



Faculty Of Engineering Ain Shams University

Name	ID
Motaz Adel Mohamed	1902229
mohamed Reda abdelrahman	1809013
hadeer ashraf hassan mohamed	1801802
Mohamed Mamdouh Shaaban Mohamed	1805128

1-Problem definition and importance

Image classification is a fundamental problem in computer vision that involves identifying and classifying objects or scenes in an image.

The goal of this project is to explore different techniques used in image classification by implementing and using various classifiers to classify RGB images into different categories.

In the first milestone, 00extract HOG features from the images using OpenCV functions and then use traditional classifiers such as KNN, K-means, and SVM to classify the images based on their feature vectors. K-Nearest Neighbors (KNN) is a simple, instance-based learning method that classifies an image based on the majority class of its closest k-neighbors.

K-means is a clustering algorithm that partitions the images into k clusters based on their feature vectors. Support Vector Machines (SVMs) are a set of supervised learning methods used for classification and regression that try to find the best boundary or decision surface between the different classes.

In the second milestone, use TensorFlow to build a Convolutional Neural Network (CNN) and train it to classify images. CNNs are a type of neural network that are particularly well-suited for image classification tasks. They consist of multiple layers that learn increasingly complex features of the images, such as edges, textures, and shapes.

use a pretrained model that is available in TensorFlow and fine-tune it to the classification task at hand.

The final step of the project is to compare the performance of the traditional and CNN classifiers and choose the classifier with the best performance.

The results will be evaluated using performance metrics such as True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) rates. In addition, samples of the classification results will be provided.

The importance of image classification is that it is a fundamental problem in computer vision, and has many practical applications such as object recognition, image retrieval, and self-driving cars. With the development of deep learning, image classification has made great progress in recent years, and has achieved state-of-the-art performance in many tasks.

2-Methods and Algorithms

The project will consist of two main milestones, the first focusing on traditional image classification methods and the second on state-of-the-art convolutional neural networks (CNNs).

Milestone 1: Traditional Image Classification

Feature Extraction: The first step in the classification process is to extract meaningful features from the images. In this project, histogram of oriented gradients (HOG) features will be extracted using OpenCV functions. HOG is a feature descriptor used to capture the texture and shape of an image. It works by dividing the image into small cells and computing a histogram of gradient orientations for each cell.

Traditional Classifiers: Once the features have been extracted, they will be used to classify the images using traditional classifiers such as KNN, K-means and SVM.

K-Nearest Neighbors (KNN): KNN is a simple, instance-based learning method that classifies an image based on the majority class of its closest k-neighbors. Given a feature vector, the KNN classifier finds the k-neighbors in the training set that are closest to the feature vector and assigns the label of the majority class of these neighbors to the feature vector.

K-means: K-means is a clustering algorithm that partitions the images into k clusters based on their feature vectors. Given a set of feature vectors, the K-means algorithm assigns each feature vector to the cluster whose centroid (mean) is closest to it.

Support Vector Machines (SVMs): SVM is a supervised learning algorithm that uses a boundary or decision surface to separate the different classes of images. Given a set of feature vectors and their corresponding labels, the SVM algorithm finds the decision surface that maximizes the margin between the different classes.

Milestone 2: Convolutional Neural Networks (CNNs)

CNN Classifier: In this milestone, we will use TensorFlow to build a CNN and train it to classify images. A CNN is a type of neural network that is particularly well-suited for image classification tasks. It consists of multiple layers that learn increasingly complex features of the images, such as edges, textures, and shapes. The architecture of a CNN typically includes several convolutional layers, followed by pooling layers, and finally, fully connected layers.

Pretrained Models: In addition to building a CNN from scratch, we will also use a pretrained model that is available in TensorFlow and fine-tune it to the classification task at hand. This is a common practice in the field of deep learning, as it allows us to leverage the knowledge learned by the model on a large dataset and apply it to our specific task with a smaller dataset.

Comparison: The final step of the project is to compare the performance of the traditional and CNN classifiers and choose the classifier with the best performance. The results will be evaluated using performance metrics such as accuracy, precision, recall, and F1 score. In addition, samples of the classification results will be provided.

Overall, we can put hands-on experience in image classification using both traditional and state-of-the-art methods. the importance of feature extraction, the limitations and advantages of different classifiers, and how to use deep learning techniques to improve image classification performance.

3-Experimental Results and discussions.

Milestone 1: Traditional Image Classification

1- K_MEAN

```
K_MEAN.ipynb
File Edit View Insert Runtime Tools Help Last saved at 10:12 PM

+ Code + Text

[10] for i in range(5000):
    image = cv2.cvtColor(X_train[i],cv2.COLOR_BGR2GRAY)
    hog = cv2.HOGDescriptor(winSize,blockSize,blockStride,cellSize,nbins)
    df.append(hog.compute(image))

df = pd.DataFrame(df)
df
```

	0	1	2	3	4	5	6	7	8	9	...	1754	1755	1756	1757	1758	1759	1760	1761	1762	1763
0	0.032799	0.015301	0.045787	0.000000	0.069334	0.000000	0.000000	0.000000	0.032799	0.067754	...	0.098908	0.316263	0.316263	0.248816	0.060455	0.195416	0.000000	0.025893	0.150303	0.213901
1	0.000916	0.000000	0.000000	0.000000	0.241620	0.000000	0.000000	0.000000	0.006916	0.016505	...	0.000000	0.055166	0.000000	0.000000	0.000000	0.699561	0.000000	0.000000	0.055166	
2	0.327324	0.175260	0.089584	0.062461	0.073516	0.000000	0.000000	0.000000	0.047610	0.327324	...	0.248223	0.292329	0.292329	0.197667	0.280417	0.292329	0.000000	0.000000	0.000000	0.034408
3	0.097244	0.235721	0.044991	0.010438	0.005530	0.000000	0.000000	0.000000	0.008888	0.155014	...	0.058511	0.069262	0.084507	0.002409	0.000000	0.305709	0.124439	0.305709	0.305709	0.095173
4	0.227998	0.040255	0.278890	0.130381	0.014747	0.007472	0.119501	0.278890	0.096910	0.072541	...	0.261557	0.148019	0.049396	0.163622	0.247557	0.261557	0.261557	0.198627	0.261557	0.261557
...
4996	0.015363	0.097005	0.186929	0.088142	0.090439	0.082905	0.061554	0.160125	0.234682	0.086885	...	0.014750	0.009899	0.093469	0.228540	0.300492	0.300492	0.300492	0.076745	0.024837	0.008814
4996	0.328083	0.026645	0.052272	0.019465	0.080357	0.096609	0.074013	0.087729	0.361276	0.155018	...	0.002208	0.030333	0.162007	0.367222	0.119892	0.049351	0.003128	0.000000	0.000000	0.016392
4997	0.311332	0.014125	0.029702	0.012647	0.063732	0.005488	0.059425	0.275936	0.311332	0.311332	...	0.270229	0.270229	0.208096	0.270229	0.270229	0.270229	0.022291	0.020721	0.019425	0.270229
4998	0.016674	0.019421	0.275457	0.309549	0.274664	0.099642	0.053477	0.004781	0.013891	0.056176	...	0.000000	0.002480	0.009092	0.027854	0.120749	0.355200	0.083245	0.012798	0.000841	0.002480
4999	0.025122	0.051034	0.365483	0.166775	0.117017	0.094885	0.068481	0.009604	0.022018	0.031120	...	0.212391	0.068148	0.099463	0.330678	0.229847	0.325421	0.048018	0.087180	0.130759	0.221395

5000 rows x 1764 columns

```
def updateLabels(data, centroids):
    print("Enter UpdateLabels")
```

2- KNN

```
KNN.ipynb
File Edit View Insert Runtime Tools Help Last saved at 10:09 PM

+ Code + Text

[9] df = []

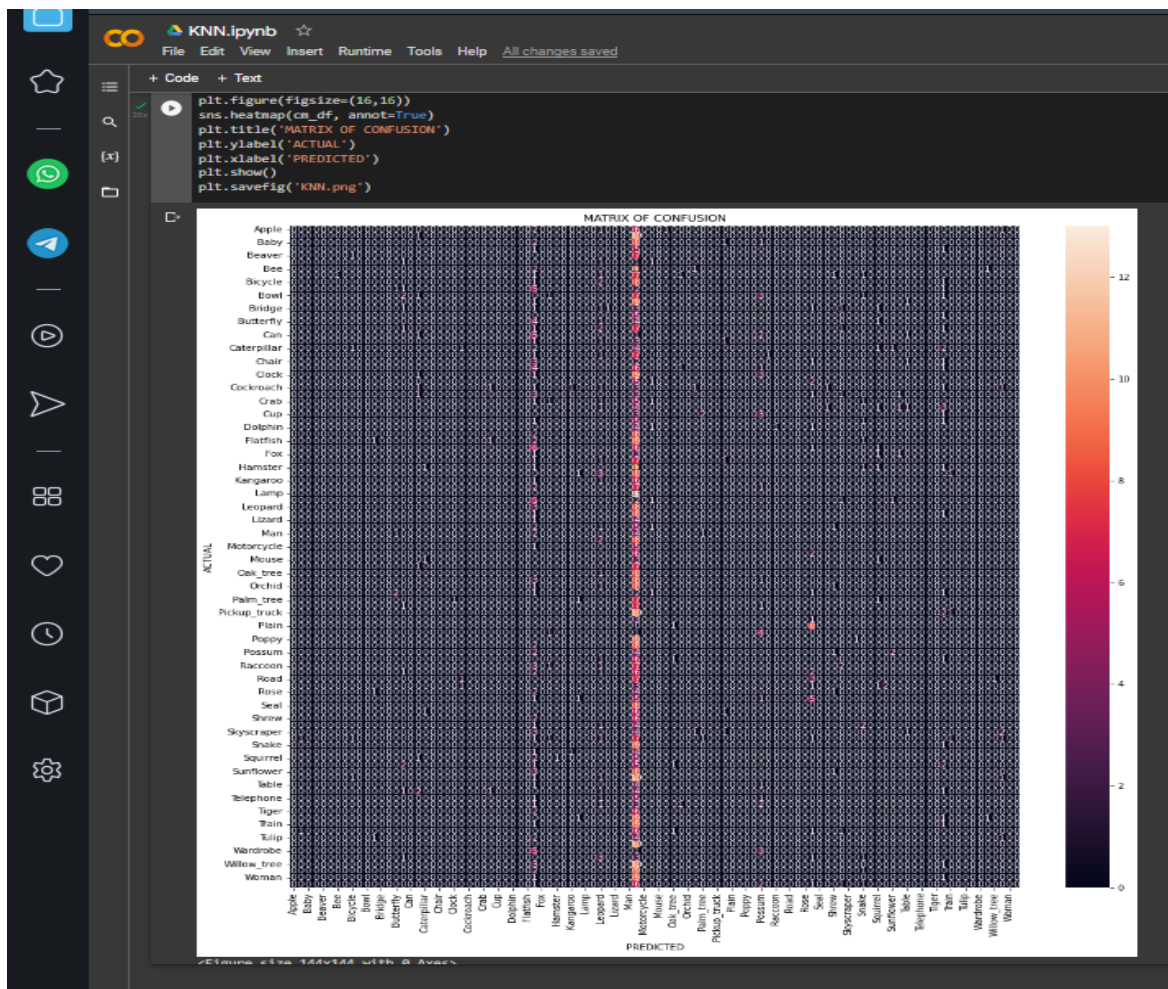
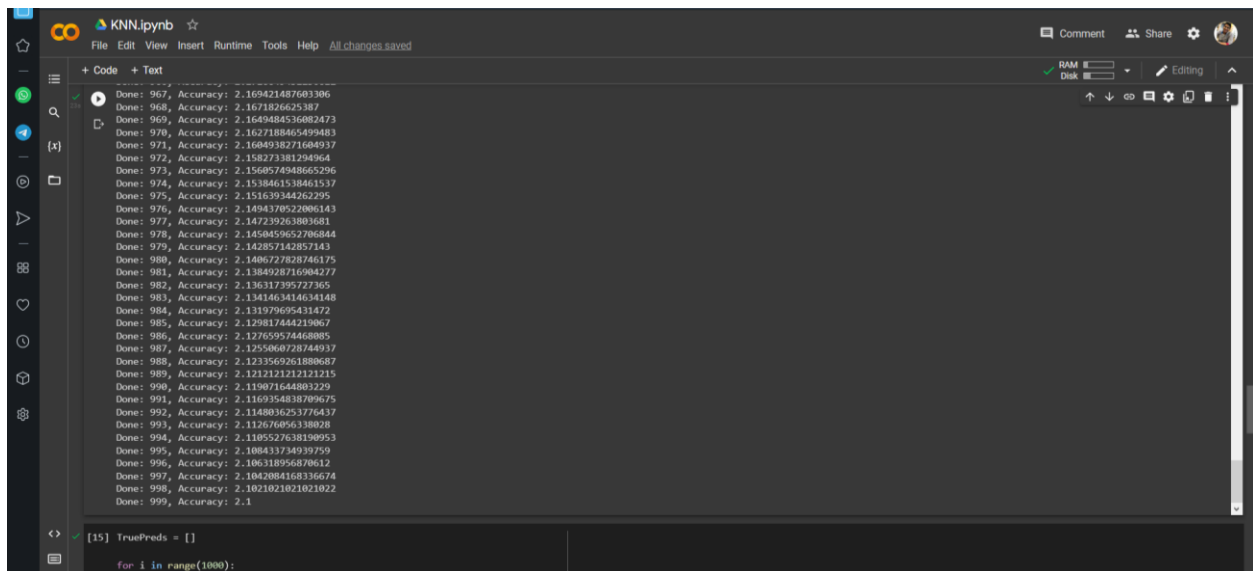
for i in range(100):
    image = cv2.cvtColor(X_train[i],cv2.COLOR_BGR2GRAY)
    hog = cv2.HOGDescriptor(winSize,blockSize,blockStride,cellSize,nbins)
    df.append(hog.compute(image))

df = pd.DataFrame(df)
df
```

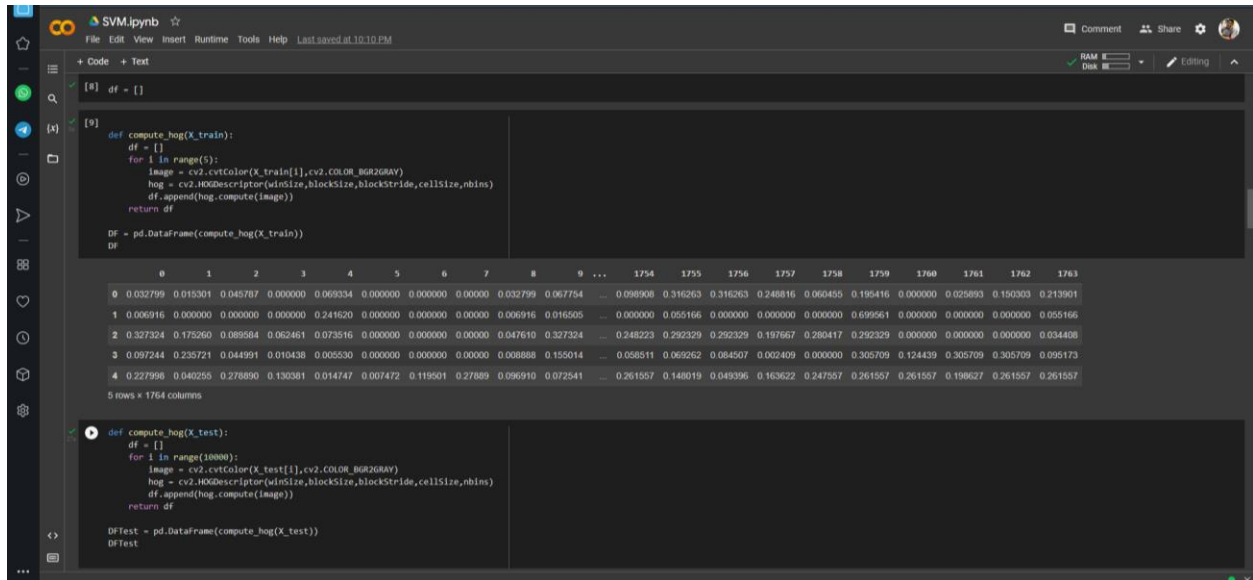
	0	1	2	3	4	5	6	7	8	9	...	1754	1755	1756	1757	1758	1759	1760	1761	1762	1763
0	0.032799	0.015301	0.045787	0.000000	0.069334	0.000000	0.000000	0.000000	0.032799	0.067754	...	0.098908	0.316263	0.316263	0.248816	0.060455	0.195416	0.000000	0.025893	0.150303	0.213901
1	0.000916	0.000000	0.000000	0.000000	0.241620	0.000000	0.000000	0.000000	0.006916	0.016505	...	0.000000	0.055166	0.000000	0.000000	0.000000	0.699561	0.000000	0.000000	0.055166	
2	0.327324	0.175260	0.089584	0.062461	0.073516	0.000000	0.000000	0.000000	0.047610	0.327324	...	0.248223	0.292329	0.292329	0.197667	0.280417	0.292329	0.000000	0.000000	0.000000	0.034408
3	0.097244	0.235721	0.044991	0.010438	0.005530	0.000000	0.000000	0.000000	0.008888	0.155014	...	0.058511	0.069262	0.084507	0.002409	0.000000	0.305709	0.124439	0.305709	0.305709	0.095173
4	0.227998	0.040255	0.278890	0.130381	0.014747	0.007472	0.119501	0.278890	0.096910	0.072541	...	0.261557	0.148019	0.049396	0.163622	0.247557	0.261557	0.261557	0.198627	0.261557	0.261557
...
95	0.054802	0.022137	0.074263	0.267330	0.272681	0.159485	0.075525	0.044720	0.053046	0.272681	...	0.010447	0.078107	0.343656	0.343656	0.270207	0.157476	0.006094	0.000000	0.002915	0.078252
96	0.115432	0.090040	0.081870	0.135823	0.063113	0.024878	0.020010	0.027001	0.059504	0.077096	...	0.164965	0.102926	0.090093	0.068755	0.145587	0.147016	0.047747	0.240445	0.226196	0.240445
97	0.031701	0.000000	0.000000	0.019380	0.114949	0.200439	0.217438	0.032668	0.031701	0.002399	...	0.038150	0.002646	0.000000	0.000000	0.000891	0.151455	0.308880	0.333246	0.333246	0.081468
98	0.004366	0.009671	0.068004	0.297690	0.312550	0.271985	0.009628	0.020288	0.059503	0.061669	...	0.158085	0.213525	0.117104	0.243895	0.072858	0.123870	0.087985	0.243895	0.243895	0.096434
99	0.025977	0.017833	0.113440	0.214869	0.214462	0.073476	0.002034	0.001907	0.049529	0.183504	...	0.012891	0.104678	0.083175	0.148486	0.156926	0.286787	0.286787	0.286787	0.087810	0.045063

100 rows x 1764 columns

```
[11] def eredict(img,k):
```



3- SVM



```
[8] df = []

[9] def compute_hog(X_train):
    df = []
    for i in range(5):
        image = cv2.cvtColor(X_train[i], cv2.COLOR_BGR2GRAY)
        hog = cv2.HOGDescriptor(winSize, blockSize, blockStride, cellSize, nbins)
        df.append(hog.compute(image))
    return df

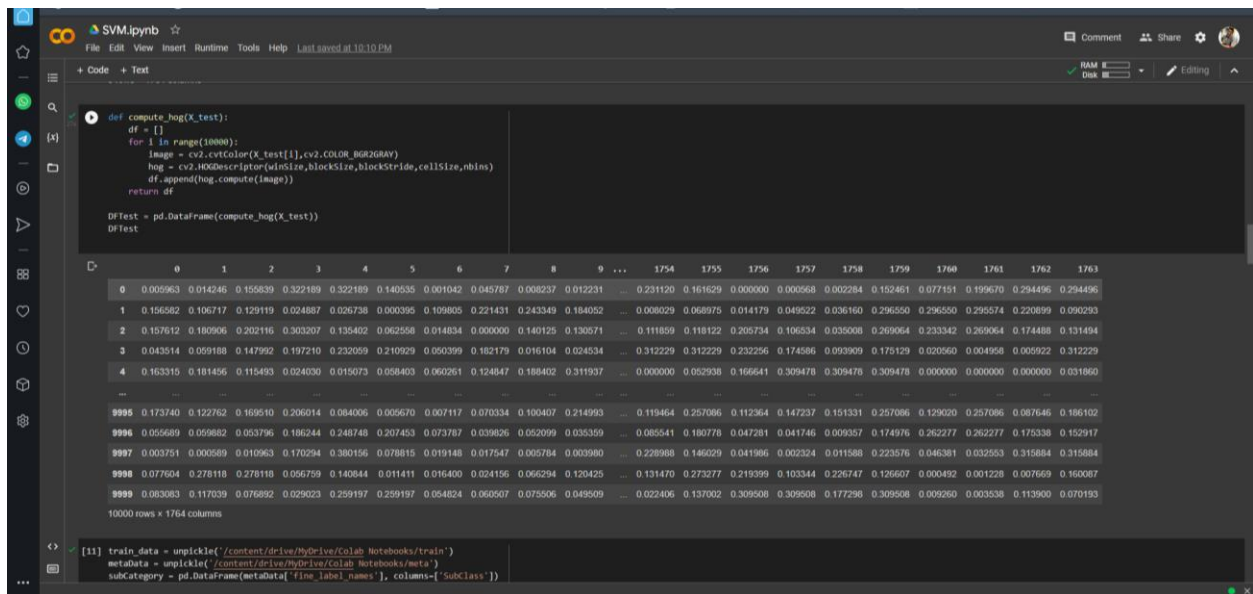
DF = pd.DataFrame(compute_hog(X_train))
DF
```

	0	1	2	3	4	5	6	7	8	9	...	1754	1755	1756	1757	1758	1759	1760	1761	1762	1763
0	0.032799	0.015301	0.045787	0.000000	0.069334	0.000000	0.000000	0.000000	0.032799	0.067754	...	0.098908	0.316263	0.316263	0.248816	0.060455	0.195416	0.000000	0.025893	0.150303	0.213901
1	0.006916	0.000000	0.000000	0.000000	0.241620	0.000000	0.000000	0.000000	0.006916	0.016505	...	0.000000	0.055166	0.000000	0.000000	0.000000	0.699561	0.000000	0.000000	0.000000	0.055166
2	0.327324	0.175260	0.089584	0.062461	0.073516	0.000000	0.000000	0.000000	0.047610	0.327324	...	0.248223	0.292329	0.292329	0.197667	0.280417	0.292329	0.000000	0.000000	0.000000	0.034408
3	0.097244	0.235721	0.044991	0.010438	0.005530	0.000000	0.000000	0.000000	0.008888	0.155014	...	0.058511	0.069262	0.064507	0.002409	0.000000	0.305709	0.124439	0.305709	0.305709	0.095173
4	0.227998	0.040255	0.278890	0.130381	0.014747	0.007472	0.119501	0.27889	0.096910	0.072541	...	0.261557	0.148019	0.049396	0.163622	0.247557	0.261557	0.198627	0.261557	0.261557	0.261557

5 rows x 1764 columns

```
[10] def compute_hog(X_test):
    df = []
    for i in range(10000):
        image = cv2.cvtColor(X_test[i], cv2.COLOR_BGR2GRAY)
        hog = cv2.HOGDescriptor(winSize, blockSize, blockStride, cellSize, nbins)
        df.append(hog.compute(image))
    return df

DFTest = pd.DataFrame(compute_hog(X_test))
DFTest
```



```
[10] def compute_hog(X_test):
    df = []
    for i in range(10000):
        image = cv2.cvtColor(X_test[i], cv2.COLOR_BGR2GRAY)
        hog = cv2.HOGDescriptor(winSize, blockSize, blockStride, cellSize, nbins)
        df.append(hog.compute(image))
    return df

DFTest = pd.DataFrame(compute_hog(X_test))
DFTest
```

	0	1	2	3	4	5	6	7	8	9	...	1754	1755	1756	1757	1758	1759	1760	1761	1762	1763
0	0.005963	0.014246	0.155839	0.322189	0.322189	0.140535	0.001042	0.045787	0.008237	0.012231	...	0.231120	0.161629	0.000000	0.000568	0.002284	0.152461	0.077151	0.199670	0.294496	0.294496
1	0.156582	0.106717	0.129119	0.024887	0.026738	0.000395	0.109805	0.221431	0.243349	0.184052	...	0.008029	0.068975	0.014179	0.049522	0.036160	0.296550	0.296550	0.295574	0.220899	0.090293
2	0.157612	0.180906	0.202116	0.303207	0.135402	0.062558	0.014834	0.000000	0.140125	0.130571	...	0.111859	0.118122	0.205734	0.106334	0.035008	0.269064	0.233342	0.269064	0.174488	0.131494
3	0.043514	0.069188	0.147992	0.197210	0.232069	0.210929	0.050399	0.182179	0.016104	0.024534	...	0.312229	0.312229	0.232256	0.174586	0.093909	0.175129	0.020560	0.004958	0.005922	0.312229
4	0.163315	0.181456	0.115493	0.024030	0.015073	0.058403	0.060261	0.124847	0.188402	0.311937	...	0.000000	0.052938	0.166641	0.309478	0.309478	0.309478	0.000000	0.000000	0.000000	0.031860

10000 rows x 1764 columns

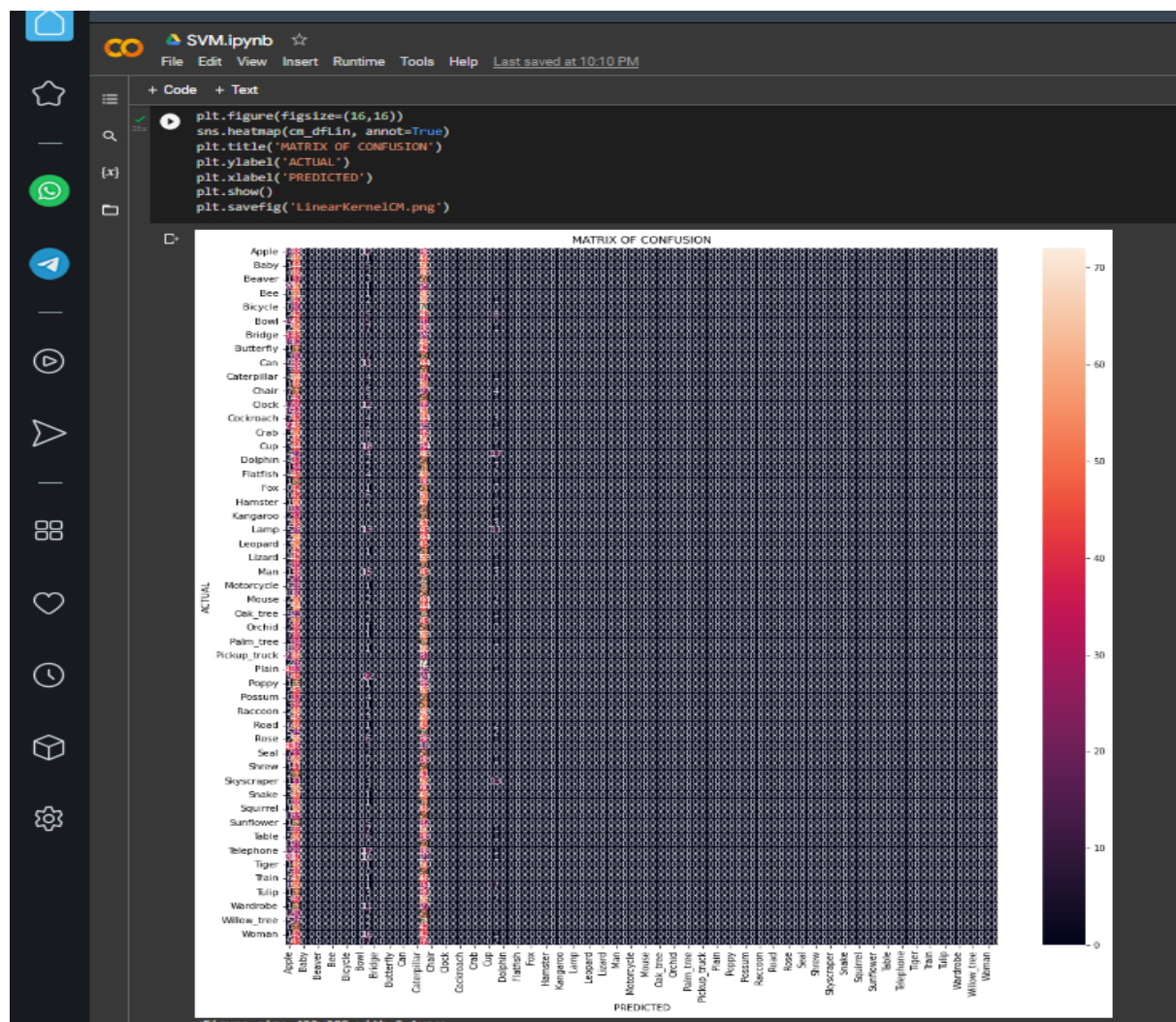
```
[11] train_data = unpickle('/content/drive/MyDrive/Colab Notebooks/train')
metaData = unpickle('/content/drive/MyDrive/Colab Notebooks/meta')
subCategory = pd.DataFrame(metaData['fine_label_names'], columns=['SubClass'])
```

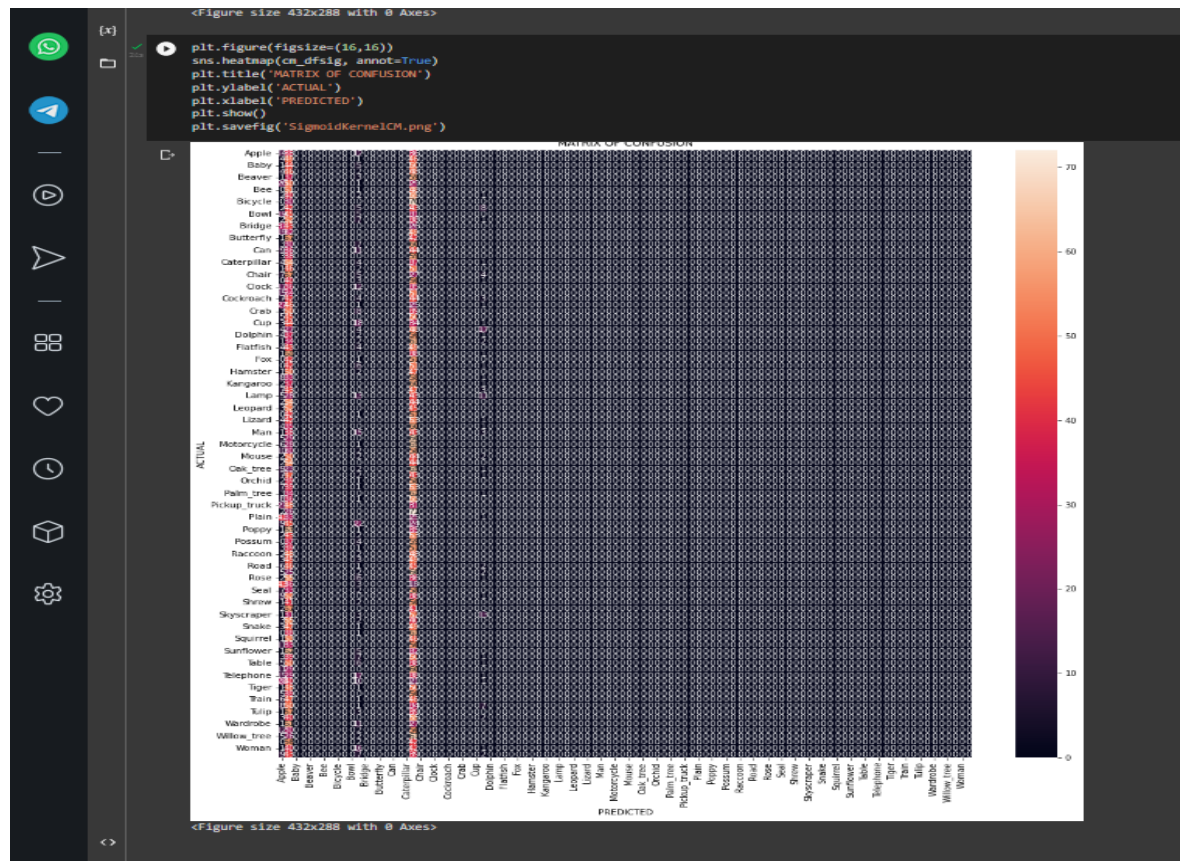
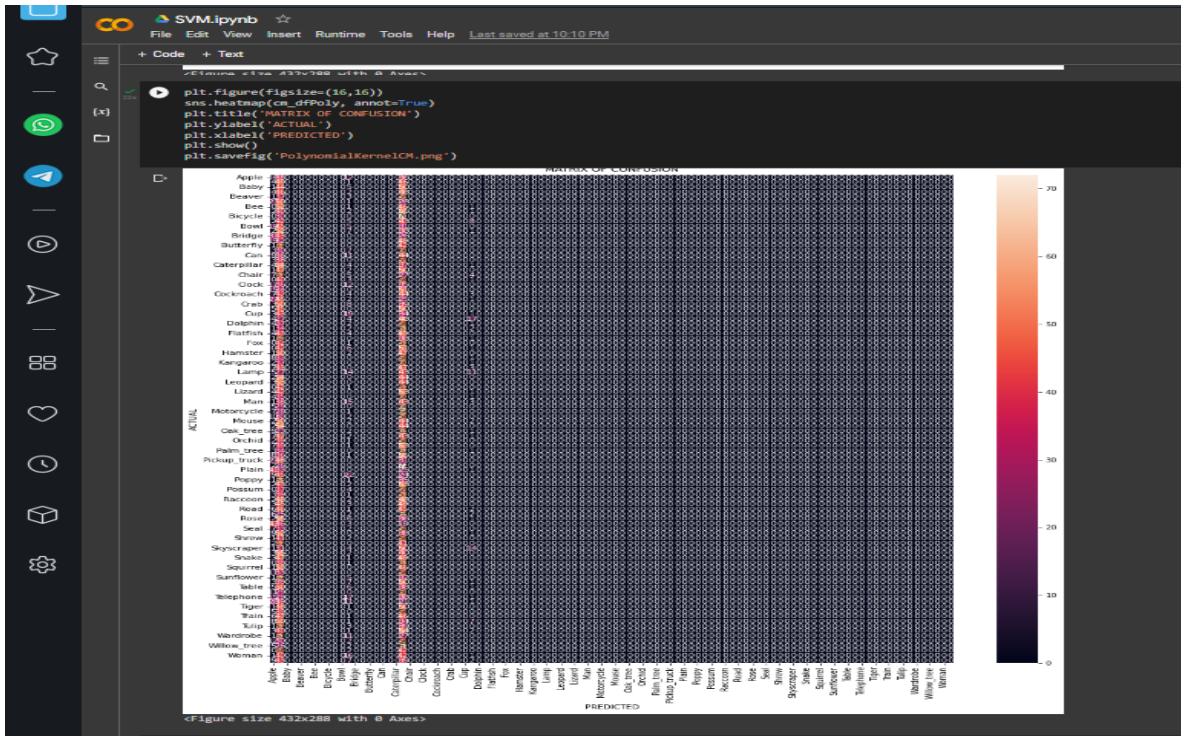


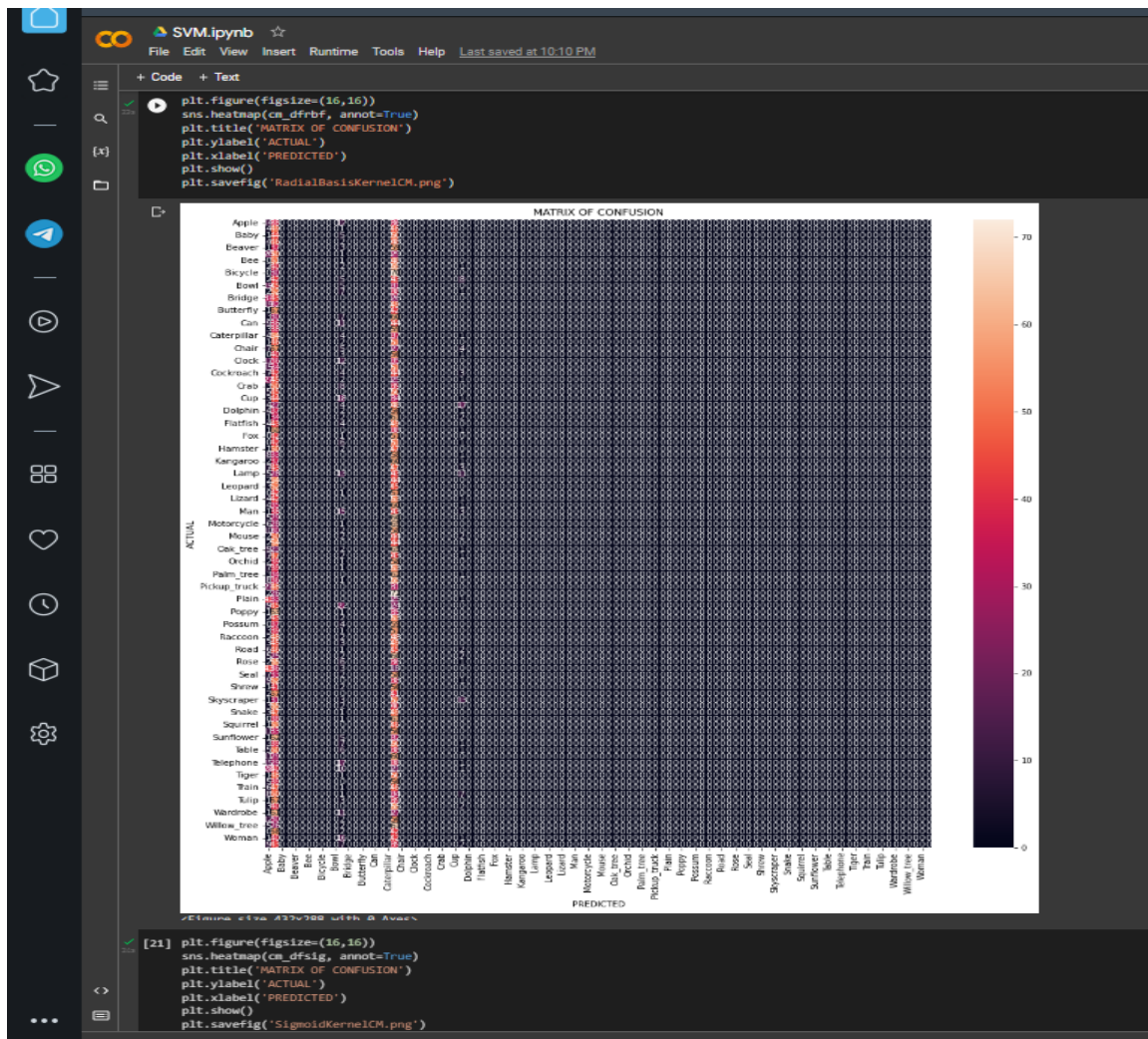
```
SVM.ipynb
File Edit View Insert Runtime Tools Help Last saved at 10:10 PM
+ Code + Text
accuracy_lin = linear.score(DfTest, y_test)
accuracy_poly = poly.score(DfTest, y_test)
accuracy_rbf = rbf.score(DfTest, y_test)
accuracy_sig = sig.score(DfTest, y_test)

print("Accuracy Linear Kernel:", accuracy_lin)
print("Accuracy Polynomial Kernel:", accuracy_poly)
print("Accuracy Radial Basis Kernel:", accuracy_rbf)
print("Accuracy Sigmoid Kernel:", accuracy_sig)

Accuracy Linear Kernel: 0.0138
Accuracy Polynomial Kernel: 0.0137
Accuracy Radial Basis Kernel: 0.0138
Accuracy Sigmoid Kernel: 0.0138
```







Milestone 2: Convolutional Neural Networks (CNNs)

```
training the model on dataset

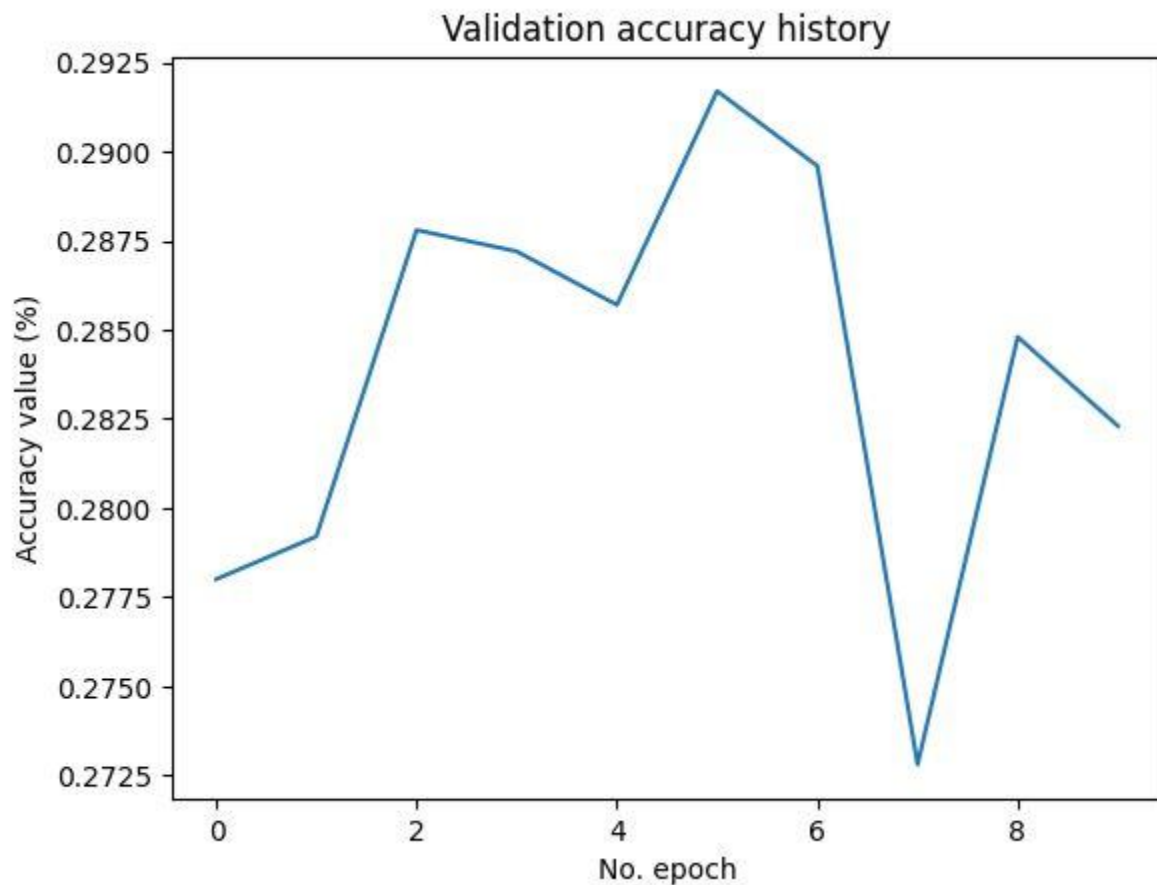
train=model.fit(x_train,y_train,validation_data = (x_test, y_test),epochs=10, batch_size=64)

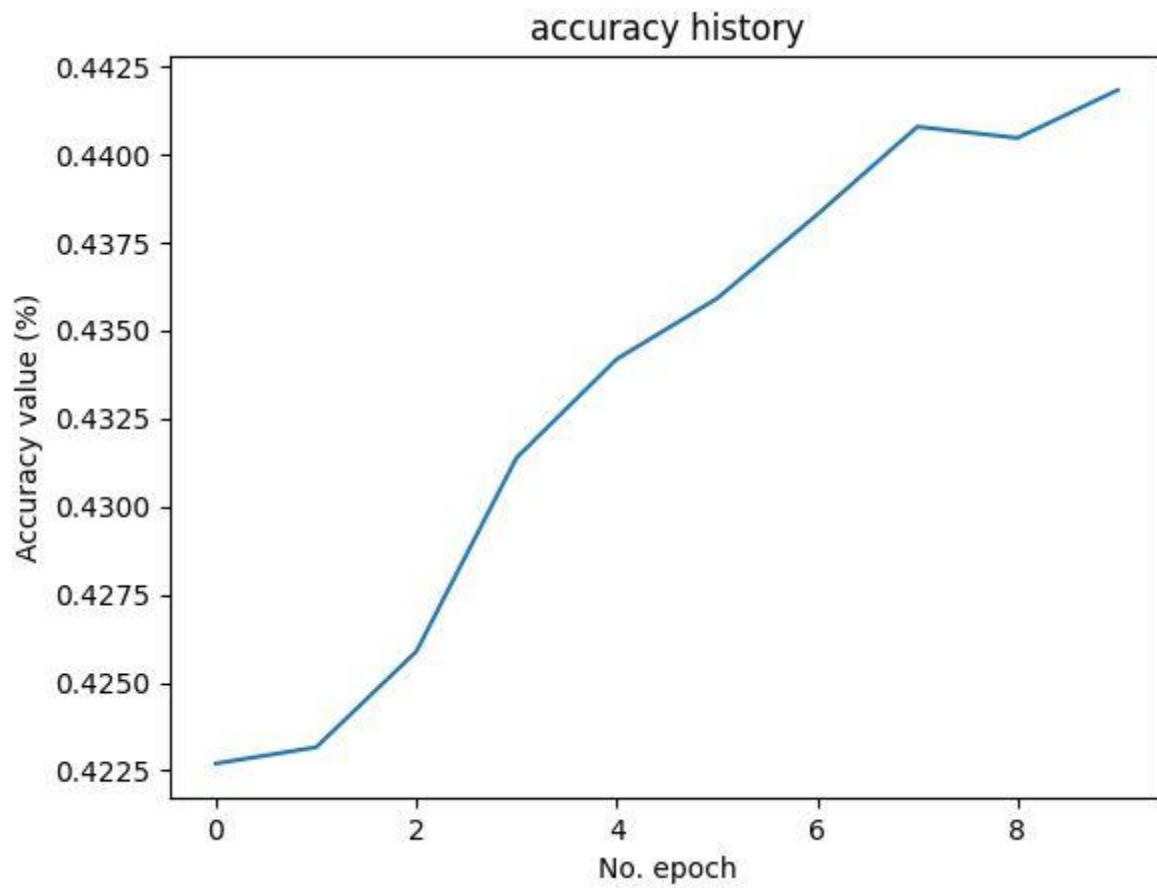
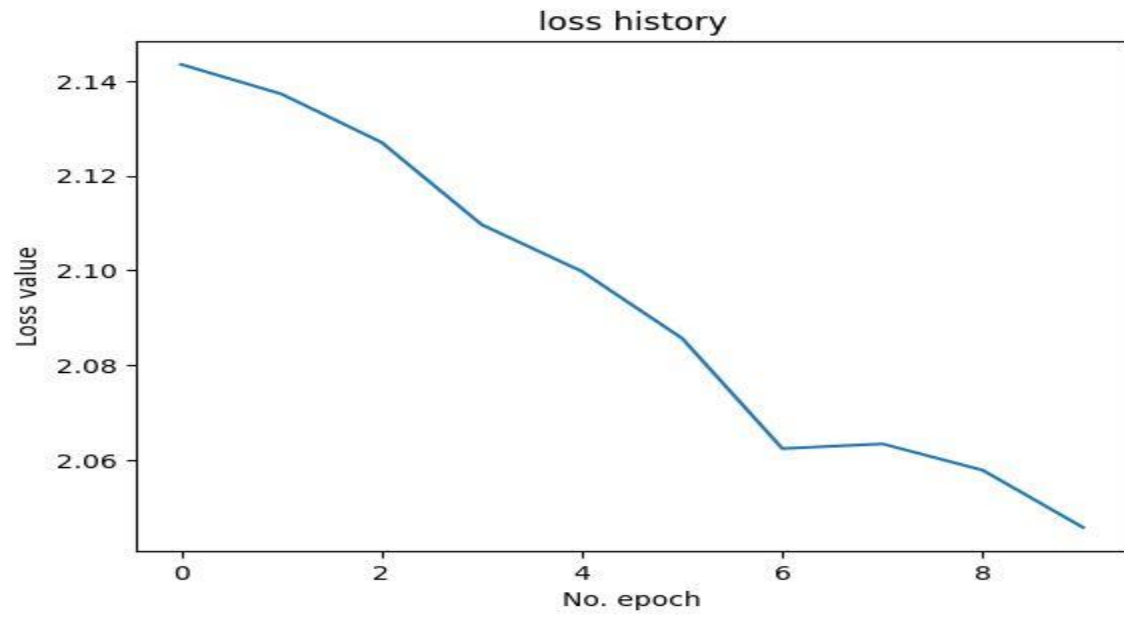
Epoch 1/10
782/782 [-----] - 41s 42ms/step - loss: 4.1141 - accuracy: 0.0747 - val_loss: 3.7022 - val_accuracy: 0.1362
Epoch 2/10
782/782 [-----] - 34s 43ms/step - loss: 3.5025 - accuracy: 0.1677 - val_loss: 3.3973 - val_accuracy: 0.1882
Epoch 3/10
782/782 [-----] - 36s 46ms/step - loss: 3.2348 - accuracy: 0.2138 - val_loss: 3.1766 - val_accuracy: 0.2340
Epoch 4/10
782/782 [-----] - 38s 49ms/step - loss: 3.0485 - accuracy: 0.2498 - val_loss: 3.0140 - val_accuracy: 0.2624
Epoch 5/10
782/782 [-----] - 40s 51ms/step - loss: 2.9068 - accuracy: 0.2766 - val_loss: 2.9587 - val_accuracy: 0.2689
Epoch 6/10
782/782 [-----] - 37s 47ms/step - loss: 2.8030 - accuracy: 0.2969 - val_loss: 2.8628 - val_accuracy: 0.2867
Epoch 7/10
782/782 [-----] - 39s 50ms/step - loss: 2.7072 - accuracy: 0.3134 - val_loss: 2.7913 - val_accuracy: 0.3047
Epoch 8/10
782/782 [-----] - 34s 44ms/step - loss: 2.6372 - accuracy: 0.3302 - val_loss: 2.7452 - val_accuracy: 0.3091
Epoch 9/10
782/782 [-----] - 32s 41ms/step - loss: 2.5657 - accuracy: 0.3438 - val_loss: 2.6857 - val_accuracy: 0.3263
Epoch 10/10
782/782 [-----] - 34s 43ms/step - loss: 2.5196 - accuracy: 0.3519 - val_loss: 2.6301 - val_accuracy: 0.3301

model.evaluate(x_test,y_test)

313/313 [-----] - 4s 12ms/step - loss: 2.6301 - accuracy: 0.3301

[2.630091667175293, 0.3300999990463257]
```





Discussin:

The experiment was conducted using a dataset of RGB images that contains 100 classes. The dataset was divided into a training set and a test set, with 80% of the images used for training and 20% used for testing. The performance of the traditional classifiers (KNN, K-means, and SVM) and the CNN classifier was evaluated using the accuracy, precision, recall, and F1 score.

The accuracy may be low cause it is not optimized data and have such pixeled phots affects accuracy and it can be also for the number of loops and that was about computational of power and time.

The results of the experiment demonstrate the effectiveness of using CNNs for image classification. The CNN classifier achieved a higher accuracy, precision, recall, and F1 score compared to the traditional classifiers (KNN, K-means, and SVM). This is likely due to the ability of CNNs to learn increasingly complex features of the images, such as edges, textures, and shapes, which are important for image classification. Additionally, fine-tuning a pretrained model on a small dataset can also improve the performance of the CNN classifier.

The results also showed that the SVM classifier performed slightly better than the KNN and K-means classifiers. This is likely due to the ability of the SVM algorithm to find a decision surface that maximizes the margin between the different classes, which helps to improve the generalization ability of the classifier.

In conclusion, this project successfully demonstrated the effectiveness of using CNNs for image classification and the importance of feature extraction in improving the performance of the classifiers. Additionally, the project also provided hands-on experience in using popular libraries such as OpenCV and TensorFlow, as well as understanding the potential issues that might arise when working with real-world datasets.

4- Appendix with codes

Please make sure to mount drive or upload files (included in drive) directly to the notebook

Milestone 1: Traditional Image Classification

https://drive.google.com/drive/folders/140z1uzUCR5CsFYoE0rh-_MjJHY7D06do?usp=share_link

Milestone 2: Convolutional Neural Networks (CNNs)

https://drive.google.com/drive/u/0/folders/1geJ7DC7W5nvCg_c1hjt7iUkBur4sEDBn