] - 574s 1s/step - loss: 0.0213 - acc: 0.9925 - val_loss: 0.0180 - val_acc: 0.9912
Epoch 6/20
500/500 [==============================] - 605s 1s/step - loss: 0.0190 - acc: 0.9939 - val_loss: 0.0114 - val_acc: 0.9937
Epoch 7/20
500/500 [==============================] - 599s 1s/step - loss: 0.0168 - acc: 0.9949 - val_loss: 0.0079 - val_acc: 0.9962
Epoch 8/20
500/500 [==============================] - 574s 1s/step - loss: 0.0156 - acc: 0.9948 - val_loss: 0.0090 - val_acc: 0.9975
Epoch 9/20
500/500 [==============================] - 572s 1s/step - loss: 0.0131 - acc: 0.9956 - val_loss: 0.0248 - val_acc: 0.9937
Epoch 10/20
500/500 [==============================] - 569s 1s/step - loss: 0.0113 - acc: 0.9963 - val_loss: 0.0414 - val_acc: 0.9925
Epoch 11/20
500/500 [==============================] - 571s 1s/step - loss: 0.0108 - acc: 0.9960 - val_loss: 0.0185 - val_acc: 0.9962
Epoch 12/20
500/500 [==============================] - 580s 1s/step - loss: 0.0088 - acc: 0.9971 - val_loss: 0.0070 - val_acc: 0.9975
Epoch 13/20
500/500 [==============================] - 572s 1s/step - loss: 0.0092 - acc: 0.9970 - val_loss: 0.0118 - val_acc: 0.9975
Epoch 14/20
500/500 [==============================] - 573s 1s/step - loss: 0.0071 - acc: 0.9972 - val_loss: 0.0233 - val_acc: 0.9962
Epoch 15/20
500/500 [==============================] - 573s 1s/step - loss: 0.0060 - acc: 0.9974 - val_loss: 0.0171 - val_acc: 0.9962
Epoch 16/20
500/500 [==============================] - 569s 1s/step - loss: 0.0074 - acc: 0.9978 - val_loss: 0.0066 - val_acc: 0.9975
Epoch 17/20
```

```
500/500 [==============================] - 573s 1s/step - loss: 0.0071 - acc: 0.9972 - val_loss: 0.0233 - val_acc: 0.9962
Epoch 15/20
500/500 [==============================] - 573s 1s/step - loss: 0.0060 - acc: 0.9974 - val_loss: 0.0171 - val_acc: 0.9962
Epoch 16/20
500/500 [==============================] - 569s 1s/step - loss: 0.0074 - acc: 0.9978 - val_loss: 0.0066 - val_acc: 0.9975
Epoch 17/20
500/500 [==============================] - 577s 1s/step - loss: 0.0041 - acc: 0.9986 - val_loss: 0.0070 - val_acc: 0.9987
Epoch 18/20
500/500 [==============================] - 583s 1s/step - loss: 0.0052 - acc: 0.9977 - val_loss: 0.0334 - val_acc: 0.9950
Epoch 19/20
500/500 [==============================] - 580s 1s/step - loss: 0.0041 - acc: 0.9982 - val_loss: 0.0097 - val_acc: 0.9950
Epoch 20/20
500/500 [==============================] - 572s 1s/step - loss: 0.0037 - acc: 0.9988 - val_loss: 0.0083 - val_acc: 0.9975
```

In [10]:
```
model.save("model_cnn_project_P1.h5")
```

In [11]:
```
from tensorflow.keras import backend as K

K.clear_session()
del model
```

In [12]:
```
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,)

test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
        train_dir,
        target_size=(128, 128),
        batch_size=32,
        class_mode='binary')

validation_generator = test_datagen.flow_from_directory(
        validation_dir,
        target_size=(128, 128),
        batch_size=32,
        class_mode='binary')
```

```
Found 10000 images belonging to 2 classes.
Found 800 images belonging to 2 classes.
```

In [13]:
```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu',
                        input_shape=(128, 128, 3)))
model.add(layers.MaxPooling2D((2, 2)))
```

In [13]:
```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu',
                        input_shape=(128, 128, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dropout(0.5))
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(loss='binary_crossentropy',
              optimizer=optimizers.RMSprop(lr=1e-4),
              metrics=['acc'])
```

In [14]:
```
history = model.fit_generator(
      train_generator,
      steps_per_epoch=300,
      epochs=10,
      validation_data=validation_generator,
      validation_steps=25)
```

```
C:\Users\User\AppData\Local\Temp\ipykernel_9160\3626876644.py:1: UserWarning: `Model.fit_generator` is deprecat
emoved in a future version. Please use `Model.fit`, which supports generators.
  history = model.fit_generator(
```

```
Epoch 1/10
```

```
      validation_data=validation_generator,
      validation_steps=25)
```

```
C:\Users\User\AppData\Local\Temp\ipykernel_9160\3626876644.py:1: UserWarning: `Model.fit_generator` is deprecated and will b
emoved in a future version. Please use `Model.fit`, which supports generators.
  history = model.fit_generator(
```

```
Epoch 1/10
300/300 [==============================] - 588s 2s/step - loss: 0.3200 - acc: 0.8640 - val_loss: 0.2088 - val_acc: 0.9137
Epoch 2/10
300/300 [==============================] - 591s 2s/step - loss: 0.2193 - acc: 0.9145 - val_loss: 0.1092 - val_acc: 0.9700
Epoch 3/10
300/300 [==============================] - 572s 2s/step - loss: 0.1862 - acc: 0.9284 - val_loss: 0.1104 - val_acc: 0.9600
Epoch 4/10
300/300 [==============================] - 595s 2s/step - loss: 0.1679 - acc: 0.9388 - val_loss: 0.0829 - val_acc: 0.9787
Epoch 5/10
300/300 [==============================] - 691s 2s/step - loss: 0.1565 - acc: 0.9429 - val_loss: 0.0729 - val_acc: 0.9812
Epoch 6/10
300/300 [==============================] - 640s 2s/step - loss: 0.1553 - acc: 0.9417 - val_loss: 0.1002 - val_acc: 0.9675
Epoch 7/10
300/300 [==============================] - 570s 2s/step - loss: 0.1400 - acc: 0.9477 - val_loss: 0.0606 - val_acc: 0.9837
Epoch 8/10
300/300 [==============================] - 569s 2s/step - loss: 0.1411 - acc: 0.9497 - val_loss: 0.0523 - val_acc: 0.9825
Epoch 9/10
300/300 [==============================] - 564s 2s/step - loss: 0.1294 - acc: 0.9534 - val_loss: 0.0475 - val_acc: 0.9837
Epoch 10/10
300/300 [==============================] - 564s 2s/step - loss: 0.1160 - acc: 0.9596 - val_loss: 0.0393 - val_acc: 0.9875
```

In [15]:
```
from tensorflow.keras.applications import VGG19

conv_base = VGG19(weights='imagenet',
                  include_top=False,
                  input_shape=(128, 128, 3))
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg19/vgg19_weights_tf_dim_ordering_tf_kernels_notop.h5
80134624/80134624 [==============================] - 10s 0us/step
```

In [16]:
```
from tensorflow.keras import models
from tensorflow.keras import layers

model = models.Sequential()
model.add(conv_base)
model.add(layers.Flatten())
model.add(layers.Dense(256, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
```

In [17]:
```
from tensorflow.keras import optimizers

model.compile(loss='binary_crossentropy',
              optimizer=optimizers.RMSprop(lr=2e-5),
              metrics=['acc'])
```

In [18]:
```
checkpoint_cb = keras.callbacks.ModelCheckpoint("CNN_Final_Project_Model-{epoch:02d}.h5")
```


In [22]:
```
history = model.fit_generator(
      train_generator,
```

In [18]:
```
checkpoint_cb = keras.callbacks.ModelCheckpoint("CNN_Final_Project_Model-{epoch:02d}.h5")
```

In [22]:
```
history = model.fit_generator(
      train_generator,
      steps_per_epoch=5,
      epochs=5,
      validation_data=validation_generator,
      validation_steps=25,
      callbacks=[checkpoint_cb])
```

```
C:\Users\User\AppData\Local\Temp\ipykernel_9160\3897155067.py:1: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.
  history = model.fit_generator(
Epoch 1/5
5/5 [==============================] - 781s 183s/step - loss: 0.0872 - acc: 0.9563 - val_loss: 0.1848 - val_acc: 0.9275
Epoch 2/5
5/5 [==============================] - 780s 183s/step - loss: 0.0925 - acc: 0.9688 - val_loss: 0.0202 - val_acc: 0.9950
Epoch 3/5
5/5 [==============================] - 780s 183s/step - loss: 0.0858 - acc: 0.9812 - val_loss: 0.0094 - val_acc: 0.9975
Epoch 4/5
5/5 [==============================] - 781s 183s/step - loss: 0.0812 - acc: 0.9688 - val_loss: 0.0115 - val_acc: 0.9975
Epoch 5/5
5/5 [==============================] - 777s 182s/step - loss: 0.0491 - acc: 0.9937 - val_loss: 0.0095 - val_acc: 0.9987
```

Epoch 2/5
5/5 [==============================] - 780s 183s/step - loss: 0.0925 - acc: 0.9688 - val_loss: 0.0202 - val_acc: 0.9950
Epoch 3/5
5/5 [==============================] - 780s 183s/step - loss: 0.0858 - acc: 0.9812 - val_loss: 0.0094 - val_acc: 0.9975
Epoch 4/5
5/5 [==============================] - 781s 183s/step - loss: 0.0812 - acc: 0.9688 - val_loss: 0.0115 - val_acc: 0.9975
Epoch 5/5
5/5 [==============================] - 777s 182s/step - loss: 0.0491 - acc: 0.9937 - val_loss: 0.0095 - val_acc: 0.9987

In [23]:
```
test_generator = test_datagen.flow_from_directory(
        test_dir,
        target_size=(128, 128),
        batch_size=32,
        class_mode='binary')
```

```
Found 992 images belonging to 2 classes.
```

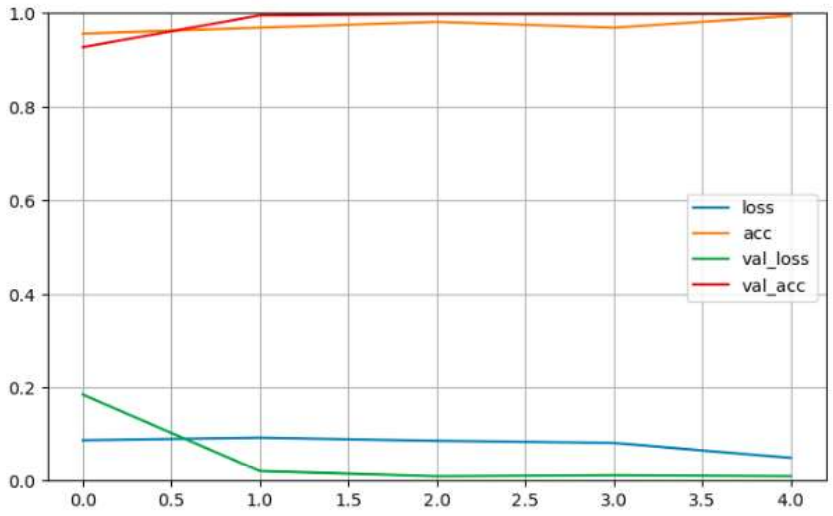
In [24]:
```
model.evaluate_generator(test_generator, steps=31)
```

```
C:\Users\User\AppData\Local\Temp\ipykernel_9160\1766430975.py:1: UserWarning: `Model.evaluate_generator` is deprecated and will be removed in a future version. Please use `Model.evaluate`, which supports generators.
  model.evaluate_generator(test_generator, steps=31)
```

Out[24]:
```
[0.018437422811985016, 0.9939516186714172]
```

Out[24]: [0.018437422811985016, 0.9939516186714172]

In [25]:
```
pd.DataFrame(history.history).plot(figsize=(8, 5))
plt.grid(True)
plt.gca().set_ylim(0, 1)
plt.show()
```



In [28]:
```
conv_base.summary()
```

```
Model: "vgg19"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 128, 128, 3)]     0
block1_conv1 (Conv2D)        (None, 128, 128, 64)      1792
block1_conv2 (Conv2D)        (None, 128, 128, 64)      36928
block1_pool (MaxPooling2D)   (None, 64, 64, 64)        0
block2_conv1 (Conv2D)        (None, 64, 64, 128)       73856
block2_conv2 (Conv2D)        (None, 64, 64, 128)       147584
block2_pool (MaxPooling2D)   (None, 32, 32, 128)       0
block3_conv1 (Conv2D)        (None, 32, 32, 256)       295168
block3_conv2 (Conv2D)        (None, 32, 32, 256)       590080
block3_conv3 (Conv2D)        (None, 32, 32, 256)       590080
block3_conv4 (Conv2D)        (None, 32, 32, 256)       590080
block3_pool (MaxPooling2D)   (None, 16, 16, 256)       0
block4_conv1 (Conv2D)        (None, 16, 16, 512)       1180160
block4_conv2 (Conv2D)        (None, 16, 16, 512)       2359808
block4_conv3 (Conv2D)        (None, 16, 16, 512)       2359808
block4_conv4 (Conv2D)        (None, 16, 16, 512)       2359808
block4_pool (MaxPooling2D)   (None, 8, 8, 512)         0
block5_conv1 (Conv2D)        (None, 8, 8, 512)         2359808
block5_conv2 (Conv2D)        (None, 8, 8, 512)         2359808
block5_conv3 (Conv2D)        (None, 8, 8, 512)         2359808
block5_conv4 (Conv2D)        (None, 8, 8, 512)         2359808
block5_pool (MaxPooling2D)   (None, 4, 4, 512)         0
```

```
block4_conv4 (Conv2D)       (None, 16, 16, 512)       2359808

block4_pool (MaxPooling2D)  (None, 8, 8, 512)         0

block5_conv1 (Conv2D)       (None, 8, 8, 512)         2359808

block5_conv2 (Conv2D)       (None, 8, 8, 512)         2359808

block5_conv3 (Conv2D)       (None, 8, 8, 512)         2359808

block5_conv4 (Conv2D)       (None, 8, 8, 512)         2359808

block5_pool (MaxPooling2D)  (None, 4, 4, 512)         0

=================================================================
Total params: 20,024,384
Trainable params: 20,024,384
Non-trainable params: 0
_________________________________________________________________
```