# COLOR DETECTION

Major project report is submitted in partial fulfilment of requirement for the degree of Bachelor of Technology

in

**Computer Science and Engineering**

By

SUNNY MITTAL (181230)
KARTIK GUPTA (181290)

**UNDER THE SUPERVISION OF**

DR PARDEEP KUMAR KHOKHAR

Department of Computer Science Engineering and Information Technology

Jaypee University of Information Technology, Waknaghat, 173234, Himachal Pradesh, INDIA

# DECLARATION

We hereby declare that this project has been done by us under the supervision of **DR PARDEEP KUMAR KHOKHAR, PROFESSOR,** Jaypee University of Information Technology. We also declare that neither this project nor any part of this project has been submitted elsewhere for award of any degree or diploma.

**Supervised by:**
**(DR PARDEEP KUMAR KHOKHAR)**
PROFESSOR
Department of Computer Science & Engineering and Information Technology
Jaypee University of Information Technology

**Submitted by:**
**(SUNNY MITTAL) (181230)**
**(KARTIK GUPTA) (181290)**
Computer Science & Engineering Department
Jaypee University of Information Technology

# CERTIFICATE

This is to certify that the work which is being presented in the project report titled **“COLOR DETECTION**” in partial fulfilment of the requirements for the award of the degree of B.Tech in Computer Science And Engineering and submitted to the Department of Computer Science And Engineering, Jaypee University of Information Technology, Waknaghat is an authentic record of work carried out by “SUNNY MITTAL 181230 and Kartik Gupta 181290**”** under the supervision of **DR PARDEEP KUMAR KHOKHAR**, Department of Computer Science and Engineering, Jaypee University of Information Technology, Waknaghat.

SUNNY MITTAL (181230)
KARTIK GUPTA (181290)

The statement made above is correct and is best of our knowledge.

(DR PARDEEP KUMAR KHOKHAR)
PROFESSOR
Computer Science Engineering and Information Technology
Jaypee University of Information Technology, Waknaghat,

# ACKNOWLEDGMENT

Firstly, we express our heartiest thanks and gratefulness to almighty God for His divine blessings that made it possible for us to complete the project work successfully.

We are really grateful and wish our indebtedness to our Supervisor **DR PARDEEP KUMAR KHOKHAR, PROFESSOR**, Department of CSE Jaypee University of Information Technology, Waknaghat. Deep Knowledge & interest of our supervisor in the field to carry out this project. His endless patience, scholarly guidance, continuous encouragement, energetic constant supervision, criticism that was constructive, valuable advice, checking our drafts and correcting all of them at every stage have made it possible to complete this project.

We would express our gratitude to **DR PARDEEP KUMAR KHOKHAR,** Department of CSE, for helping us finish our project.

We would also welcome each one of those individuals who have helped me straightforwardly or in a roundabout way in making this project a win. In this unique situation, we want to thank our staff individuals, instructing as well non-instructing, who developed their help and facilitated our undertaking.

Finally, we must acknowledge with due respect the constant support and patience of our parents.

SUNNY MITTAL
KARTIK GUPTA

# COLOR CLASSIFICATION USING MACHINE LEARNING

Machine learning as artificial intelligence continue to be fields focused on real problems. Machine Learning uses computers to make prophecy based on provided set .

By the usage of machine learning methods such as unsupervised & supervised learning we are able to process large data and solve classification problems. In this project, we have applied supervised learning, which is the most often used task-driven classification type machined learning. The objective of this project classification of different color shades using machine learning under ideal and different non-ideal conditions. In this project, we used a binary classification technique of supervised learning to classify different colors.

# TABLE OF CONTENTS

LIST TABLES

LIST OF FIGURES

# CHAPTER 1

# INTRODUCTION

## 1.1 Problem and Purpose

The problem addressed in the project consists of classification of color shades under different ideal and non-ideal lighting conditions using algorithms of Machine Vision Machine Learning . Examples of non-ideal light conditions include bright light, dark light, etc.

Our aim is recognising a color from images captured from a camera or to check it with various lighting conditions in order to check the prediction. Along with this experiment, the other purpose build or tune the learned model  so find some design, predict the data to study the difference between the machined prediction with ideal or nonideal compared with ideal light condition.

# Chapter 2
# BACKGROUND OF THE STUDY

## 2.1 Review of Research

Machine studying algorithms are state-of-the-art strategies that may be used to remedy the shadeation type problem. Machine Vision area is shifting from conventional statistical techniques and algorithms to Neural Network techniques [6].

Many demanding situations exist in Machine Vision. By and large, mind community techniques are extraordinarily useful for express troubles contrasted with commonplace algorithms. Color type the use of K-Nearest Neighbors Machine Learning set of rules with function extraction is one conventional methods of shadeation popularity. While spotlight extraction is a sizable variable, consist of extraction like Color Histogram, Color Correlogram, Color Moments can likewise be utilized. Another conventional gadget studying set of rules is the KMeans set of rules which may be used to extract colorations from photos to categorise every photograph from a set of photos primarily based totally on shadeation area values, wherein any color type area may be used like RGB, CYMK, HSV, etc.

Artificial Neural Network approved to make computational fashions that surpass the presentation of synthetic understanding with commonplace algorithms. One instance is Convolutional Neural Networks (CNN), which can be used usually and efficiently to remedy tough photograph-pushed sample popularity problems. Therefore, CNN are used for type on this project.

## 2.2 Machine Learning

Machine Learning (ML) is one of the energetic regions of Artificial Intelligence. The goal of ML is to permit the gadget to learn. We additionally need to deliver the gadget a few cappotential to reply to feedback [6]. The essential distinction among conventional programming and ML is that in preference to instructions, we want to enter statistics [17]. Also, in preference to a predefined response, the aim of gadget mastering algorithms is to assist the gadget discover ways to reply.

Data also are vital in ML. If we plan to apply ML, then it's far vital to gain excessive quality & various datasets [17]. If the statistics set is better, then the set of rules and the product can be better. An instance is proven beneath in Figure 1.

Figure 1: Data Flywheel [20]

In gadget mastering, we use techniques to assist our software locate styles in large datasets. As proven in determine 2, an unique statistics set divided right into a education set and check set, normally in a 70:30 ratio, respectively.
A education set is a smaller set this is used to music the algorithms. In turn, with those algorithms' assist, we are able to create a version on the way to paintings for the bigger dataset of the unique statistics.

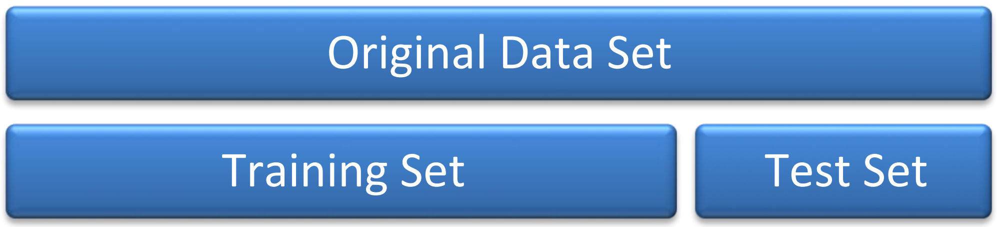


Figure 2: Data set

If we create a easy version, this could paintings for small education sets, however it's miles much less bendy while we're searching out massive data. This is usually referred to as underfitting [23]. If you're underfitting your version to the data, it’s now no longer shooting sufficient information, so it makes an misguided prediction.

On the opposite hand, we may want to create a version that’s bendy sufficient to paintings with the dataset however so complicated and tough to understand. This is usually referred to as overfitting [23].

### 2.1.1 **Different sorts of Machine Learning**

There are 3 approaches a device can learn [6], as proven beneath in Figure 3.

1) Supervised Learning
2) Unsupervised Learning
3) Reinforcement Learning

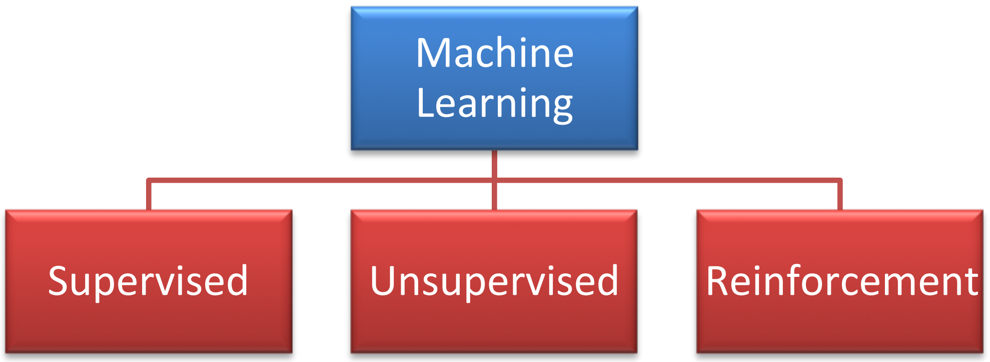


Figure 3: Types of Machine Learning [6]

#### 2.1.1.1 Supervised Learning

When we understand sufficient approximately the information, we will use supervised getting to know. In supervised getting to know, we display the system the relationship among exceptional variables and regarded outputs. In supervised ML, the variables consist of classified pattern information and regarded output known as accurate output. The classified information is the input, that is the impartial variable, and structured variable will be the output. Supervised getting to know is task-driven (type) with 3 type sorts as proven underneath in Figure 4.

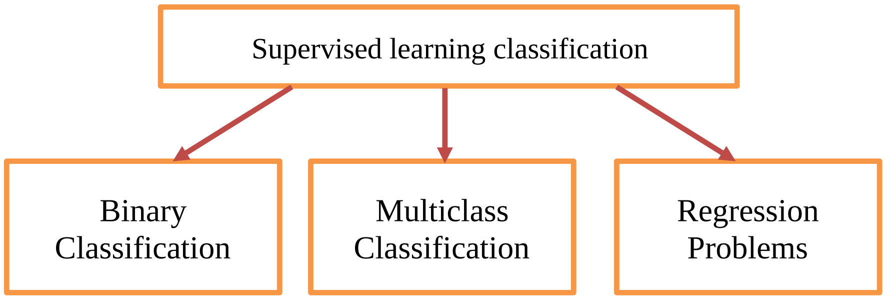


Figure 4: Supervised Learning Categories

### 2.1.1.2 Unsupervised Learning

Learning and enhancing with the aid of using trial and mistakess is the important thing to unsupervised getting to know. Unlike supervised getting to know [19], we aren't giving categorised information or displaying the proper solution to the gadget. Instead, we're the use of extraordinary algorithms to permit the gadget create connections with the aid of using reading and gazing the information.

Unsupervised getting to know is a information-driven (clustering) getting to know set of rules, the gadget receives unlabeled information, with the aid of using reading and gazing the information. The gadget clusters the information into extraordinary groups. The key component with unsupervised getting to know is get right of entry to to a huge quantity of information. The greater information, the simpler it's miles for a gadget to examine and look at developments that could cause a profitable cluster.

#### 2.1.1.3 Reinforcement

Reinforcement getting to know is absolutely extraordinary from supervised and unsupervised getting to know [17]. Reinforcement getting to know has the gadget iterate to constantly enhance the outcome. Overtime, the gadget ought to zero-in and get nearer and toward excessive fine output [19].

In reinforcement getting to know the set of rules learns to react to an environment. You are reinforcing positive methods which you need the gadget to behave. Instead of simply look at and examine, we're giving the gadget a totally clean goal.

Q-getting to know is a sort of reinforcement getting to know, it's miles one of the maximum promising regions in gadget getting to know. A gadget can play video games or run algorithms after which study the consequences of players. If a tremendous occasion occurs, then the gadget can analyze or improve the set of rules [22] and preserve getting to know.

## 2.3 Classification

As we noticed above, type is a supervised getting to know sort of gadget getting to know. It makes a decision the elegance to which the information belong to. This method is to categorize enter information into respective and extraordinary wide variety of instructions and assign labels to each elegance [2].

Classification has types [2]:

1) Binomial/Binary

2) Multi-Class

<u>Classification may be utilized in following packages</u> [2]:

• Is acquired e-mail a junk mail or ham?

• Verify signature/handwriting popularity.

• What is the elegance of a given image?

• Document type.

• Medical x-ray labelling .

• Face popularity from the picture and label names [15].

• Classification of kinds of vegetation or soil.

## 2.4 Applications

This venture offers with shadeation type the use of getting to know methods. Color type the use of gadget imaginative and prescient and gadget getting to know has many packages [10]. Some packages are indexed below.

**Agriculture**

• To differentiate among crop and soil while the use of robotics to become aware of and reduce vegetation.

• To decide ripe end result.

• To become aware of and discard rotten vegetables/end result the use of robot arm.

• To locate nutrient deficiency primarily based totally on leaf yellowing.

**Autonomous vehicles**

• Traffic sign detection and popularity [15].

**Road and Safety**

• Vehicle shadeation identity in extraordinary mild conditions

Carpentry and family services

• Identify timber kind primarily based totally on shadeation

• Determine wall paint shadeation for repainting

**Cosmetology**

• Determine pores and skin tone

Landscape architecture

• For panorama observations from aerial view and type of land, forest, water regions [10].

## 2.5 Artificial Neural Networks

As we understand gadget getting to know algorithms are designed to are searching for out extraordinary styles withinside the information. Artificial Neural Networks are an incredibly strong manner for machines to locate the styles. This way we are able to classify numerous thousand snap shots in an instant, or we are able to translate to extraordinary languages and transcribe audio into textual content files [22]. Because synthetic neural networks are a sort of gadget getting to know, we nevertheless want very huge datasets.

We can use supervised getting to know with a small schooling set after which have the community locate styles. Then we use those styles to run in opposition to our take a look at information. We also can use unsupervised getting to know, wherein we permit the neural community locate styles in unlabeled information. Our community would possibly locate new clusters that we haven't considered. This works mainly nicely for large information sets [22].

There are a pair of factors we need to preserve in thoughts with synthetic neural networks. The first is identical as gadget getting to know [22], we're going to depend on a huge quantity of splendid information. That's how synthetic neural networks locate new styles. If we don't deliver the information, then our synthetic neural community will now no longer have the possibility to analyze new things [22]. The 2nd component to preserve in thoughts is that while [22] we're operating on an synthetic neural community, we're going to be the use of an empirical approach. That way that we'll run small experiments to attempt to fine-track our community to get higher consequences. The contemporary-day synthetic neural community offers us masses of possibilities to extrade our configurations. These are referred to as the networks' hyper-parameters.

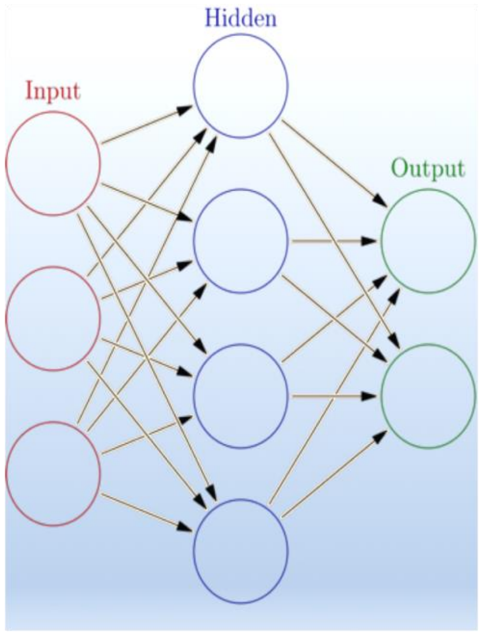


Figure 5: Neural Network Layers [14]

Artificial Neural Networks use plenty of the identical language as neuroscience. Our mind is full of neurons. We research new matters whilst the cells shape networks of connections. In a sense, inside our mind, there's a community of connections that assist us pick out and classify new matters. An synthetic neural community additionally makes use of neurons, besides those neurons have a numerical cost as a manner to maintain information. Much like our organic mind, an synthetic neural community receives a whole lot of its electricity from the connections among distinct neurons. An synthetic neural community organizes those neurons into layers. There are enter layers, numerous hidden layers and an output layer as proven in determine 5. A neural community will have many distinct hidden layers. In fact, greater layers we've the greater processing we are able to do to discover a complicated pattern. This synthetic neural community regularly known as deep learning. It can research many new matters due to the fact we've an synthetic neural community this is numerous layers deep.

# Chapter 3

# LEARNING

## 3.1 CNN Neural Network Model

The structure of a convolutional neural network is shown below in Figure 6.

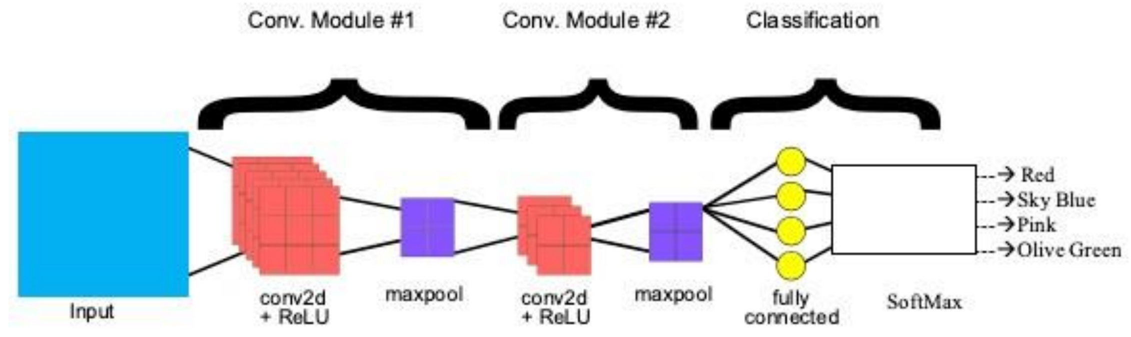


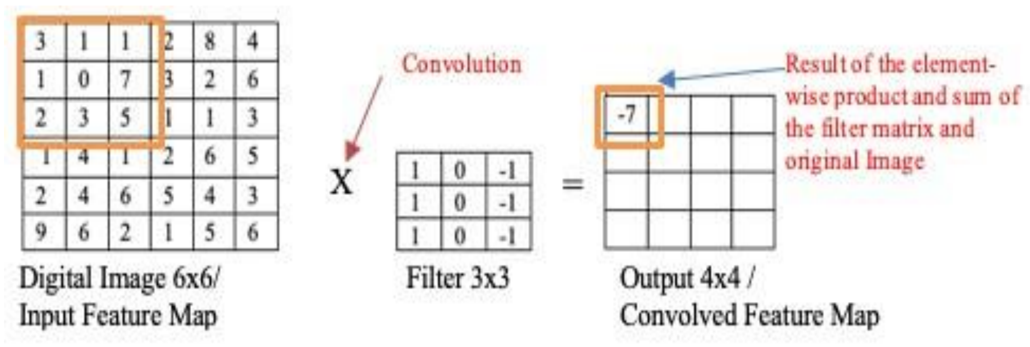
Figure 6: End-to-end structure of a convolution neural network [18].

Fully related neural networks normally don’t paintings properly on pics. This is due to the fact every pixel is an enter, as we upload greater layers, the quantity of parameters will increase exponentially [19]. What makes one photograph distinguishable from every other is its unique structure [19]. Areas near every different or location regions are surprisingly great in pics. CNN can be used to extract a higher- and higherlevel illustration of photograph contents.

### 3.1.1 Convolution

At first, CNN receives enter characteristic map that is a third-dimensional matrix in which the dimensions of the primary dimensions corresponds to the duration and width of the pics in pixels. The length of the 0.33 measurement is 3 (RGB shadeation photograph) [18].

In CNN, a convolution extracts one tile (3x3 or 5x5 pixels) of the enter characteristic map and applies filters (identical length because the tile) over them and produces an output characteristic map or convolved characteristic.
In this convolution step, as proven under in Figure 7, the filters efficiently slide over the enter characteristic map's grid left to proper and pinnacle to bottom, one pixel at a time [15], extracting every respective tile [18].



3 1 1 2 8 4
1 0 7 3 2 6
2 3 5 1 1 3
1 4 1 2 6 5
2 4 6 5 4 3
9 6 2 1 5 6
Digital Image 6x6/ Input Feature Map
Convolution
X
1 0 -1
1 0 -1
1 0 -1
Filter 3x3
=
-7
Output 4x4 / Convolved Feature Map
Result of the element-wise product and sum of the filter matrix and original Image

Figure 7: Convolution Neural Network [18]

For every clear out out-tile, the CNN does the element-sensible multiplication of the clear out out matrix and the tile matrix, after which upload all of the factors of the ensuing matrix to get a unmarried price. This operation is just like the dot product. Each of ensuing values from this dot product for each clear out out-tile pair is then output withinside the convolved function matrix.

During education, the CNN “learns” the top-rated values for the clear out out matrices that allow it to extract significant functions from the enter function map [18]. As the variety of filters carried out to the enter function map will increase, so does the variety of functions the CNN can extract however education time additionally will increase as extra filters are delivered on CNN. Also, each new clear out out delivered to the community offers much less incremental price than the preceding one. Thus, we want to assemble a community that makes use of the minimal variety of filters had to extract the functions essential for correct image classification [18].

### 3.1.2 Activation capabilities

An activation characteristic is a part of the neural community. The activation characteristic decides  if a neuron fires or not. To make certain that there may be a few nonlinearity in our community, we want to ensure that those activation capabilities are nonlinear [19]. We can use a step characteristic as our activation characteristic, which offers an output of 0 or one. If the output is above a sure threshold, then neuron is fired, and we've one. If the price of the output is much less than the threshold, then it isn't always fired, and we've a 0 [19].

### 3.1.3 ReLU

The ReLU is one of the maximum famous Activation capabilities. The ReLU or “Rectified Linear Unit” output, a 0 for any price of x this is much less than 0. For any price of x identical to or extra than 0, the characteristic returns x. After every convolution operation of CNN, the community applies a Rectified Linear Unit

(ReLU) transformation to the convolved function, so that you can introduce nonlinearity into the model.

### 3.1.4 Pooling

Pooling is the subsequent step after ReLU, wherein the CNN reduces the sampling of the convolved function, and the variety of dimensions of function map, whilst nevertheless retaining the maximum vital function information. This manner is referred to as max pooling, as proven in Figure eight below, and its miles one of the maximum not unusual place algorithms.

There are different swimming pools as well, just like the common pool and min pool. Max pooling works withinside the equal way as convolution. It slides over the function map and extracts tiles of a special length. Then shape every extracted tile, the most price out of that tile is outputted to a brand-new function map, and all different values are discarded. Max pooling operations take parameters [18]. The length of max- pooling clear out out is generally 2x2 pixels.

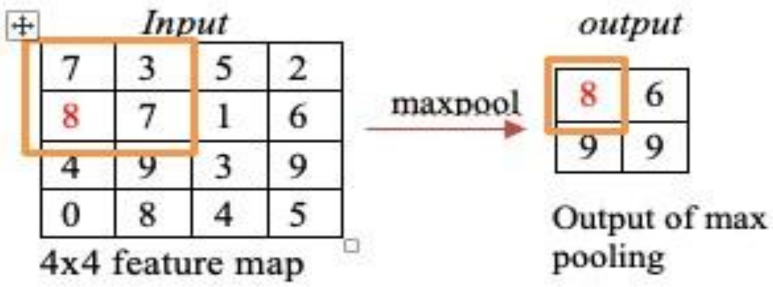


Figure 8: maxpool [18]

The distance, in pixel, isolating every extracted tile is known as stride. It's distinct from convolution, in which convolution filters slide over the function map pixel via way of means of pixel, however in max pooling, the stride determines the places in which every tile is extracted. For a 2x2 clear out out size, a stride of two specifies that the max pooling operation will extract all non-overlapping 2x2 tiles from the function map.

### 3.1.5 Fully Connected Layers

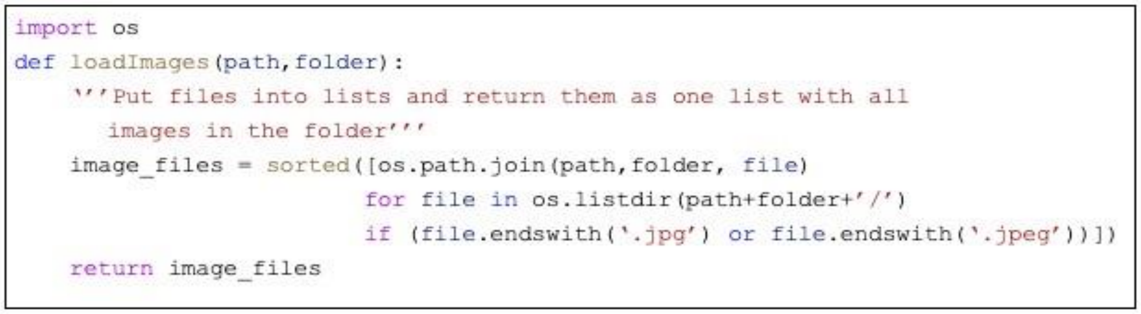
One or greater absolutely linked layers are on the quit of a convolutional neural network. Two layers are absolutely linked while each node withinside the first layer is hooked up to each node withinside the second  layer [18]. Their task is to carry out class primarily based totally at the function extracted via way of means of the convolutions. Mostly, the quit is absolutely linked with neurons [18]. This very last absolutely linked layer consists of a SoftMax activation function, which offers an output opportunity cost from zero to at least one for every of the class labels the version is attempting to predict [18]. Figure eight illustrates the quit-to-quit shape of a convolution neural network.

## 3.2 Model Implementation

As we know, deep studying is the maximum not unusualplace form of device studying used to categorise pictures the use of CNN. For this, the satisfactory library in Python is Keras, which makes it quite easy to construct a CNN version.

The first step is to obtain the dataset. Either download the dataset or create your own dataset. We need 3:1 proportion of training image and test images respectively, and then we need to load those image files to train the model.

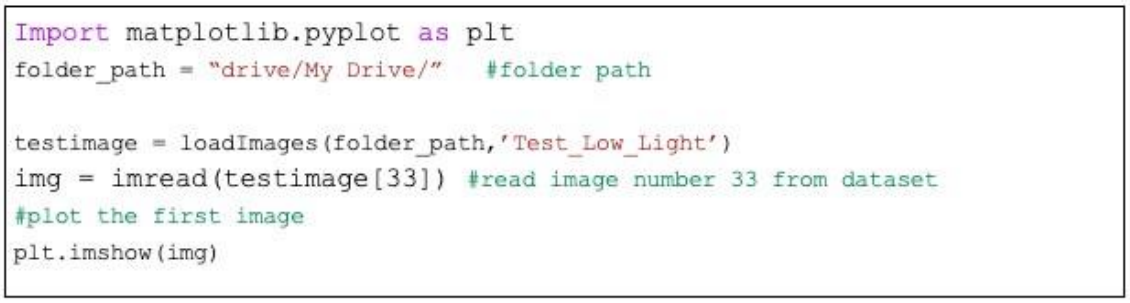
Here is the definition to load image files:



```
import os
def loadImages(path,folder):
    '''Put files into lists and return them as one list with all
      images in the folder'''
    image_files = sorted([os.path.join(path,folder, file)
                          for file in os.listdir(path+folder+'/')
                          if (file.endswith('.jpg') or file.endswith('.jpeg'))])
    return image_files
```

### 3.2.1 Data analysis

Let's check the image from the dataset, as shown below in Figure 9. To plot the image from dataset we need 'plot' function, and to check its size we need to use 'shape' function.



```
Import matplotlib.pyplot as plt
folder_path = "drive/My Drive/"   #folder path

testimage = loadImages(folder_path,'Test_Low_Light')
img = imread(testimage[33]) #read image number 33 from dataset
#plot the first image
plt.imshow(img)
```

```
<matplotlib.image.AxesImage at 0x7f987e532358>
```

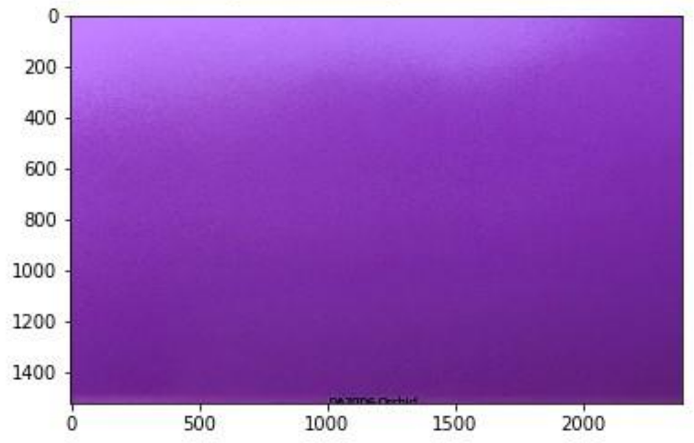


Figure 9: Read image from dataset

Let’s check the size of image above by using *shape* function.

```
#check shape of our image
testimage[0].shape
```

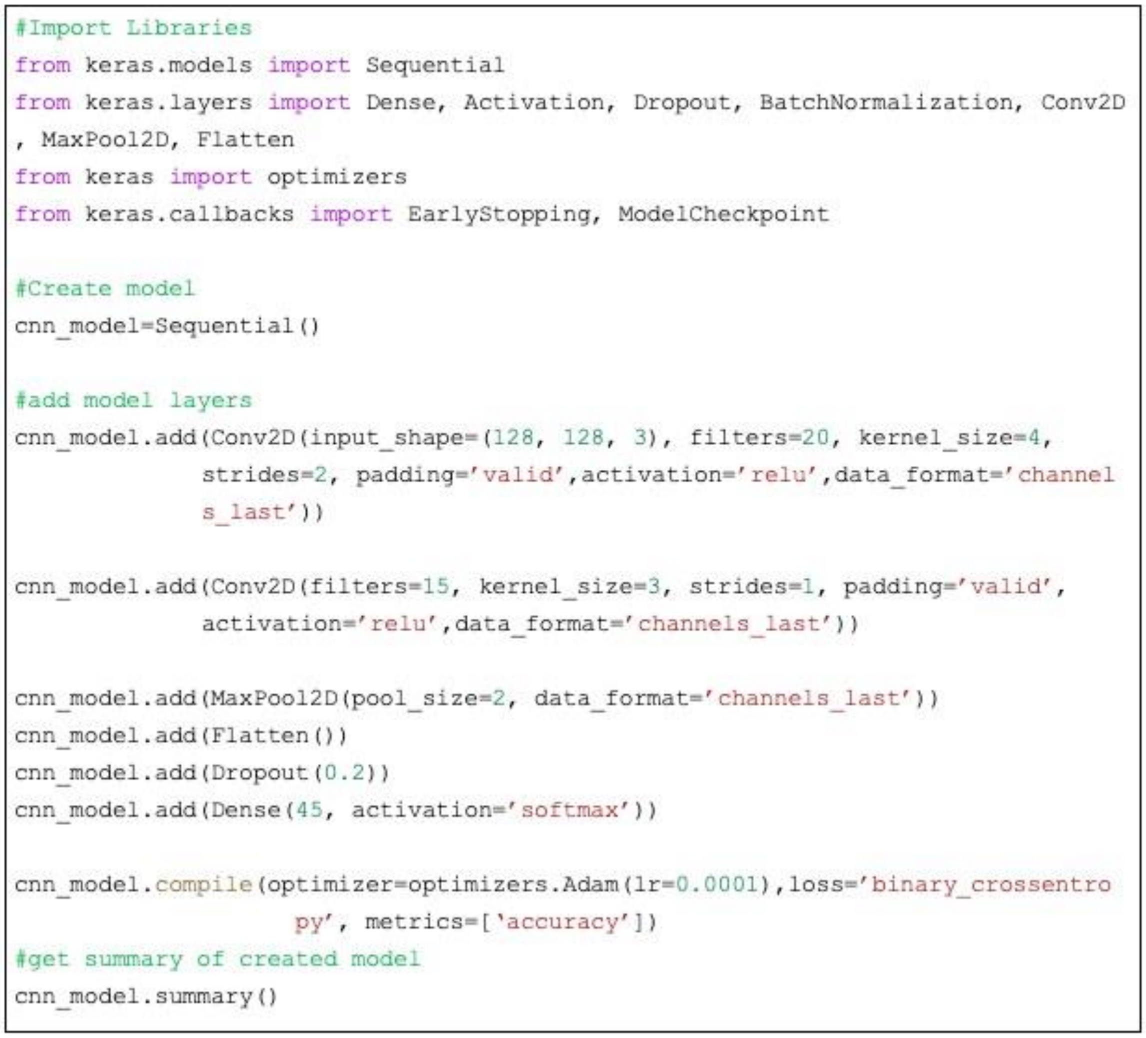
Figure 10 below is the 2D shape of the above image, with the 3 signifying that the image is RGB

```
(1510, 2361, 3)
```

Figure 10: Image size

### 3.2.2 Building the model

Once the image files are loaded, we are ready to build our model. The code is below:



```
#Import Libraries
from keras.models import Sequential
from keras.layers import Dense, Activation, Dropout, BatchNormalization, Conv2D
, MaxPool2D, Flatten
from keras import optimizers
from keras.callbacks import EarlyStopping, ModelCheckpoint

#Create model
cnn_model=Sequential()

#add model layers
cnn_model.add(Conv2D(input_shape=(128, 128, 3), filters=20, kernel_size=4,
              strides=2, padding='valid',activation='relu',data_format='channel
              s_last'))

cnn_model.add(Conv2D(filters=15, kernel_size=3, strides=1, padding='valid',
              activation='relu',data_format='channels_last'))

cnn_model.add(MaxPool2D(pool_size=2, data_format='channels_last'))
cnn_model.add(Flatten())
cnn_model.add(Dropout(0.2))
cnn_model.add(Dense(45, activation='softmax'))

cnn_model.compile(optimizer=optimizers.Adam(lr=0.0001),loss='binary_crossentro
                  py', metrics=['accuracy'])
#get summary of created model
cnn_model.summary()
```

We are the use of Keras library in python so let’s import that first [13]. The version kind we're the use of is sequential kind. Sequential version kind is the perfect technique to construct a version in Keras. Using this Keras library we are able to construct a version layer via way of means of layer. You can see above withinside the code that we're including version layers one via way of means of one. The upload() feature is used to feature layer to our CNN version.

In our CNN version, the primary  layers are Conv2D layers. Conv2D is a convolution layer to cope with our enter pictures [18]. Because our enter photo is a 2-Dimensional matrix.

For our convolution we want a clear out out matrix, the dimensions of clear out out matrix is described via way of means of Kernel. So, right here the Kernel_size is 20. A Kernel length of 20 method we are able to have 20x20 clear out out matrix [13].

Here Activation is the activation feature for the layers [18]. For the primary  layers we're the use of the activation feature ReLU or Rectified Liner Activation. This ReLU activation feature works properly with neural networks.

The first conv2D layer additionally takes in an enter form. So, we're defining an enter form of every enter photo 128,128,three, in which the three method that the pictures are shadeation now no longer grayscale.

MaxPool2D layer lets in to carry out a pooling operation to calculate the most price in every Kernel patch. We can upload most pooling operation to the version via way of means of simply including MaxPooling2D or MaxPool2D layer supplied via way of means of Keras [13].

In among the MaxPool2D layers and the Dense layer, there's a ‘Flatten’ layer. This flatten layer is a connection among the MaxPool or Convolution layer and dense layer. It serves as a connection among  layers [18].

‘Dense’ is a widespread layer kind used in lots of instances for neural networks, which we are able to use for the output layer [3]. We may have forty five nodes in our output layer, every feasible final results for every shadeation (0-44). The remaining however maximum vital activation is ‘softmax’. Softmax activation withinside the dense layer makes the outputs sum as much as 1 [3]. So, that we are able to test the chances for the output. The version predicts the shadeation primarily based totally on the best probability [3].

### 3.2.3 Compiling the version

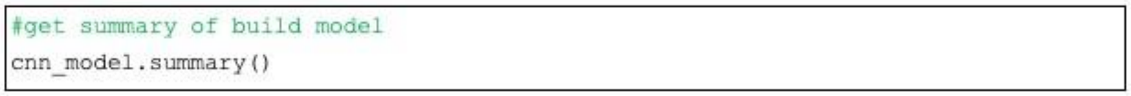
We constructed our version, now we want to bring together it. To bring together the version, we want 3 parameters as proven below.

1) Optimizer:
The optimizer controls the studying fee(lr). In our case, we're the use of ‘adam’ as an optimizer [3]. This optimizer adjusts the studying fee of training. Here we used adam lr=0.0001 all through the training.

The studying fee comes to a decision how rapid or gradual the ultimate weights for the version are calculated. Slow ultimate weight calculation via way of means of smaller studying fee results in extra correct weights, however as it’s gradual the time to compute the weights for the version may be longer.

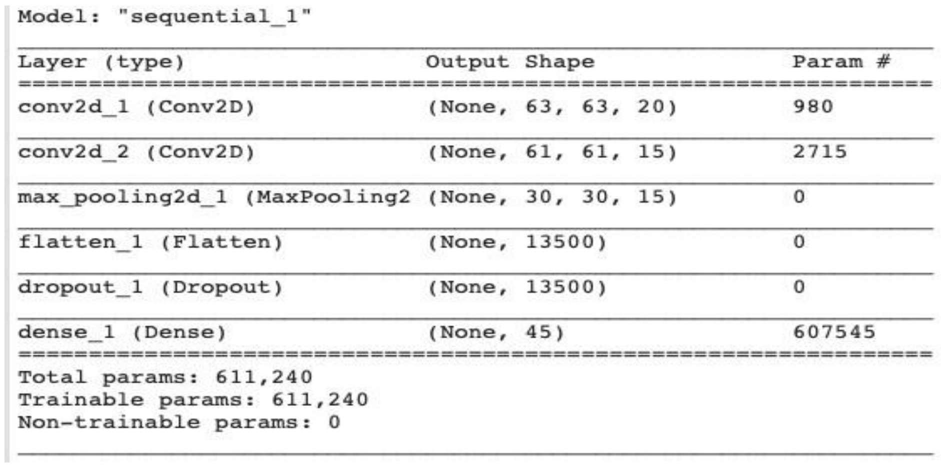
2) Loss:
We are the use of ‘binary_crossentropy’ for our loss feature [3].
three) Metric: To interpret easily, we're the use of the ‘accuracy’ metric to look the accuracy rating at the validation set while we teach the version [21].

Now let’s get a summary of our model. To create a summary of the model we need to call a ***summary()*** function which is shown below.



```
#get summary of build model
cnn_model.summary()
```

Summary of our model is shown in figure 11.



```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 63, 63, 20)        980
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 61, 61, 15)        2715
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 30, 30, 15)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 13500)             0
_________________________________________________________________
dropout_1 (Dropout)          (None, 13500)             0
_________________________________________________________________
dense_1 (Dense)              (None, 45)                607545
=================================================================
Total params: 611,240
Trainable params: 611,240
Non-trainable params: 0
_________________________________________________________________
```

**Figure 11: Summary of the model**

## 3.3 Model Training and Prediction

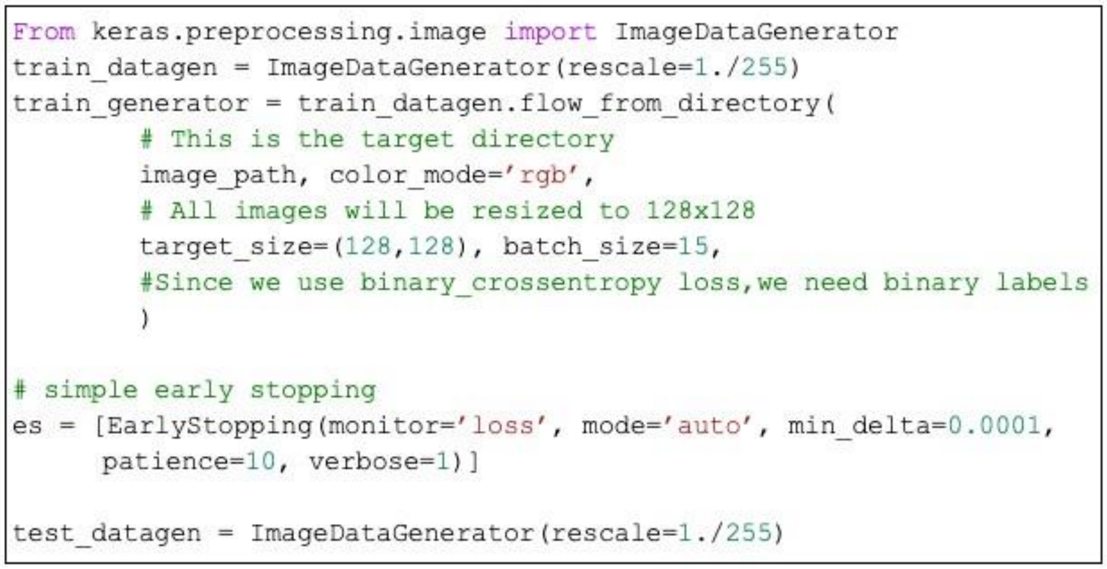
### 3.3.1 Training the model

So far, we constructed and compiled our model. Now it’s time to teach the model. In the sphere of system studying and system imaginative and prescient we want a big dataset of photograph and video files, and that’s one of the challenges [7]. Because whilst loading and processing a big wide variety of pix or video records set, we likely encountered a state of affairs of now no longer having sufficient reminiscence in our system [7]. So, for loading and processing pix in Keras library, we want to construct Data Generators.

### 3.3.2 ImageDataGenerator function

In our software of photograph classification, the ImageDataGenerator magnificence may be very useful. We can use this records generator in a couple of ways, relying at the approach we need to use [7]. Here we're the use of flow_from_directory approach [13]. This approach takes a course to the listing which incorporates the pix and the augmentation parameters as proven in discern 12.

First, we want to import the libraries required to for records generator. Then we create a records generator with photograph augmentation.

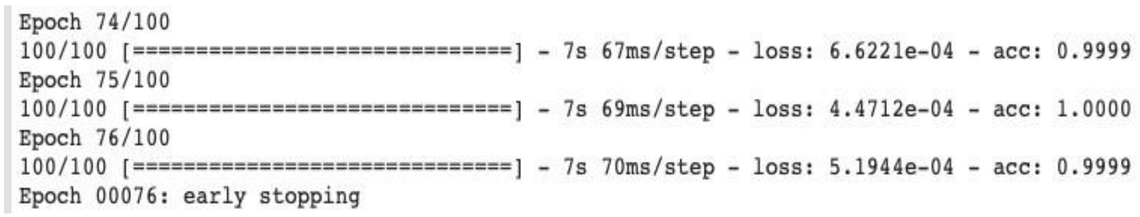


```
From keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator(rescale=1./255)
train_generator = train_datagen.flow_from_directory(
        # This is the target directory
        image_path, color_mode='rgb',
        # All images will be resized to 128x128
        target_size=(128,128), batch_size=15,
        #Since we use binary_crossentropy loss,we need binary labels
        )

# simple early stopping
es = [EarlyStopping(monitor='loss', mode='auto', min_delta=0.0001,
      patience=10, verbose=1)]

test_datagen = ImageDataGenerator(rescale=1./255)
```

We used the flow_from_directory approach from the ImageDataGenerator magnificence of keras.preprocessing.photograph library. The augmentation of the photograph is furnished as a controversy to the flow_from_directory approach [7].

The parameters of the approach are as follows: route,  color_mode, target_size, batch_size [13].  The route argument is wanted to specify the route of the photograph, color_mode is for specifying the shadeation mode of an enter photograph. The target_size parameter is to set the dimensions of the output photograph, and the batch_size parameter is to specify the range of photographs in line with batch going to be processed.

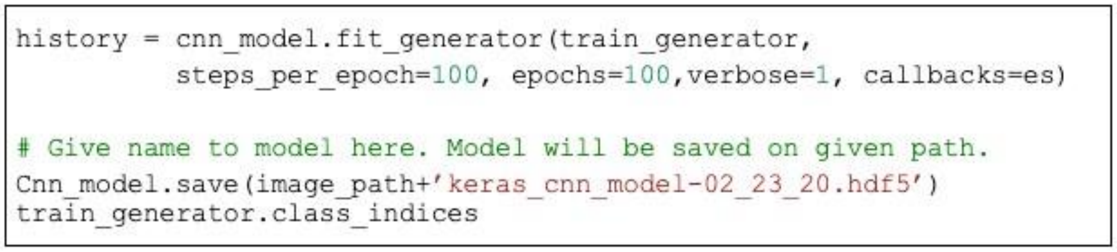
The epoch is like an iteration, so the range of epochs we outline is the range of instances the version will cycle via the data [3]. The more the range of epochs, the greater our version will improve, however as much as a sure factor only, then the version will forestall enhancing throughout every epoch [7]. That's why we're using 'EarlyStopping(es).' So that schooling stops if there may be no full-size development in few new epochs [7], as proven under in Figure 12.



```
Epoch 74/100
100/100 [==============================] - 7s 67ms/step - loss: 6.6221e-04 - acc: 0.9999
Epoch 75/100
100/100 [==============================] - 7s 69ms/step - loss: 4.4712e-04 - acc: 1.0000
Epoch 76/100
100/100 [==============================] - 7s 70ms/step - loss: 5.1944e-04 - acc: 0.9999
Epoch 00076: early stopping
```

Figure 12: Early stopping

In our model we defined ***100 steps/epoch*** for ***100 epochs***.  Also, in ***'EarlyStopping(es)'*** we defined ***min_delta=0.0001*** and ***patience=10*** *[13]*. This means in the last 10 epochs, if there is no minimum improvement of 0.0001 in accuracy then training will stop, as we don't want to further spend time on training if there is no improvement.



```
history = cnn_model.fit_generator(train_generator,
          steps_per_epoch=100, epochs=100,verbose=1, callbacks=es)

# Give name to model here. Model will be saved on given path.
Cnn_model.save(image_path+'keras_cnn_model-02_23_20.hdf5')
train_generator.class_indices
```

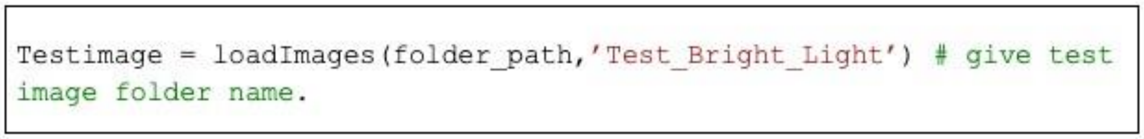
When the model is created, the ***'fit_generator()'*** method is used with the following parameters: train_generator, steps_per_epoch, epochs. Then the ***fit_generator*** method is run to fit the created model [7].

S***ave*** method is used to save the trained model, as this model is used for prediction.

This is the reason why ***ImageDataGenerator*** is an easy way to load all images from the specific path and process them as per the augmentation parameter in batches for image classification application.

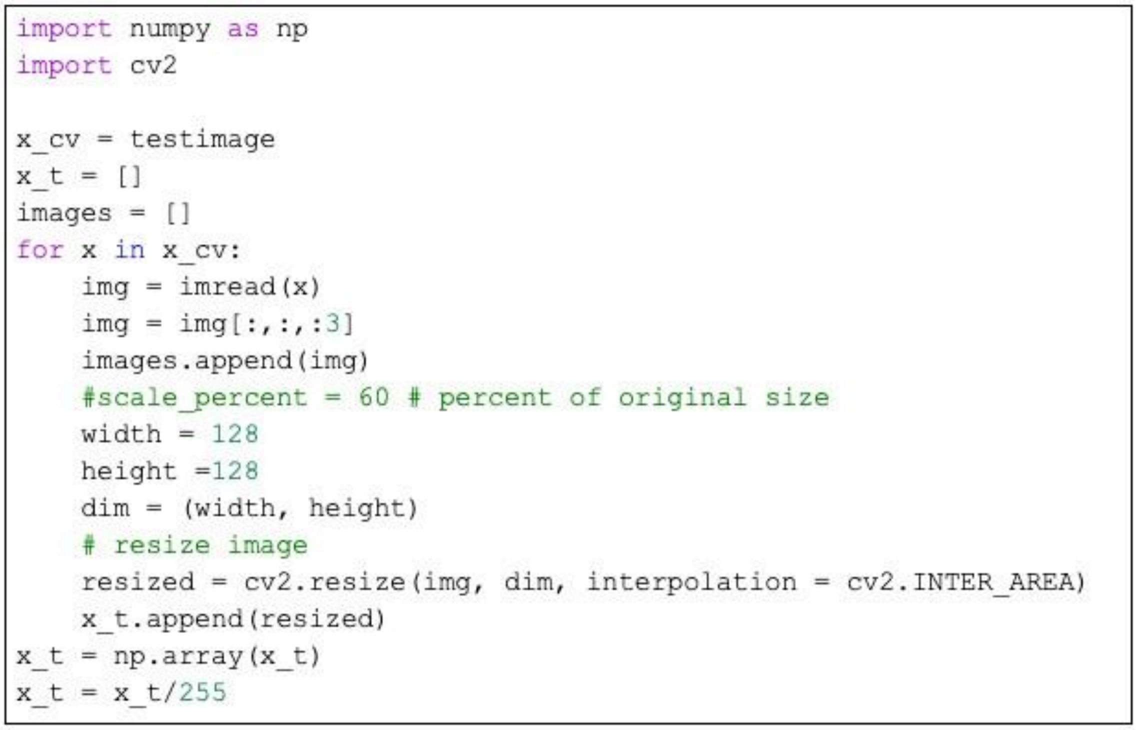
### 3.3.3 Prediction

Now, let's load the images using the previous ***loadImage()*** function for tests or prediction.



```
Testimage = loadImages(folder_path,'Test_Bright_Light') # give test
image folder name.
```

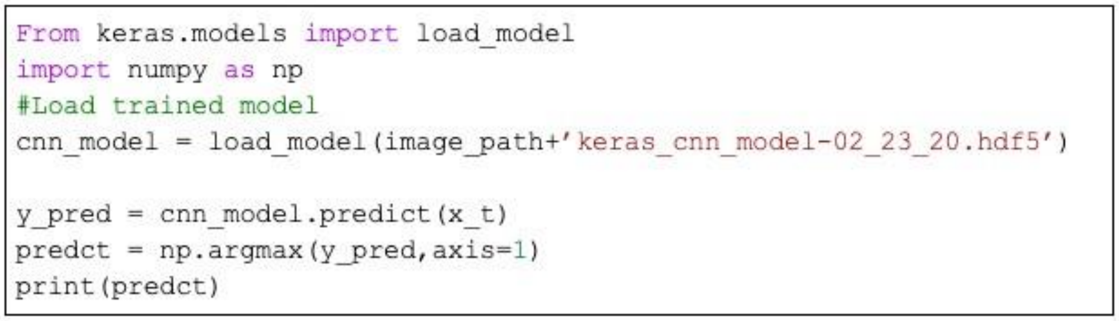
Original images should be resized before giving them to a trained model for prediction. By using the CV2 computer vision library we can resize all test images to 128X128. Then we need to append all images to one array using the NumPy library.



```
import numpy as np
import cv2

x_cv = testimage
x_t = []
images = []
for x in x_cv:
    img = imread(x)
    img = img[:,:,:3]
    images.append(img)
    #scale_percent = 60 # percent of original size
    width = 128
    height =128
    dim = (width, height)
    # resize image
    resized = cv2.resize(img, dim, interpolation = cv2.INTER_AREA)
    x_t.append(resized)
x_t = np.array(x_t)
x_t = x_t/255
```

### 3.3.4 Load Trained Model for Prediction

Trained model is loaded by using the ***load_model*** method of the ***ImageDataGenerator*** class [7]. This will load a trained model from the respected location. This ***load_model*** method takes a path for the model and loads the trained model for prediction [7]. Then ***predict*** method is used to predict each test image using the given trained model [7].



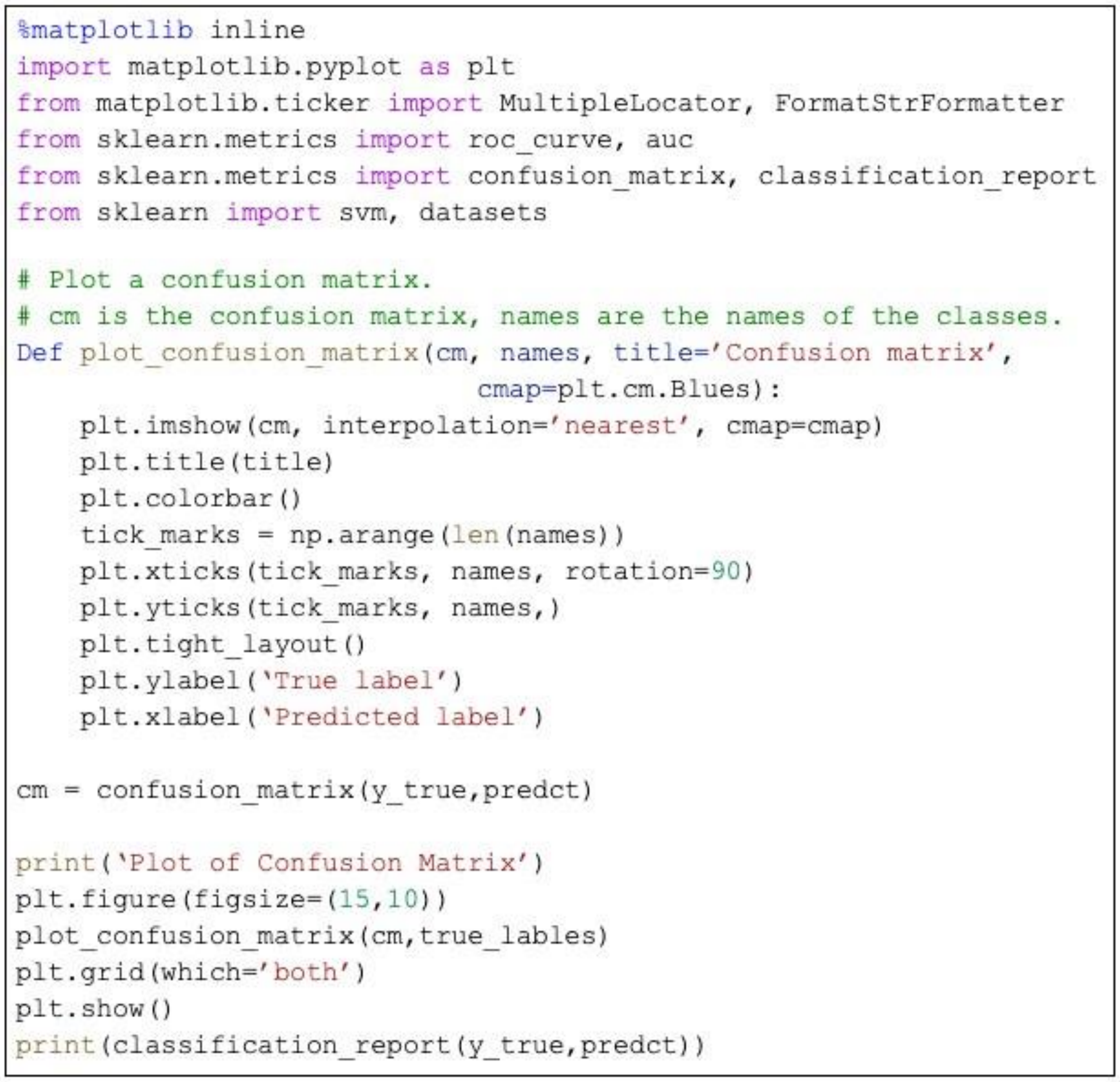
```
From keras.models import load_model
import numpy as np
#Load trained model
cnn_model = load_model(image_path+'keras_cnn_model-02_23_20.hdf5')

y_pred = cnn_model.predict(x_t)
predct = np.argmax(y_pred,axis=1)
print(predct)
```

### 3.3.5 Confusion Matrix

To summarize the overall performance and prediction end result of our type algorithm, we used a confusion matrix technique. As we've got extra than  type training, type accuracy by myself may be deceptive if we've got an unequal quantity of observations in lots of training withinside the dataset.

To get a higher concept of what our type version is predicting, calculating a confusion matrix and representing it visually is a great option. The general quantity of accurate and wrong prediction effects is summarized after which damaged down via way of means of elegance.

The Confusion_Matrix() characteristic will calculate the confusion matrix and go back the end result as an array, which indicates how our type version is careworn with which elegance while it makes predictions [11]. It offers perception now no longer most effective on what mistakes are being made via way of means of the classifier however extra specially the varieties of mistakes. We can print this array or plot it the usage of matplotlib.pyplot and interpret the effects.



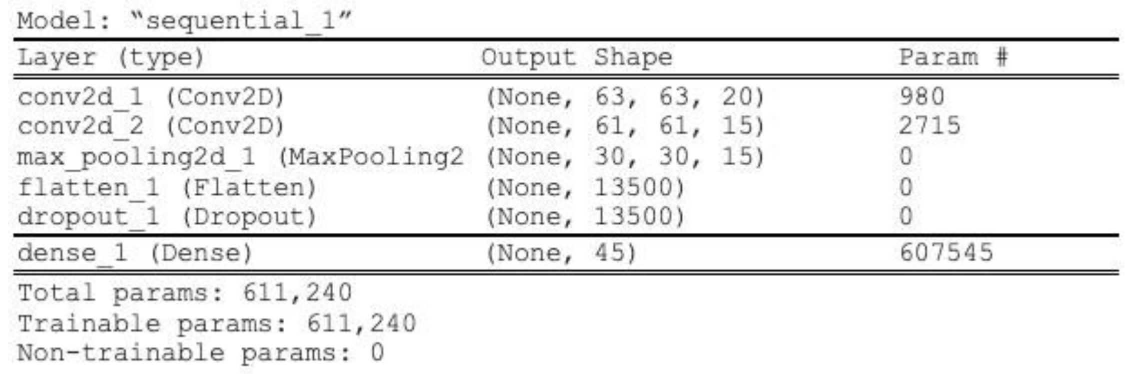
```
%matplotlib inline
import matplotlib.pyplot as plt
from matplotlib.ticker import MultipleLocator, FormatStrFormatter
from sklearn.metrics import roc_curve, auc
from sklearn.metrics import confusion_matrix, classification_report
from sklearn import svm, datasets

# Plot a confusion matrix.
# cm is the confusion matrix, names are the names of the classes.
Def plot_confusion_matrix(cm, names, title='Confusion matrix',
                             cmap=plt.cm.Blues):
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(names))
    plt.xticks(tick_marks, names, rotation=90)
    plt.yticks(tick_marks, names,)
    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')

cm = confusion_matrix(y_true,predct)

print('Plot of Confusion Matrix')
plt.figure(figsize=(15,10))
plot_confusion_matrix(cm,true_lables)
plt.grid(which='both')
plt.show()
print(classification_report(y_true,predct))
```

# Chapter 4

# RESULTS

## 4.1 Model Evaluation

This is the summary of the implemented model. As we are using a sequential type of model, summary title says that the model is *sequential_1*. We can get this model summary, as shown in Figure 13 below, by using *summary()* function at the end of the model.



```
Model: "sequential_1"
________________________________________________________________
Layer (type)                 Output Shape              Param #
================================================================
conv2d_1 (Conv2D)            (None, 63, 63, 20)        980
conv2d_2 (Conv2D)            (None, 61, 61, 15)        2715
max_pooling2d_1 (MaxPooling2 (None, 30, 30, 15)        0
flatten_1 (Flatten)          (None, 13500)             0
dropout_1 (Dropout)          (None, 13500)             0
________________________________________________________________
dense_1 (Dense)              (None, 45)                607545
================================================================
Total params: 611,240
Trainable params: 611,240
Non-trainable params: 0
```

Figure 13: Output of summary function

### 4.1.1 Ideal Condition

We provide a set of real color images, as shown in Figure 14 below, to the machine to predict colors. For that, we have created a set of color images using color code in the paint tool and saved them as a .jpg file. Later, we fed the images to a machine to learn and predict under ideal condition.

Figure 14: Ideal olive color

### 4.1.2 Different Light Condition

Here we're developing  unique (non- perfect) mild situations to check what the set of rules predicts from its preceding mastering. We will test the end result/prediction whilst there are unique mild situations, and what distinction is expected whilst we in comparison with the real colour (perfect) situation of the equal colour. Also, we're the use of our device learnings techniques to distinguish cases. For that, we're the use of the confusion matrix approach to examine the end result of various mild situations with perfect mild situations.

We are developing unique mild situations as follows:
1) Bright-mild situation,
2) Low-mild situation.

#### 4.1.2.1 Bright mild situation

Bright mild situation is whilst a colour (coloured photograph here) is uncovered to vivid mild, and the photograph is captured and supplied to the set of rules to expect the colour primarily based totally at the preceding mastering primarily based totally on perfect situation.

In our test, we're thinking about morning mild as a vivid mild scenario. We uncovered forty-five perfect colour color photographs to this vivid mild. After that, we once more captured all forty-five photographs, that are now below vivid mild situation.

Now we've our version educated below perfect situation. We fed the forty-five captured photographs as an enter to the educated version. Here are the expected effects proven withinside the confusion matrix in determine 15.

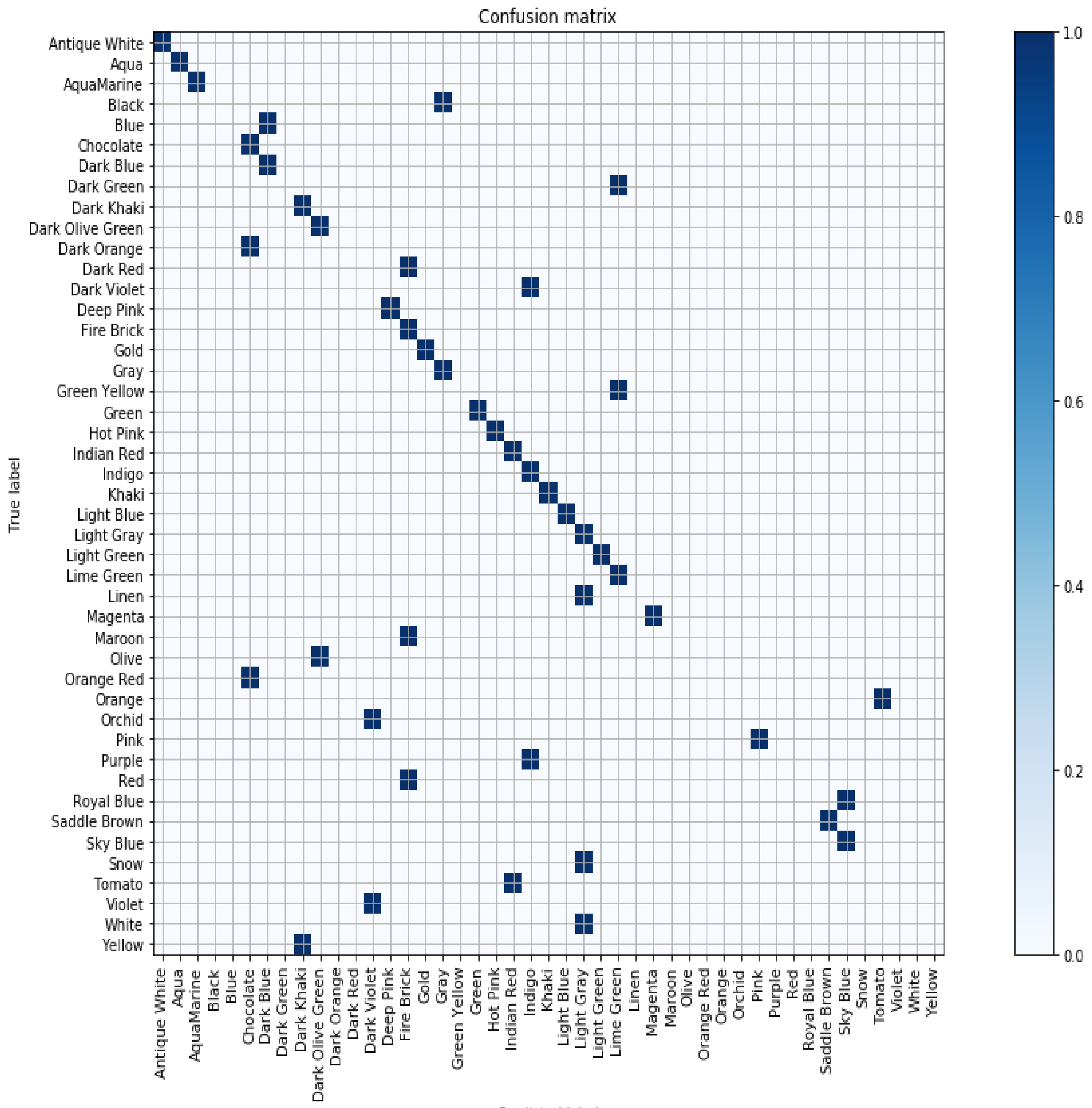
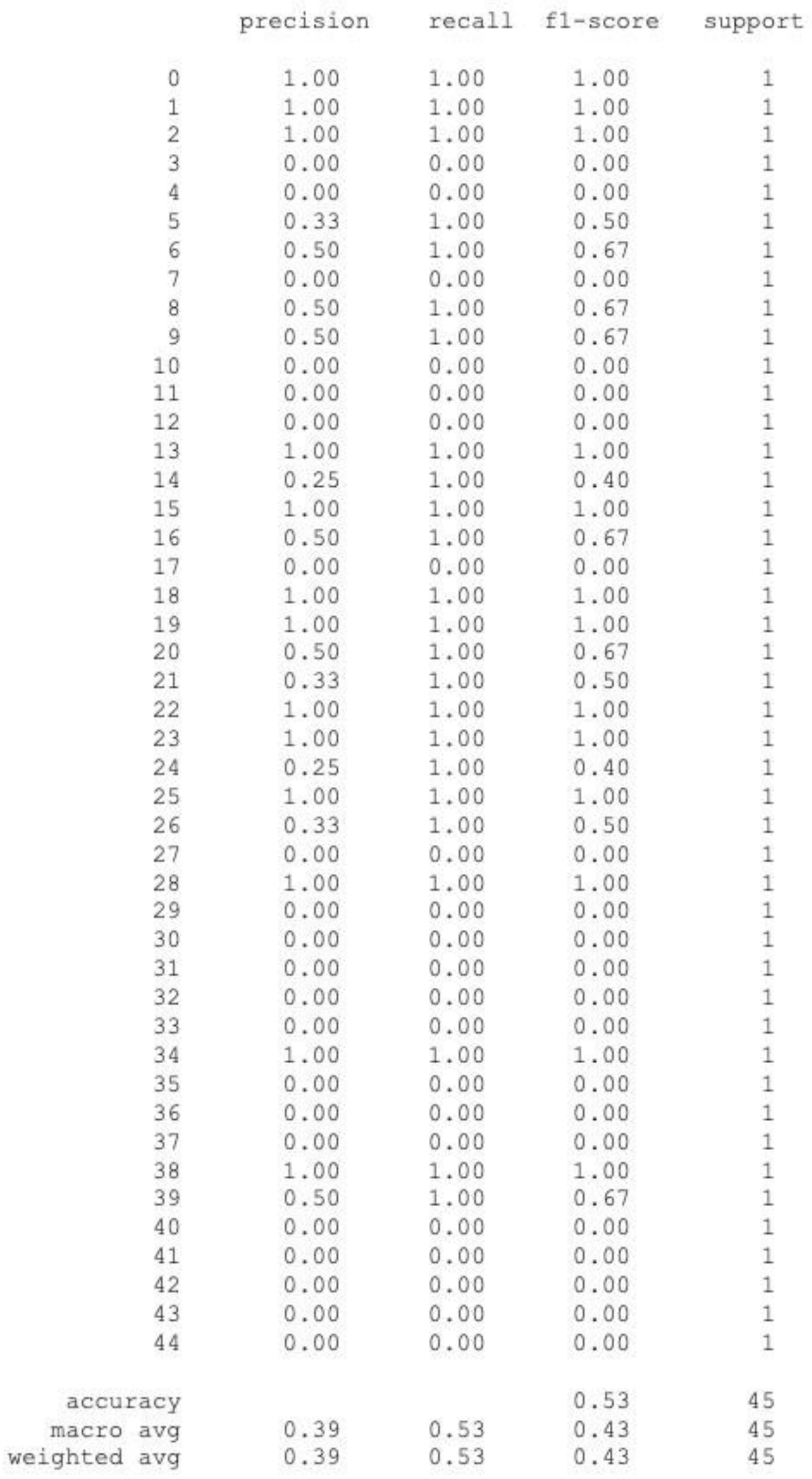


Figure 15: Confusion Matrix – Bright light condition

In this confusion matrix, we can see that the algorithm confused between the ideal condition and bright light condition images. We are expecting that different light conditions will change the prediction. The overall accuracy report is shown below.

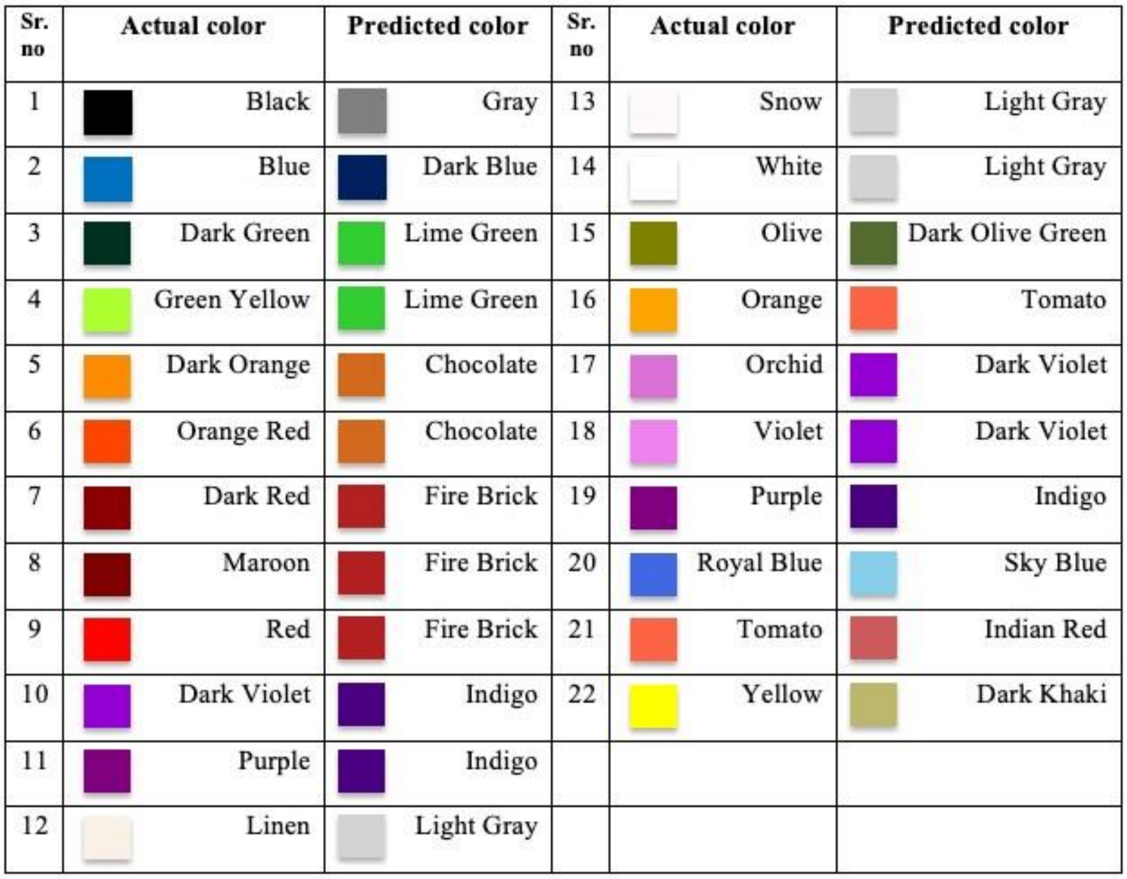
**Classification report:**



```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         1
           1       1.00      1.00      1.00         1
           2       1.00      1.00      1.00         1
           3       0.00      0.00      0.00         1
           4       0.00      0.00      0.00         1
           5       0.33      1.00      0.50         1
           6       0.50      1.00      0.67         1
           7       0.00      0.00      0.00         1
           8       0.50      1.00      0.67         1
           9       0.50      1.00      0.67         1
          10       0.00      0.00      0.00         1
          11       0.00      0.00      0.00         1
          12       0.00      0.00      0.00         1
          13       1.00      1.00      1.00         1
          14       0.25      1.00      0.40         1
          15       1.00      1.00      1.00         1
          16       0.50      1.00      0.67         1
          17       0.00      0.00      0.00         1
          18       1.00      1.00      1.00         1
          19       1.00      1.00      1.00         1
          20       0.50      1.00      0.67         1
          21       0.33      1.00      0.50         1
          22       1.00      1.00      1.00         1
          23       1.00      1.00      1.00         1
          24       0.25      1.00      0.40         1
          25       1.00      1.00      1.00         1
          26       0.33      1.00      0.50         1
          27       0.00      0.00      0.00         1
          28       1.00      1.00      1.00         1
          29       0.00      0.00      0.00         1
          30       0.00      0.00      0.00         1
          31       0.00      0.00      0.00         1
          32       0.00      0.00      0.00         1
          33       0.00      0.00      0.00         1
          34       1.00      1.00      1.00         1
          35       0.00      0.00      0.00         1
          36       0.00      0.00      0.00         1
          37       0.00      0.00      0.00         1
          38       1.00      1.00      1.00         1
          39       0.50      1.00      0.67         1
          40       0.00      0.00      0.00         1
          41       0.00      0.00      0.00         1
          42       0.00      0.00      0.00         1
          43       0.00      0.00      0.00         1
          44       0.00      0.00      0.00         1

    accuracy                           0.53        45
   macro avg       0.39      0.53      0.43        45
weighted avg       0.39      0.53      0.43        45
```

The above classification report gives accuracy in percentage. Here the accuracy is 53%. It means 47% of the time, predicated wrong under bright light conditions. You can look at the confusion matrix to see which color is predicted wrong. This also shows in the summary Table 1 below.

**Confusion matrix summary**



| Sr. no | Actual color | Predicted color | Sr. no | Actual color | Predicted color |
|---|---|---|---|---|---|
| 1 | ⬛ Black | ⬜ Gray | 13 | ⬜ Snow | ⬜ Light Gray |
| 2 | 🟦 Blue | 🟦 Dark Blue | 14 | ⬜ White | ⬜ Light Gray |
| 3 | 🟩 Dark Green | 🟩 Lime Green | 15 | 🟩 Olive | 🟩 Dark Olive Green |
| 4 | 🟩 Green Yellow | 🟩 Lime Green | 16 | 🟧 Orange | 🟥 Tomato |
| 5 | 🟧 Dark Orange | 🟫 Chocolate | 17 | 🟪 Orchid | 🟪 Dark Violet |
| 6 | 🟧 Orange Red | 🟫 Chocolate | 18 | 🟪 Violet | 🟪 Dark Violet |
| 7 | 🟥 Dark Red | 🟥 Fire Brick | 19 | 🟪 Purple | 🟪 Indigo |
| 8 | 🟥 Maroon | 🟥 Fire Brick | 20 | 🟦 Royal Blue | 🟦 Sky Blue |
| 9 | 🟥 Red | 🟥 Fire Brick | 21 | 🟥 Tomato | 🟥 Indian Red |
| 10 | 🟪 Dark Violet | 🟪 Indigo | 22 | 🟨 Yellow | 🟫 Dark Khaki |
| 11 | 🟪 Purple | 🟪 Indigo | | | |
| 12 | ⬜ Linen | ⬜ Light Gray | | | |

Table 1: Confusion Matrix Summary – Bright light condition

This precis offers the general assessment with the real shadeation and what the set of rules predicts the given brilliant mild circumstance photo. Let’s speak some examples from Table 1. It seems like the set of rules expected grey shadeation rather than black, and lime inexperienced rather than darkish inexperienced.

Also, we will take a look at that the set of rules made different mistakes. The set of rules expected the identical shadeation for 3 one of a kind colorings of the identical color. Linen, snow, white shadeation from white color all were given expected as a mild grey shadeation. Orchid and violet were given expected as darkish violet. One extra instance of device’s confusion is, for darkish violet and purple, the device expected indigo shadeation.

### 4.1.2.2 Low mild circumstance

Low mild circumstance is while a shadeation (coloured photo here) is uncovered to Low mild/Low-depth mild, and device imaginative and prescient captures this and gives it to the set of rules to are expecting the shadeation primarily based totally on its preceding studying primarily based totally on perfect conditions.

So, in our test, we're thinking about indoor night mild as a low mild scenario. Considering night
five pm indoor mild as a low mild circumstance, we uncovered forty-five Ideal shadeation color pix to this low mild.
After that, we once more captured all forty-five pix, which might be now below a low mild circumstance.

We have our version skilled below perfect conditions. We fed this forty-five captured pix as an input. The expected effects are proven withinside the confusion matrix in determine 16.

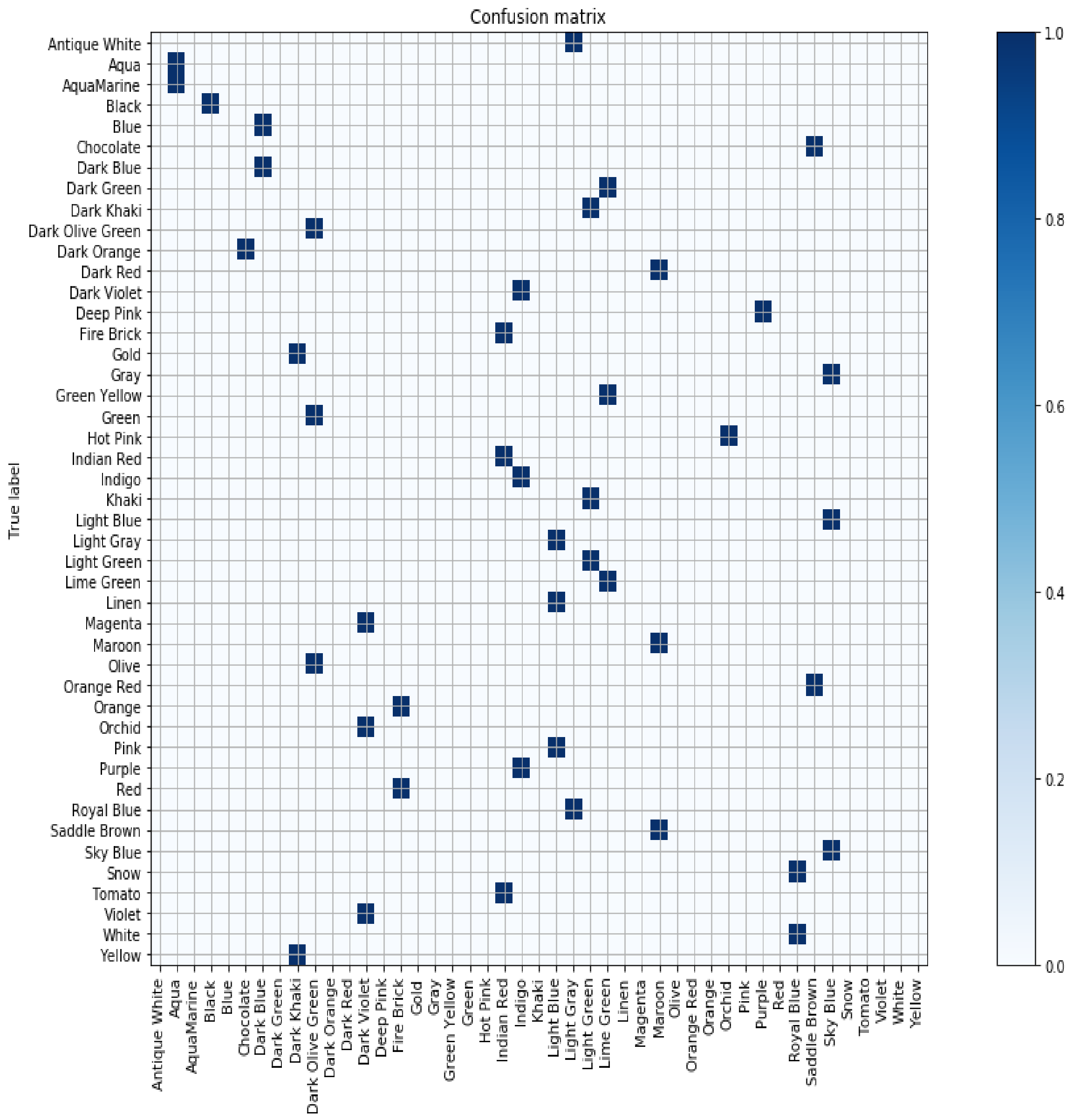
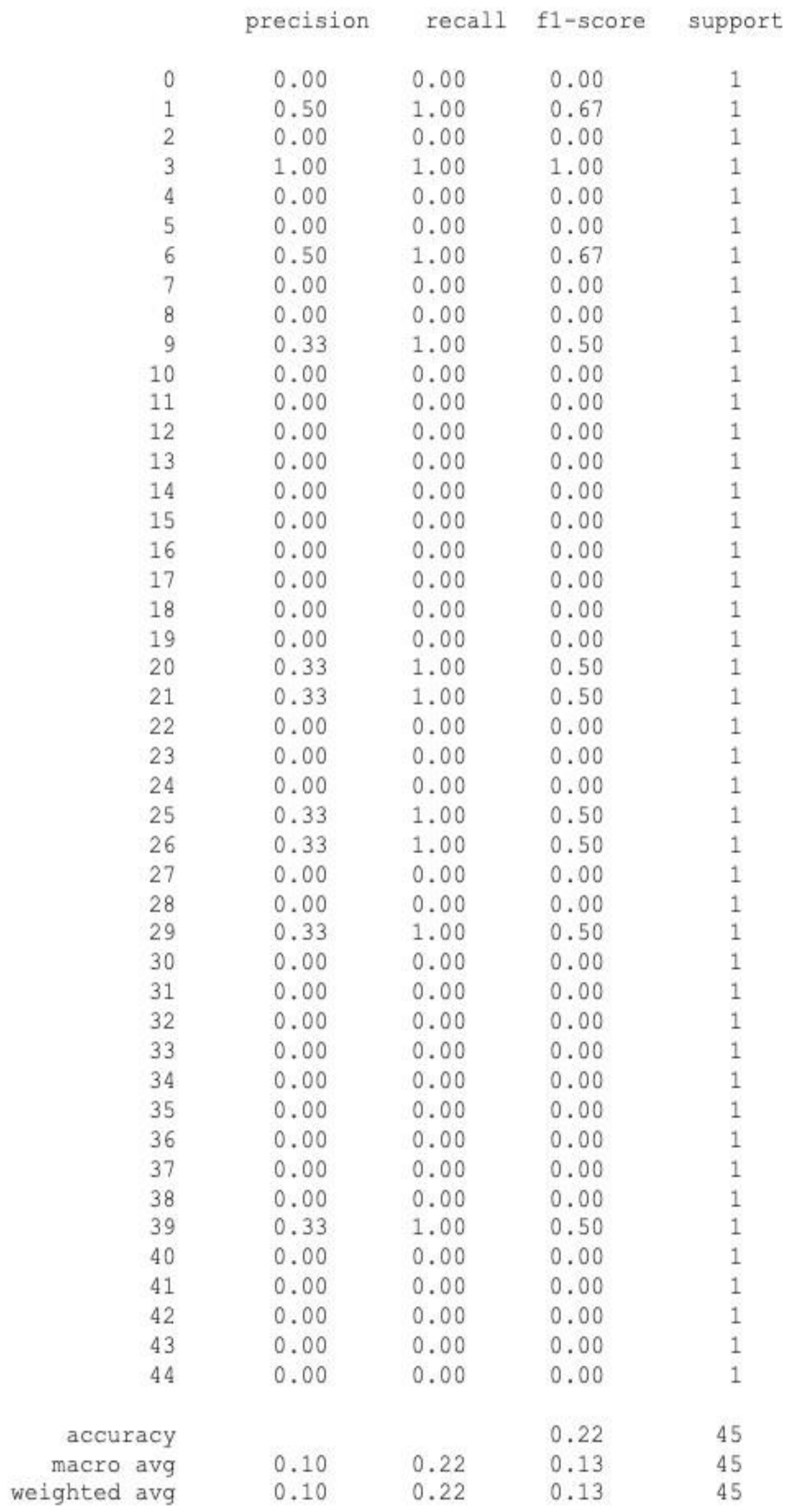


Figure 16: Confusion Matrix – Low light condition

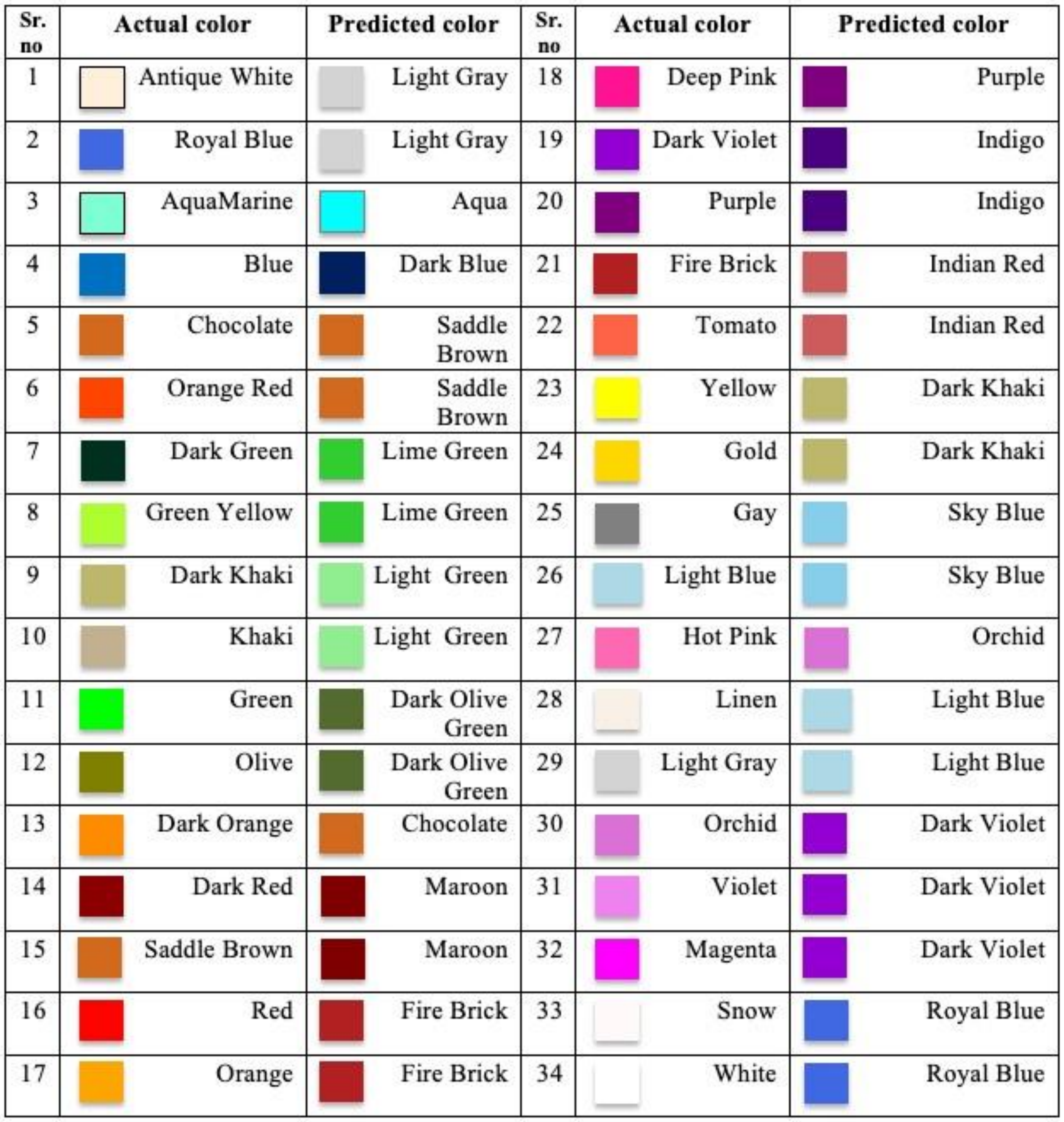
In this confusion matrix, we can see that the machine confused with the ideal condition and low light condition images. In low light, data looks scattered more than the previous non-ideal condition. It means the accuracy is less here. Let’s see the overall accuracy report, which is shown below.

**Classification report:**



```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         1
           1       0.50      1.00      0.67         1
           2       0.00      0.00      0.00         1
           3       1.00      1.00      1.00         1
           4       0.00      0.00      0.00         1
           5       0.00      0.00      0.00         1
           6       0.50      1.00      0.67         1
           7       0.00      0.00      0.00         1
           8       0.00      0.00      0.00         1
           9       0.33      1.00      0.50         1
          10       0.00      0.00      0.00         1
          11       0.00      0.00      0.00         1
          12       0.00      0.00      0.00         1
          13       0.00      0.00      0.00         1
          14       0.00      0.00      0.00         1
          15       0.00      0.00      0.00         1
          16       0.00      0.00      0.00         1
          17       0.00      0.00      0.00         1
          18       0.00      0.00      0.00         1
          19       0.00      0.00      0.00         1
          20       0.33      1.00      0.50         1
          21       0.33      1.00      0.50         1
          22       0.00      0.00      0.00         1
          23       0.00      0.00      0.00         1
          24       0.00      0.00      0.00         1
          25       0.33      1.00      0.50         1
          26       0.33      1.00      0.50         1
          27       0.00      0.00      0.00         1
          28       0.00      0.00      0.00         1
          29       0.33      1.00      0.50         1
          30       0.00      0.00      0.00         1
          31       0.00      0.00      0.00         1
          32       0.00      0.00      0.00         1
          33       0.00      0.00      0.00         1
          34       0.00      0.00      0.00         1
          35       0.00      0.00      0.00         1
          36       0.00      0.00      0.00         1
          37       0.00      0.00      0.00         1
          38       0.00      0.00      0.00         1
          39       0.33      1.00      0.50         1
          40       0.00      0.00      0.00         1
          41       0.00      0.00      0.00         1
          42       0.00      0.00      0.00         1
          43       0.00      0.00      0.00         1
          44       0.00      0.00      0.00         1

    accuracy                           0.22        45
   macro avg       0.10      0.22      0.13        45
weighted avg       0.10      0.22      0.13        45
```

The above classification report gives the accuracy in percent. Here the accuracy is 22%. It means 88% of the time, predicated wrong under low light conditions. You can look at the confusion matrix to see which color is predicted wrong. This is also show in the summary Table 2 below. **Confusion matrix summary.**



| Sr. no | Actual color | Predicted color | Sr. no | Actual color | Predicted color |
|---|---|---|---|---|---|
| 1 | ⬜ Antique White | ⬜ Light Gray | 18 | 🟪 Deep Pink | 🟪 Purple |
| 2 | 🟦 Royal Blue | ⬜ Light Gray | 19 | 🟪 Dark Violet | 🟪 Indigo |
| 3 | 🟩 AquaMarine | 🟦 Aqua | 20 | 🟪 Purple | 🟪 Indigo |
| 4 | 🟦 Blue | 🟦 Dark Blue | 21 | 🟥 Fire Brick | 🟥 Indian Red |
| 5 | 🟧 Chocolate | 🟧 Saddle Brown | 22 | 🟧 Tomato | 🟥 Indian Red |
| 6 | 🟧 Orange Red | 🟧 Saddle Brown | 23 | 🟨 Yellow | 🟨 Dark Khaki |
| 7 | 🟩 Dark Green | 🟩 Lime Green | 24 | 🟨 Gold | 🟨 Dark Khaki |
| 8 | 🟩 Green Yellow | 🟩 Lime Green | 25 | ⬛ Gay | 🟦 Sky Blue |
| 9 | 🟨 Dark Khaki | 🟩 Light Green | 26 | 🟦 Light Blue | 🟦 Sky Blue |
| 10 | 🟫 Khaki | 🟩 Light Green | 27 | 🟪 Hot Pink | 🟪 Orchid |
| 11 | 🟩 Green | 🟩 Dark Olive Green | 28 | ⬜ Linen | 🟦 Light Blue |
| 12 | 🟩 Olive | 🟩 Dark Olive Green | 29 | ⬜ Light Gray | 🟦 Light Blue |
| 13 | 🟧 Dark Orange | 🟧 Chocolate | 30 | 🟪 Orchid | 🟪 Dark Violet |
| 14 | 🟥 Dark Red | 🟥 Maroon | 31 | 🟪 Violet | 🟪 Dark Violet |
| 15 | 🟫 Saddle Brown | 🟥 Maroon | 32 | 🟪 Magenta | 🟪 Dark Violet |
| 16 | 🟥 Red | 🟥 Fire Brick | 33 | ⬜ Snow | 🟦 Royal Blue |
| 17 | 🟧 Orange | 🟥 Fire Brick | 34 | ⬜ White | 🟦 Royal Blue |

Table 2: Confusion Matrix Summary – Low light condition

This precis above offers the general evaluation with the real/perfect shadeation and what the device predicts with the given low mild situation. It looks as if the device were given stressed with more than one colorings. Let’s speak some examples from Table 2. From this desk, maximum stressed colorings are shiny. For instance, mild blue changed into anticipated as sky blue. Then for mild grey and linen, the device is predicting mild blue. The olive shadeation is getting a anticipated darkish olive shadeation.

Also, the set of rules anticipated the identical shadeation for 3 exceptional colorings of the identical colouration. Orchid, violet, and magenta have been anticipated as darkish violet. One greater proper instance of device confusion is for red, and orange, the set of rules anticipated hearthplace brick shadeation for each of them. It appears that the set of rules is predicting darker colouration than the real shadeation in low mild conditions.

## Overall observation

Many times, we without problems apprehend the shadeation through simply seeing it. Sometimes we are able to wager the shadeation colouration own circle of relatives however can’t without problems inform which genuine shadeation it's miles. So, even for us to don't forget or to wager, we've a research desk in our mind, which allows us to perceive the shadeation. So, it's miles the identical with ML, we offer the device with a research desk for all shadeation and could permit the device learn. Next time whilst a skilled device receives the enter images, it makes use of the research desk.

So, right here we are able to see some examples to visualize—figures 17-20 display orange, mild blue, orchid, darkish violet perfect shadeation images, and additionally shiny and occasional mild situation images. You can see how an awful lot distinction withinside the real/perfect shadeation photo and the non-perfect images.

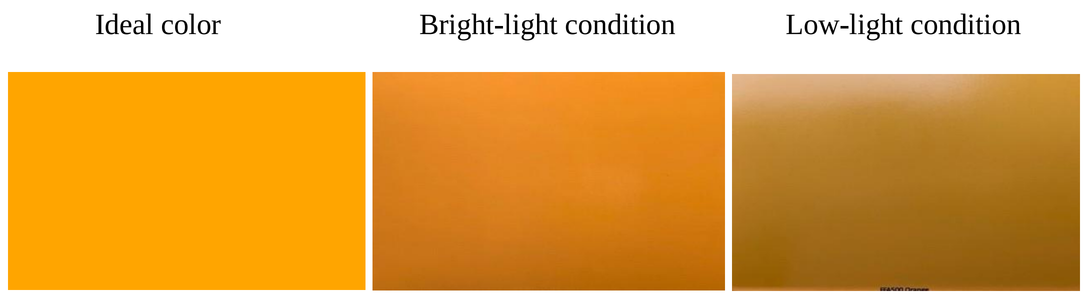


Figure 17: Orange color in all the conditions

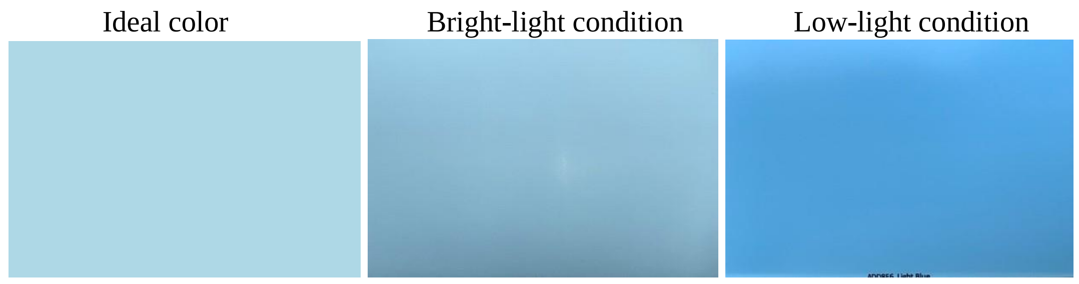


Figure 18: Light blue color in all the conditions

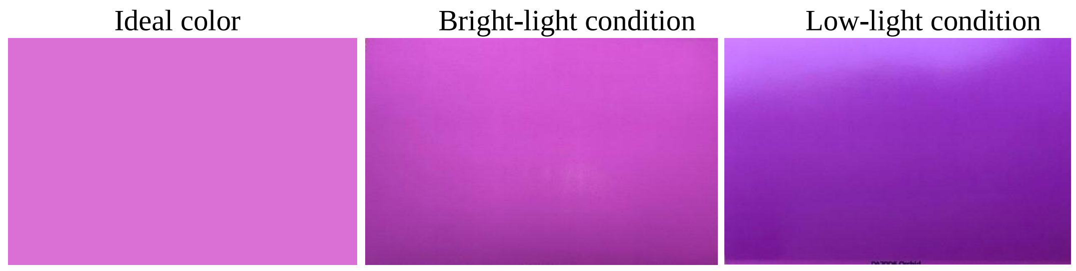


Figure 19: Orchid color in all the conditions

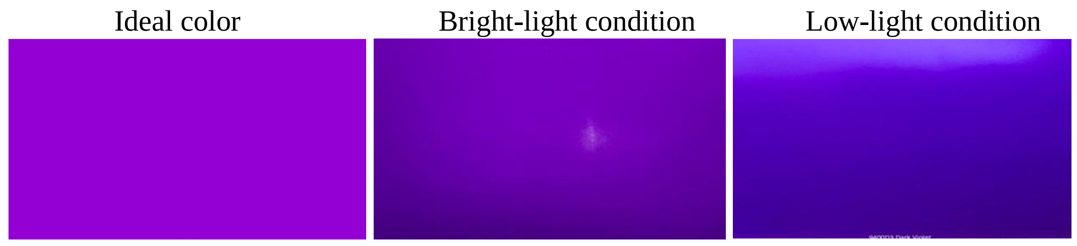


Figure 20: Dark violet color in all the conditions

From these figures, we can relate how the machine learning model failed to make the right prediction.

If you see the summary Tables 1-2, it says that orchid color in both non-ideal conditions was predicted as dark violet color. Because, in figure 19, the orchid color's bright light and low light condition image look like Figure 20's ideal color, which is dark violet.

# Chapter 5

# CONCLUSION AND FUTURE WORK

Using Machine Vision and Machine Learning algorithms, we are able to classify different color shades under different ideal and non-ideal light conditions like bright light, low/dark light, etc. Natural Daylight was considered to be a bright light source, and evening light/sunset light to be a low light source. This natural light source was needed in this project to create the non-ideal condition for a machine learning model. We trained and tested the machine learning model based on the data to classify different color shades.

The result of the low light condition gives us enough knowledge and visual output to conclude that in the low light condition, the predictions were darker than the actual color. Hence, the machine learning model predicts the relatively darker color shade of that actual color.

In addition, from the observation of the bright light condition, we can conclude that the machine learning model predicts a dark color relatively less dark and bright color relatively more bright shade of that actual color.

Based on the results of both non-ideal conditions, we conclude that different light conditions can alter the colors that machine vision captured and processed further, which can lead to differing color predictions.

This project area is expected to have many future works. We can extend this project work by adding a few other non-ideal light conditions like,

1) Sharp/direct focused light,
2) Direct sunlight/natural light,
3) Defused white/warm light
4 ) Using UV filters to the camera.

Also, we can change photo image material like

- Matt finished images
- Glossy finished image.

All we need is an excellent image data set to train any model. We can take this project work to the next level by using the same methodology, but just differently. It's like a recurring process for a few days to gather image data for creating your own image dataset. We can capture images on fixed intervals all day long for multiple days, each day, capture a new color, from sunrise to sunset at a fixed location. It's like a time-lapse of color images along with the time stamp, and the intensity of the light using hardware/sensor.

Create a data set of all these images, timestamp, and sensor data together. We can give this vast data set to the ML model to predict the color difference and, based on that, learn again using reinforcement learning and predict correct color based on different light conditions with light intensity. This extensive future work can be implemented to solve real-world problems using AI and ML.

# REFERENCES

[1] "Artificial neural network - Wikipedia ", [Online].

Available: https://en.wikipedia.org/wiki/Artificial_neural_network [Accessed Oct 10,2019]

[2] "Classification – Machine Learning", [Online].

Available: https://www.simplilearn.com/classification-machine-learning-tutorial

[Accessed Oct 10, 2019].

[3] Eijaz Allibhai, "Building A Deep Learning Model using Keras - Towards Data", [Online].

Available:https://towardsdatascience.com/building-a-deep-learning-model-using-keras-1548ca149d37 [Accessed Apr 14, 2020].

[4] Esposito, Floriana & Malerba, Donato. (2010). Editorial: Machine learning in computer vision.

Applied Artificial Intelligence: An International Journal. 15. 693-705. 10.1080/088395101317018546.

[5] “Find complex patterns - lynda.com”, [Online]. Available: https://www.lynda.com/DataScience-tutorials/Find-complex-patterns/601799/729679-4.html [Accessed Oct 10, 2019].

[6] Hunter Heidenreich, “what-are-the-types-of-machine-learning”, [Online].

Available: https://towardsdatascience.com/what-are-the-types-of-machine-learning-e2b9e5d1756f [Accessed Oct 10, 2020].

[7] Ilya Michlin, “Keras data generators and how to use them”, [Online].

Available:https://towardsdatascience.com/keras-data-generators-and-how-to-use-themb69129ed779c [Accessed Nov 04, 2019].

[8] J. Sainui and P. Pattanasatean, "Color Classification based on Pixel Intensity Values," 2018

19th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), Busan, 2018, pp. 302-306.

[9] James Currier, “What Makes Data Valuable: The Truth About Data Network Effects”

[Online]. Available: https://www.nfx.com/post/truth-about-data-network-effects/

[Accessed Oct 10, 2019].

[10] Jason Brownlee, “ 9 Application of Deep Learning for Computer Vision” [Online]. https://machinelearningmastery.com/applications-of-deep-learning-for-computer-vision/ [Accessed Apr 14, 2020]

[11] Jason Brownlee, “What is a Confusion Matrix in Machine Learning” [Online].

https://machinelearningmastery.com/confusion-matrix-machine-learning/

[Accessed Oct 10, 2019]

[12] Karan Bhanot, “Color Identification in Images”, [Online].

Available:https://towardsdatascience.com/color-identification-in-images-machine-learningapplication-b26e770c4c71 [Accessed Nov 04, 2019].

[13] “Keras Documentation - Image Processing”, [Online].

Available: https://keras.io/preprocessing/image/ [Accessed Mar 09, 2020].

[14] Marco Peixeiro, “Step-by-step Guide to Building Your Own Neural Network Form Scratch”, [Online]. Available: https://towardsdatascience.com/step-by-step-guide-to-building-your-ownneural-network-from-scratch-df64b1c5ab6e [Accessed Nov 04, 2019].

[15] Madhav, “Object Detection using Python OpenCV”, [Online]. Available:

https://circuitdigest.com/tutorial/object-detection-using-python-opencv [Accessed Apr 14, 2020].

[16] Mandy Sidana, " Intro to types of classification algorithms in Machine Learning", [Online].

Available:https://medium.com/@Mandysidana/machine-learning-types-of-classification-9497bd4f2e14 [Accessed Nov 04, 2019].

[17] "Machine Learning | Paldesk", [Online]. Available: https://www.paldesk.com/what-ismachine-learning/ [Accessed Apr 14, 2020].

[18]"ML Practicum: Image Classification", [Online]. Available:

https://developers.google.com/machine-learning/practica/image-classification/convolutionalneural-networks?hl=el [Accessed Nov 07, 2019].

[19] "Neural Networks and Convolutional Neural Networks Essential Training | LinkedIn Learning, formerly Lynda.com", [Online]. Available: https://www.linkedin.com/learning/neuralnetworks-and-convolutional-neural-networks-essential-training/activation-functions

[Accessed Nov 04, 2019].

[20] O'Shea, Keiron & Nash, Ryan. (2015). An Introduction to Convolutional Neural Networks.

ArXiv e-prints.

[21] Randerson, "Classify Hand-Written Digits Using. Python and Convolutional Neural Networks", [Online]. Available: https://itnext.io/classify-hand-written-digits-using-python-andconvolutional-neural-networks-26ccfc06b95c [Accessed Nov 04, 2019].

[22] "Use a neural network", [Online]. Available: https://www.lynda.com/Data-Sciencetutorials/Use-neural-network/601799/729675-4.html [Accessed Oct 10, 2019].

[23] Will Koehrsen, "Overfitting vs. Underfitting: A complete Example" [Online].

https://towardsdatascience.com/overfitting-vs-underfitting-a-complete-example-d05dd7e19765 [Accessed Oct 10, 2019]