Capstone Projects

COMPUTER HARDWARE/ SOFTWARE PRACTICAL FILE (COCSC19)

BY:
Sushant 202UCS1537
Deepak Kr Meena 2021UCS1544
Antriksh Bhandari 2021UCS1541
Kaushal Kumar 2021UCS1546

CAPSTONE PROJECT 1 : TINYML

Project Objective: Color classification using machine learning (ML) models involves building a system that can automatically classify objects based on their color..

Components Needed
Machine Learning Model (trained for color recognition using ML)

Modules used:
Data Collection and Preprocessing:
Collect a dataset of images containing various RGB colors and preprocess the images by resizing them to a consistent size and normalizing the pixel values.

Model Training and Evaluation:
Train a machine learning model for color detection using ML techniques, such as a convolutional neural network (CNN), on the preprocessed image dataset.
Evaluate the trained model's performance using validation data and metrics such as accuracy and loss.

Model Deployment:
Optimize the model for size and performance to ensure efficient execution on resource-constrained devices.

Real-Time Color Detection:
Implement algorithms to process the model's output and identify the dominant RGB color in each video frame.

User Interface:
Design and implement a user-friendly interface to interact with the color detection system.

Testing and Validation:
Conduct thorough testing of the color detection system to ensure its accuracy, reliability, and robustness in various lighting conditions and scenarios.

Tools used:
Python:
Python serves as the primary programming language for developing the color detection system. It is used for writing code to interface with the webcam, preprocess video frames, and perform color detection.

NumPy:
NumPy is a fundamental library for numerical computing in Python. It provides support for handling arrays and mathematical operations efficiently. In the project, NumPy is used for array manipulation, data preprocessing, and mathematical computations related to color detection.

IDE (Integrated Development Environment):
An IDE such as Visual Studio Code or PyCharm is used for writing, editing, and debugging Python code. IDEs offer features like syntax highlighting, code completion, and debugging tools that streamline the development process

APPLICATIONS

There are several real life applications of RGB colour detection:

1)Color Detection and Sorting:
In manufacturing and production lines, RGB color sensors are used to detect and sort items based on their colors. This is commonly seen in industries like food processing, packaging, and textile manufacturing.

2)Color Calibration in Imaging Devices:
Cameras, scanners, and other imaging devices often use RGB color sensors for color calibration. This ensures accurate reproduction of colors in photographs, prints, and digital images.

3)Display Calibration:
RGB color sensors are employed in display devices (monitors, TVs, smartphones) for calibrating and adjusting color settings. This helps achieve accurate and consistent color representation.

4)Color Measurement in Printing:
In the printing industry, RGB color sensors are used to measure and control color during the printing process. This ensures that printed materials match the intended color specifications.

5)Healthcare Applications:
Some healthcare devices use RGB color sensors for diagnostics. For example, urine analyzers might use color sensors to detect changes in the color of test strips.

6)Color Matching in Cosmetics:
In the cosmetics industry, RGB color sensors help match skin tones to the appropriate shades of makeup. This ensures that cosmetic products complement individual skin colors.

7)Automotive Paint Matching:
In the automotive industry, RGB color sensors are used to match paint colors during the manufacturing and repair of vehicles.

8)Ambient Light Adjustment in Smart Devices:
Smartphones, tablets, and other devices use RGB sensors to measure ambient light conditions. This information is then used to adjust the display brightness and color temperature for optimal viewing.

7)Home Automation:
In smart home systems, RGB color sensors can be used to control lighting based on the color and intensity of ambient light. For example, adjusting smart bulbs to mimic natural daylight.

8)Educational Tools:
RGB color sensors are often included in educational kits to teach students about color theory, light, and how sensors work.

CODE:

Importing Libraries

In [0]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import pathlib
import matplotlib.pyplot as plt
```

File display

In [0]:
```
# Importing Tensorflow
import tensorflow as tf

from tensorflow import keras
from tensorflow.keras import layers

print(tf.__version__)
```

```
2.2.0
```

In [0]:
```
!pip install git+https://github.com/tensorflow/docs # Use some functions from tensorflow_docs
```

```
Collecting git+https://github.com/tensorflow/docs
  Cloning https://github.com/tensorflow/docs to /tmp/pip-req-build-ufcwk31n
  Running command git clone -q https://github.com/tensorflow/docs /tmp/pip-req-build-ufcwk31n
Requirement already satisfied (use --upgrade to upgrade): tensorflow-docs===0.0.0f6a904b5c1e41e43dc29e43665fe1d4aca
929cf8- from git+https://github.com/tensorflow/docs in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: astor in /usr/local/lib/python3.6/dist-packages (from tensorflow-docs===0.0.0f6a904b
5c1e41e43dc29e43665fe1d4aca929cf8-) (0.8.1)
Requirement already satisfied: absl-py in /usr/local/lib/python3.6/dist-packages (from tensorflow-docs===0.0.0f6a90
4b5c1e41e43dc29e43665fe1d4aca929cf8-) (0.9.0)
Requirement already satisfied: protobuf in /usr/local/lib/python3.6/dist-packages (from tensorflow-docs===0.0.0f6a9
04b5c1e41e43dc29e43665fe1d4aca929cf8-) (3.10.0)
Requirement already satisfied: pyyaml in /usr/local/lib/python3.6/dist-packages (from tensorflow-docs===0.0.0f6a904
b5c1e41e43dc29e43665fe1d4aca929cf8-) (3.13)
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (from absl-py->tensorflow-docs===0.0.0
f6a904b5c1e41e43dc29e43665fe1d4aca929cf8-) (1.12.0)
Requirement already satisfied: setuptools in /usr/local/lib/python3.6/dist-packages (from protobuf->tensorflow-docs
===0.0.0f6a904b5c1e41e43dc29e43665fe1d4aca929cf8-) (47.1.1)
Building wheels for collected packages: tensorflow-docs
  Building wheel for tensorflow-docs (setup.py) ... done

  Created wheel for tensorflow-docs: filename=tensorflow_docs-0.0.0f6a904b5c1e41e43dc29e43665fe1d4aca929cf8_-cp36-n
one-any.whl size=114634 sha256=ae6322b51a29fdf2925c0d6d7cc4661dc6ef68da9c3eae0f37500a056fef4717
  Stored in directory: /tmp/pip-ephem-wheel-cache-nwvcme8e/wheels/eb/1b/35/fce87697be00d2fc63e0b4b395b0d9c7e391a10e
98d9a0d97f
Successfully built tensorflow-docs
```

In [0]:
```
import tensorflow_docs as tfdocs
import tensorflow_docs.plots
import tensorflow_docs.modeling
```

In [82]:
```
#plotly used for plotting Interactive Graphs
!pip install plotly
import plotly.offline as py
import plotly.express as px
import plotly.graph_objs as go
```

```
Requirement already satisfied: plotly in /usr/local/lib/python3.6/dist-packages (4.4.1)
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (from plotly) (1.12.0)
Requirement already satisfied: retrying>=1.3.3 in /usr/local/lib/python3.6/dist-packages (from plotly) (1.3.3)
```

Loading Dataset

In [0]:
```
dataset = pd.read_csv('final_data.csv')
dataset
```

Out[0]:

	red	green	blue	label
0	20	139	240	Blue
1	174	83	72	Brown
2	144	249	131	Green
3	168	25	156	Pink
4	30	182	136	Green
...	...	...	...	...
5000	15	117	39	Green
5001	107	78	245	Blue

5002	110	176	82	Green
5003	97	192	125	Green
5004	242	63	233	Pink

5005 rows × 4 columns

In [0]:
```
dataset.describe()
```

Out[0]:

	red	green	blue
count	5005.000000	5005.000000	5005.000000
mean	127.732468	126.623776	128.658741
std	73.494665	74.474670	73.715089
min	0.000000	0.000000	0.000000
25%	64.000000	61.000000	64.000000
50%	128.000000	127.000000	130.000000
75%	191.000000	191.000000	192.000000
max	254.000000	254.000000	254.000000

In [0]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5005 entries, 0 to 5004
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   red     5005 non-null   int64 
 1   green   5005 non-null   int64 
 2   blue    5005 non-null   int64 
 3   label   5005 non-null   object
dtypes: int64(3), object(1)
memory usage: 156.5+ KB
```

Pre Process, Clean the Data

File display

Eliminate Null if it exists

In [0]:
```
#To find whether null entries are present
dataset.isnull().sum()
```

Out[0]:
```
red      0
green    0
blue     0
label    0
dtype: int64
```

Converting Categorical Data of 'label` into Numerical

In [0]:
```
dataset.label.unique()
```

Out[0]:
```
array(['Blue', 'Brown', 'Green', 'Pink', 'Yellow', 'Orange', 'Purple',
       'Red', 'Grey', 'White', 'Black'], dtype=object)
```

In [0]:
```
# One-Hot-Encoding
dataset = pd.get_dummies(dataset, columns=['label'])
dataset
```

Out[0]:

	red	green	blue	label_Black	label_Blue	label_Brown	label_Green	label_Grey	label_Orange	label_Pink	label_Purple	l
0	20	139	240	0	1	0	0	0	0	0	0	
1	174	83	72	0	0	1	0	0	0	0	0	
2	144	249	131	0	0	0	1	0	0	0	0	
3	168	25	156	0	0	0	0	0	0	1	0	
4	30	182	136	0	0	0	1	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...

File display

5001	107	78	245	0	0	1	0	0	0	0	0	0
5002	110	176	82	0	0	0	0	1	0	0	0	0
5003	97	192	125	0	0	0	0	1	0	0	0	0
5004	242	63	233	0	0	0	0	0	0	0	1	0

5005 rows × 14 columns

Reorder

In [0]:
```
dataset = dataset[['red', 'green', 'blue', 'label_Red', 'label_Green', 'label_Blue', 'label_Yellow', 'label_Oran
dataset
```

Out[0]:

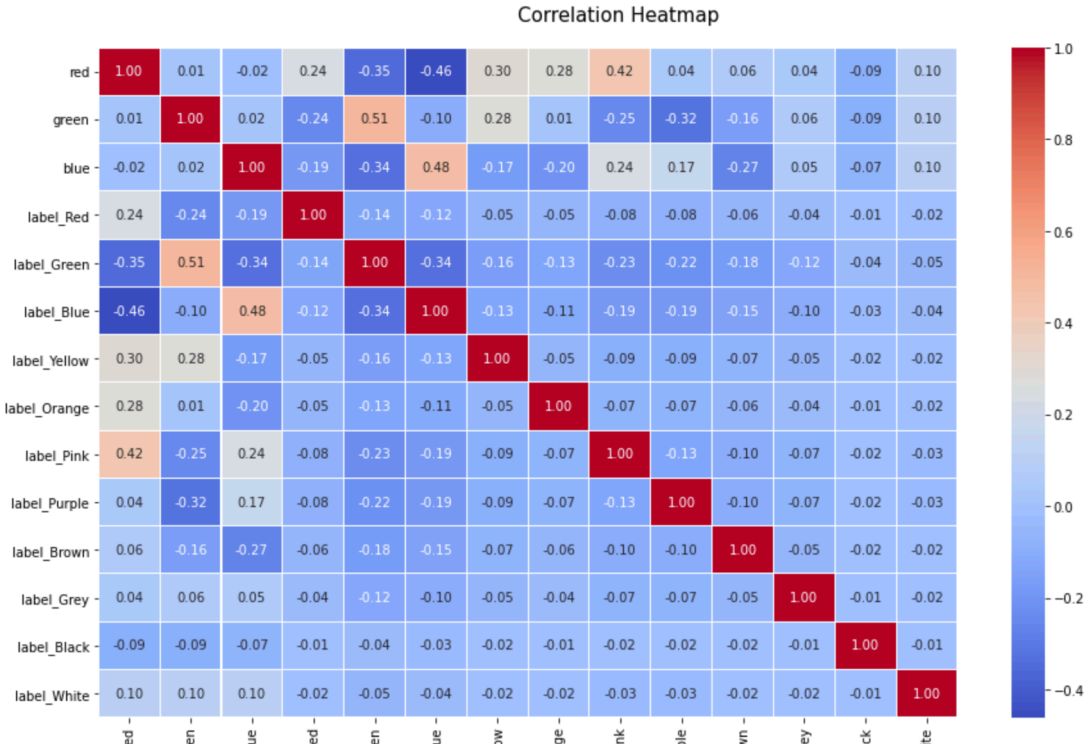
	red	green	blue	label_Red	label_Green	label_Blue	label_Yellow	label_Orange	label_Pink	label_Purple	label_Brown	l
0	20	139	240	0	0	1	0	0	0	0	0	
1	174	83	72	0	0	0	0	0	0	0	1	
2	144	249	131	0	1	0	0	0	0	0	0	
3	168	25	156	0	0	0	0	0	1	0	0	
4	30	182	136	0	1	0	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...
5000	15	117	39	0	1	0	0	0	0	0	0	
5001	107	78	245	0	0	1	0	0	0	0	0	
5002	110	176	82	0	1	0	0	0	0	0	0	
5003	97	192	125	0	1	0	0	0	0	0	0	
5004	242	63	233	0	0	0	0	0	1	0	0	

5005 rows × 14 columns

In [0]:

File display

```
# Correlation Matrix Heatmap
f, ax = plt.subplots(figsize=(15, 9))
corr = dataset.corr()
hm = sns.heatmap(round(corr,2), annot=True, ax=ax, cmap="coolwarm",fmt='.2f',
                 linewidths=.05)
f.subplots_adjust(top=0.93)
t= f.suptitle('Correlation Heatmap', fontsize=15)
```



In [0]:

File display

```
from keras import regularizers
model = keras.Sequential([
    layers.Dense(3, kernel_regularizer=regularizers.l2(0.001), activation='relu', input_shape=[len(train_dataset
    layers.Dense(24, kernel_regularizer=regularizers.l2(0.001), activation='relu'),
    layers.Dense(11)
  ])
```

Compile the model

In [0]:

```
optimizer = keras.optimizers.Adam(learning_rate=0.001)
loss_function = tf.keras.losses.CategoricalCrossentropy(from_logits=True)

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

In [0]:

```
model.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_12 (Dense)             (None, 3)                 12
_________________________________________________________________
dense_13 (Dense)             (None, 24)                96
_________________________________________________________________
dense_14 (Dense)             (None, 11)                275
=================================================================
Total params: 383
Trainable params: 383
Non-trainable params: 0
_________________________________________________________________
```

In [0]:

File display

```
history = model.fit(x=train_dataset, y=train_labels,
                    validation_split=0.3,
                    epochs=10001,
                    batch_size=48,
                    verbose=0,
                    callbacks=[tfdocs.modeling.EpochDots()],
                    shuffle=True)
```

```
Epoch: 0, accuracy:0.1167,  loss:15.2542,  val_accuracy:0.3020,  val_loss:5.9221,
.....
Epoch: 100, accuracy:0.7223,  loss:0.6345,  val_accuracy:0.7296,  val_loss:0.6229,
.....
Epoch: 200, accuracy:0.7702,  loss:0.5441,  val_accuracy:0.7587,  val_loss:0.5735,
.....
Epoch: 300, accuracy:0.7994,  loss:0.5037,  val_accuracy:0.7754,  val_loss:0.5366,
.....
Epoch: 400, accuracy:0.8308,  loss:0.4257,  val_accuracy:0.8195,  val_loss:0.4507,
.....
Epoch: 500, accuracy:0.8747,  loss:0.3342,  val_accuracy:0.8594,  val_loss:0.3548,
.....
Epoch: 600, accuracy:0.8758,  loss:0.3196,  val_accuracy:0.8561,  val_loss:0.3410,
.....
Epoch: 700, accuracy:0.8840,  loss:0.2965,  val_accuracy:0.8644,  val_loss:0.3406,
.....
Epoch: 800, accuracy:0.8844,  loss:0.3022,  val_accuracy:0.8536,  val_loss:0.3407,
.....
Epoch: 900, accuracy:0.8829,  loss:0.2927,  val_accuracy:0.8536,  val_loss:0.3377,
.....
Epoch: 1000, accuracy:0.8844,  loss:0.3019,  val_accuracy:0.8602,  val_loss:0.3305,
.....
Epoch: 1100, accuracy:0.8869,  loss:0.2853,  val_accuracy:0.8502,  val_loss:0.3727,
.....
Epoch: 1200, accuracy:0.8847,  loss:0.2937,  val_accuracy:0.8602,  val_loss:0.3296,
.....
Epoch: 1300, accuracy:0.8858,  loss:0.2918,  val_accuracy:0.8536,  val_loss:0.3519,
.....
Epoch: 1400, accuracy:0.8872,  loss:0.2870,  val_accuracy:0.8536,  val_loss:0.3577,
.....
Epoch: 1500, accuracy:0.8883,  loss:0.2856,  val_accuracy:0.8644,  val_loss:0.3491,
.....
Epoch: 1600, accuracy:0.8801,  loss:0.2876,  val_accuracy:0.8511,  val_loss:0.3418,
.....
Epoch: 1700, accuracy:0.8894,  loss:0.2811,  val_accuracy:0.8502,  val_loss:0.3794,
.....
```

In [29]:
File display

```
train_dataset = dataset.sample(frac=0.8, random_state=9) #train = 80%,  random_state = any int value means every
test_dataset = dataset.drop(train_dataset.index) #remove train_dataset from dataframe to get test_dataset
train_dataset
```

Out[29]:

	red	green	blue	label_Red	label_Green	label_Blue	label_Yellow	label_Orange	label_Pink	label_Purple	label_Brown	l
3319	81	53	243	0	0	1	0	0	0	0	0	
3450	254	118	250	0	0	0	0	0	1	0	0	
1942	127	53	69	0	0	0	0	0	0	0	1	
1803	162	4	242	0	0	0	0	0	0	1	0	
4583	13	57	85	0	0	1	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...
2768	91	197	240	0	0	1	0	0	0	0	0	
4239	167	88	12	0	0	0	0	0	0	0	1	
1745	87	237	62	0	1	0	0	0	0	0	0	
2291	190	132	64	0	0	0	0	0	0	0	1	
3566	57	4	55	0	0	0	0	0	0	1	0	

4004 rows × 14 columns

In [30]:

```
test_dataset
```

Out[30]:

	red	green	blue	label_Red	label_Green	label_Blue	label_Yellow	label_Orange	label_Pink	label_Purple	label_Brown	l
0	20	139	240	0	0	1	0	0	0	0	0	
3	168	25	156	0	0	0	0	0	1	0	0	
11	249	99	108	0	0	0	0	0	1	0	0	
12	24	161	177	0	0	1	0	0	0	0	0	
14	125	103	236	0	0	1	0	0	0	0	0	
...	...	...	...									
2768	91	197	240									
4239	167	88	12									
1745	87	237	62									
2291	190	132	64									
3566	57	4	55									

File display

4004 rows × 3 columns

In [33]:

```
test_labels = pd.DataFrame([test_dataset.pop(x) for x in ['label_Red', 'label_Green', 'label_Blue', 'label_Yello
test_labels
```

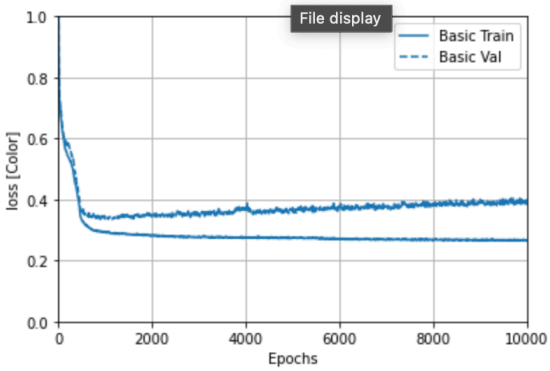
Out[33]:

	label_Red	label_Green	label_Blue	label_Yellow	label_Orange	label_Pink	label_Purple	label_Brown	label_Grey	label_B
0	0	0	1	0	0	0	0	0	0	
3	0	0	0	0	0	1	0	0	0	
11	0	0	0	0	0	1	0	0	0	
12	0	0	1	0	0	0	0	0	0	
14	0	0	1	0	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...
4988	0	1	0	0	0	0	0	0	0	
4993	0	1	0	0	0	0	0	0	0	
4994	0	0	1	0	0	0	0	0	0	
4996	0	0	0	0	1	0	0	0	0	
5001	0	0	1	0	0	0	0	0	0	

1001 rows × 11 columns

In [0]:
```
plotter.plot({'Basic': history}, metric = "loss")
plt.ylim([0, 1])
plt.ylabel('loss [Color]')
```

Out[0]:
```
Text(0, 0.5, 'loss [Color]')
```



In [0]:
```
#Install PyDrive library into Google Colab notebook
!pip install -U -q PyDrive
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from google.colab import auth
from oauth2client.client import GoogleCredentials
```

In [0]:
```
#Authenticate and create the PyDrive client
auth.authenticate_user()
gauth = GoogleAuth()
gauth.credentials = GoogleCredentials.get_application_default()
drive = GoogleDrive(gauth)
```

In [0]:
```
!pip install pyyaml h5py  # Required to save models in HDF5 format
```

```
Requirement already satisfied: pyyaml in /usr/local/lib/python3.6/dist-packages (3.13)
Requirement already satisfied: h5py in /usr/local/lib/python3.6/dist-packages (2.10.0)
Requirement already satisfied: numpy>=1.7 in /usr/local/lib/python3.6/dist-packages (from h5py) (1.18.5)
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (from h5py) (1.12.0)
```

In [0]:
```
# Save Model or weights on google drive and create on Colab directory in Google Drive

# Save the entire model to a HDF5 file.
# The '.h5' extension indicates that the model should be saved to HDF5.
model.save('colormodel_acu_89.h5')
model_file = drive.CreateFile({'title' : 'colormodel_acu_89.h5'}) # colormodel_acu_  .h5 is the name of the mode
model_file.SetContentFile('colormodel_acu_89.h5')
model_file.Upload()
```

Out[0]:
```
GoogleDriveFile({'id': '14Jxscgqc0bF34V7Nlh7dmqS9rL0EV-Yy'})
```

In [0]:
```
#Load model from google drive into colab
file_obj = drive.CreateFile({'id': '14Jxscgqc0bF34V7Nlh7dmqS9rL0EV-Yy'})
file_obj.GetContentFile('colormodel_trained_89.h5') #colormodel_trained.h5 is the name of model uploaded to cola
```

In [36]:
```
ls
```

```
adc.json  colormodel_trained_89.h5  drive/  final_data.csv  sample_data/
```

In [37]:
```
# Recreate the exact same model, including its weights and the optimizer
model = tf.keras.models.load_model('colormodel_trained_89.h5') #very important

# Show the model architecture
model.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_12 (Dense)             (None, 3)                 12
_________________________________________________________________
dense_13 (Dense)             (None, 24)                96
_________________________________________________________________
dense_14 (Dense)             (None, 11)                275
=================================================================
Total params: 383
Trainable params: 383
Non-trainable params: 0
```

In [49]:
File display

```
model.evaluate(x=train_dataset, y=train_labels)
```

```
126/126 [==============================] - 0s 1ms/step - loss: 0.2930 - accuracy: 0.8894
```

Out[49]:
```
[0.29301711916923523, 0.8893606662750244]
```

In [115…

```
from sklearn.metrics import confusion_matrix, classification_report
confusion_matrix_train = confusion_matrix(actual_encoded_train_labels, predicted_encoded_train_labels)
confusion_matrix_train
```

Out[115…

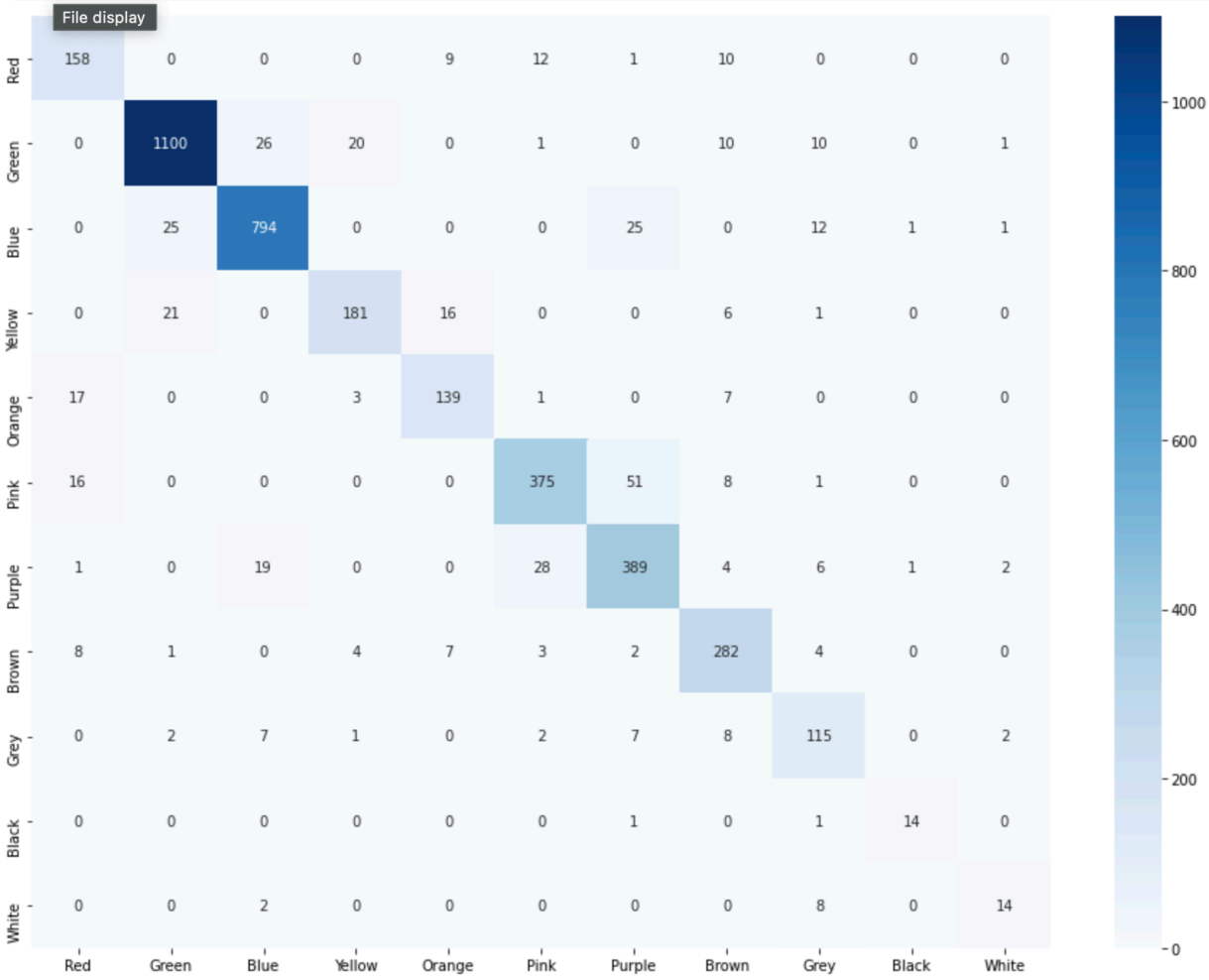
```
array([[ 158,    0,    0,    0,    9,   12,    1,   10,    0,    0,    0],
       [   0, 1100,   26,   20,    0,    1,    0,   10,   10,    0,    1],
       [   0,   25,  794,    0,    0,    0,   25,    0,   12,    1,    1],
       [   0,   21,    0,  181,   16,    0,    0,    6,    1,    0,    0],
       [  17,    0,    0,    3,  139,    1,    0,    7,    0,    0,    0],
       [  16,    0,    0,    0,    0,  375,   51,    8,    1,    0,    0],
       [   1,    0,   19,    0,    0,   28,  389,    4,    6,    1,    2],
       [   8,    1,    0,    4,    7,    3,    2,  282,    4,    0,    0],
       [   0,    2,    7,    1,    0,    2,    7,    8,  115,    0,    2],
       [   0,    0,    0,    0,    0,    0,    1,    0,    1,   14,    0],
       [   0,    0,    2,    0,    0,    0,    0,    0,    8,    0,   14]])
```

In [133…

```
f,ax = plt.subplots(figsize=(16,12))
categories = ['Red', 'Green', 'Blue', 'Yellow', 'Orange', 'Pink', 'Purple', 'Brown', 'Grey', 'Black', 'White']
sns.heatmap(confusion_matrix_train, annot=True, cmap='Blues', fmt='d',
            xticklabels = categories,
            yticklabels = categories)
plt.show()
```

```
plt.show()
```

File display



```
Epoch: 7800, accuracy:0.8940, loss:0.2690, val_accuracy:0.8669, val_loss:0.3790
```

In [103…
```
target_names = ['Red', 'Green', 'Blue', 'Yellow', 'Orange', 'Pink', 'Purple', 'Brown', 'Grey', 'Black', 'White']
print(classification_report(actual_encoded_train_labels, predicted_encoded_train_labels, target_names=target_nam
```

File display

```
              precision    recall  f1-score   support

         Red       0.79      0.83      0.81       190
       Green       0.96      0.94      0.95      1168
        Blue       0.94      0.93      0.93       858
      Yellow       0.87      0.80      0.83       225
      Orange       0.81      0.83      0.82       167
        Pink       0.89      0.83      0.86       451
      Purple       0.82      0.86      0.84       450
       Brown       0.84      0.91      0.87       311
        Grey       0.73      0.80      0.76       144
       Black       0.88      0.88      0.88        16
       White       0.70      0.58      0.64        24

    accuracy                           0.89      4004
   macro avg       0.84      0.84      0.84      4004
weighted avg       0.89      0.89      0.89      4004
```

```
Epoch: 8000, accuracy:0.8944, loss:0.2660, val_accuracy:0.8594, val_loss:0.3902,
```

.....

In [50]:
```
model.evaluate(x=test_dataset, y=test_labels)
```

```
32/32 [==============================] - 0s 1ms/step - loss: 0.4383 - accuracy: 0.8661
```

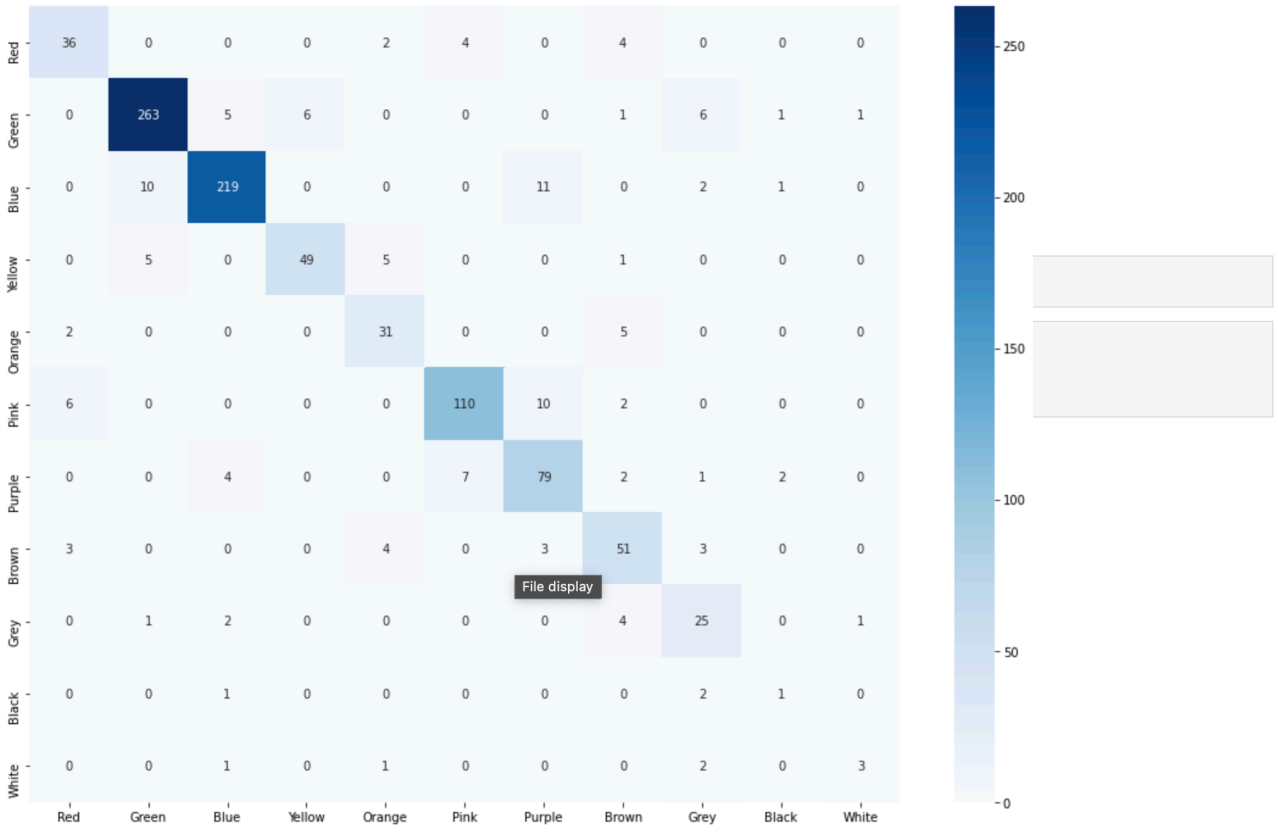
Out[50]:
```
[0.438273549079895, 0.8661338686943054]
```

In [116…
```
confusion_matrix_test = confusion_matrix(actual_encoded_test_labels, predicted_encoded_test_labels)
confusion_matrix_test
```

Out[116…
```
array([[ 36,   0,   0,   0,   2,   4,   0,   4,   0,   0,   0],
       [  0, 263,   5,   6,   0,   0,   0,   1,   6,   1,   1],
       [  0,  10, 219,   0,   0,   0,  11,   0,   2,   1,   0],
       [  0,   5,   0,  49,   5,   0,   0,   1,   0,   0,   0],
       [  2,   0,   0,   0,  31,   0,   0,   5,   0,   0,   0],
       [  6,   0,   0,   0,   0, 110,  10,   2,   0,   0,   0],
       [  0,   0,   4,   0,   0,   7,  79,   2,   1,   2,   0],
       [  3,   0,   0,   0,   4,   0,   3,  51,   3,   0,   0],
       [  0,   1,   2,   0,   0,   0,   0,   4,  25,   0,   1],
       [  0,   0,   1,   0,   0,   0,   0,   0,   2,   1,   0],
       [  0,   0,   1,   0,   1,   0,   0,   0,   2,   0,   3]])
```

In [127…
```
f,ax = plt.subplots(figsize=(16,12))
categories = ['Red', 'Green', 'Blue', 'Yellow', 'Orange', 'Pink', 'Purple', 'Brown', 'Grey', 'Black', 'White']
sns.heatmap(confusion_matrix_test, annot=True, cmap='Blues', fmt='d',
            xticklabels = categories,
            yticklabels = categories)
plt.show()
```

File display



In [117…

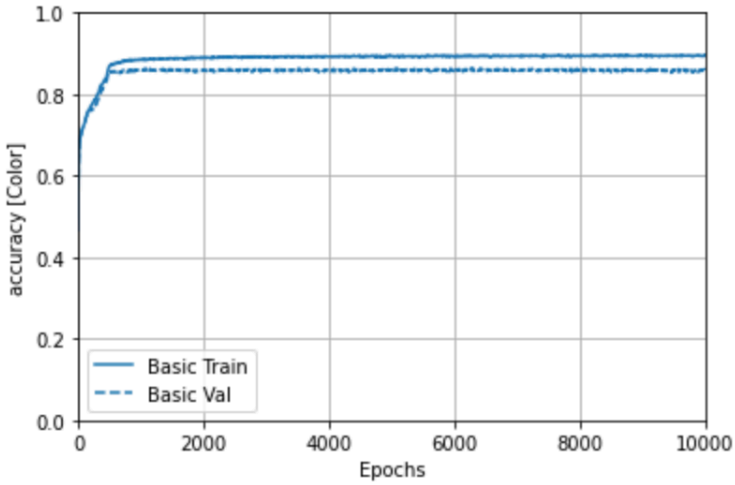
File display

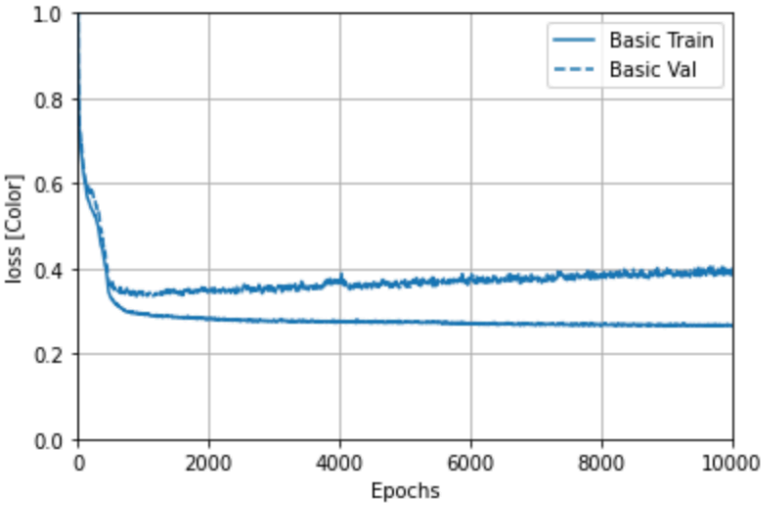
```
target_names = ['Red', 'Green', 'Blue', 'Yellow', 'Orange', 'Pink', 'Purple', 'Brown', 'Grey', 'Black', 'White']
print(classification_report(actual_encoded_test_labels, predicted_encoded_test_labels, target_names=target_names
```

```
              precision    recall  f1-score   support

         Red       0.77      0.78      0.77        46
       Green       0.94      0.93      0.94       283
        Blue       0.94      0.90      0.92       243
      Yellow       0.89      0.82      0.85        60
      Orange       0.72      0.82      0.77        38
        Pink       0.91      0.86      0.88       128
      Purple       0.77      0.83      0.80        95
       Brown       0.73      0.80      0.76        64
        Grey       0.61      0.76      0.68        33
       Black       0.20      0.25      0.22         4
       White       0.60      0.43      0.50         7

    accuracy                           0.87      1001
   macro avg       0.73      0.74      0.74      1001
weighted avg       0.87      0.87      0.87      1001
```





Accuracy for training set

	precision	recall	f1-score	support
Red	0.79	0.83	0.81	190
Green	0.96	0.94	0.95	1168
Blue	0.94	0.93	0.93	858
Yellow	0.87	0.80	0.83	225
Orange	0.81	0.83	0.82	167
Pink	0.89	0.83	0.86	451
Purple	0.82	0.86	0.84	450
Brown	0.84	0.91	0.87	311
Grey	0.73	0.80	0.76	144
Black	0.88	0.88	0.88	16
White	0.70	0.58	0.64	24
accuracy			0.89	4004
macro avg	0.84	0.84	0.84	4004
weighted avg	0.89	0.89	0.89	4004

Accuracy for test set

	precision	recall	f1-score	support
Red	0.77	0.78	0.77	46
Green	0.94	0.93	0.94	283
Blue	0.94	0.90	0.92	243
Yellow	0.89	0.82	0.85	60
Orange	0.72	0.82	0.77	38
Pink	0.91	0.86	0.88	128
Purple	0.77	0.83	0.80	95
Brown	0.73	0.80	0.76	64
Grey	0.61	0.76	0.68	33
Black	0.20	0.25	0.22	4
White	0.60	0.43	0.50	7
accuracy			0.87	1001
macro avg	0.73	0.74	0.74	1001
weighted avg	0.87	0.87	0.87	1001