Object Detection

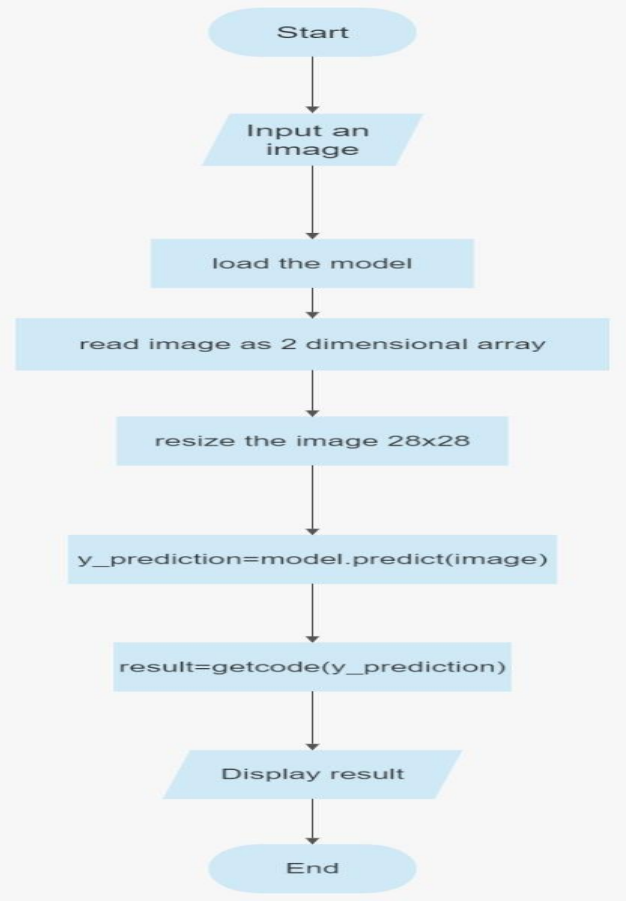
1.Project idea

-Object detection is an important computer vision task used to detect instances of visual objects of certain classes (for example, humans, animals, cars, or buildings) in digital images such as photos or video frames.

- Object Detection using Artificial Neural Networks.

-it classifies image into one of the ten given classes.

-our project detect ten type of clothes (-T-shirt/top ,Trouser, Pullover, Dress, Coat,Sandal ,Shirt, Sneaker, Bag, Ankle boot)



2.Main Functionalities

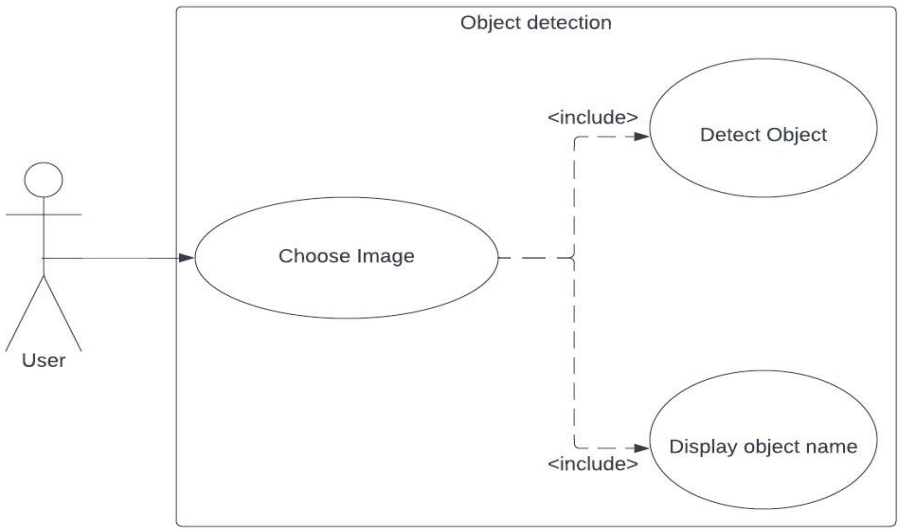
Project shows the name of the object in a given photo and say its name

Steps:-

1-choose image from device.

2-system predicts what these images are.

3-the system shows images and it's name.



3.Similar applications in the market

-**medical field** (desktop application): image recognition is mainly used to improve clinical diagnosis and case studies, such as cancer cells,white blood cells, chromosomal examination, repair surgery control design and so on.

-Remote sensing image recognition: through processing and analyzing satellite remote sensing pictures and aerospace images, this technology can be used for geology and geophysics, weather forecasting, intelligent detection of agricultural diseases, environmental pollution detection,military target detection and tracking, guidance and so on

- Tracking objects

It is needless to point out that in the field of security and surveillance object detection would play an even more important role. With object tracking it would be easier to track a person in a video. Object tracking could also be used in tracking the motion of a ball during a match. In the field of traffic monitoring too object tracking plays a crucial role.

-Self driving cars

Another unique application of object detection technique is definitely self-driving cars. A self-driving car can only navigate through a street safely if it could detect all the objects such as people, other cars, road signs on the road, in order to decide what action to take.

-Detecting a vehicle

In a road full of speeding vehicles object detection can help in a big way by tracking a particular vehicle and even its number plate. So, if a car gets into an accident or, breaks traffic rules then it is easier to detect that particular car using object detection model and thereby decreasing the rate of crime while enhancing security.

4. An initial literature review of Academic publications (papers) relevant to the idea (at least 5 papers)

1-**Automated image identification, detection and fruit counting of top-view pineapple crown using machine learning**

Abstract:
Automated fruit identification or recognition using image processing is a key element in precision agriculture for performing object detection in large crop plots. Automation of fruit recognition for the captured top-view of RGB based images using an unmanned aerial vehicle (UAV) is a challenge. Image analysis demonstrated the difficulty of processing the captured image under variant illumination in natural environment and with textured objects of non-ideal geometric shapes. However, this is subjected to certain consideration settings and image-processing algorithms. The study presents an automatic method for identifying and recognizing the pineapple's crown images in the designated plot using image processing and further counts the detected images using machine learning classifiers namely artificial neural network (ANN), support vector machine (SVM), random forest (RF), naive Bayes (NB), decision trees (DT) and k-nearest neighbors (KNN). The high spatial-resolution aerial images were pre-processed and segmented, and its extracted features were analyzed according to shape, color and texture for recognizing the pineapple crown before classifying it as fruit or non-fruit. Feature fusion using one-way analysis of variance (ANOVA) was incorporated in this study to optimize the performance of machine learning classifier. The algorithm was quantitatively analyzed and validated for performance via accuracy, specificity, sensitivity and precision. The detection for the pineapple's crown images with ANN-GDX classification has demonstrated best performance fruit counting with accuracy of 94.4% and has thus demonstrated clear potential application of an effective RGB images analysis for the pineapple industry.

2-Automated plant identification using artificial neural network and support vector machine

Abstract:

Ficus is one of the largest genera in plant kingdom reaching to about 1000 species worldwide. While taxonomic keys are available for identifying most species of Ficus, it is very difficult and time consuming for interpretation by a nonprofessional thus requires highly trained taxonomists. The purpose of the current study is to develop an efficient baseline automated system, using image processing with pattern recognition approach, to identify three species

of Ficus, which have similar leaf morphology. Leaf images from three different Ficus species namely F. benjamina, F. pellucidopunctata and F. sumatrana were selected. A total of 54 leaf image samples were used in this study. Three main steps that are image pre-processing, feature extraction and recognition were carried out to develop the proposed system. Artificial neural network (ANN) and support vector machine (SVM) were the implemented recognition models. Evaluation results showed the ability of the proposed system to recognize leaf images with an accuracy of 83.3%. However, the ANN model performed slightly better using the AUC evaluation criteria. The system developed in the current study is able to classify the selected Ficus species with acceptable accuracy.

3-Automated Object Identification Using Optical Video Cameras on Construction Sites

Abstract:

Visual recording devices such as video cameras, CCTVs, or webcams have been broadly used to facilitate work progress or safety monitoring on construction sites. Without human intervention, however, both real-time reasoning about captured scenes and interpretation of recorded images are challenging tasks. This article presents an exploratory method for automated object identification using standard video cameras on construction sites. The proposed method supports real-time detection and classification of mobile heavy equipment and workers. The background subtraction algorithm extracts motion pixels from an image sequence, the pixels are then grouped into regions to represent moving objects, and finally the regions are identified as a certain object using classifiers. For evaluating the method, the formulated computer-aided process was implemented on actual construction sites, and promising results were obtained. This article is expected to contribute to future applications of automated monitoring systems of work zone safety or productivity.

4-Toolchain Development for Automated Scene Reconstruction using Artificial Neural Network Object Detection and Photogrammetry for the Application in Driving Simulators

Abstract:

This paper presents an automated process chain for the reconstruction of characteristic 3D objects, which can be used in a simulation environment. The process chain can distinguish between recurring objects such as trees and cars and specific objects like buildings. To acquire this, it detects and classifies objects in images from a previously recorded video. In contrast to the specific objects, which are reconstructed during the

workflow of the process chain, the recurrent objects are loaded from already existing models and are placed multiple times into the simulation environment. In terms of quality a visual comparison between the two integrated programs for the reconstruction (Metashape and Meshroom) is carried out. Furthermore the accuracy of the positioning of standard objects in the Unity game engine is examined.

5-Design and development of a machine vision system using artificial neural network-based algorithm for automated coal characterization

Abstract:

Coal is heterogeneous in nature, and thus the characterization of coal is essential before its use for a specific purpose. Thus, the current study aims to develop a machine vision system for automated coal characterizations. The model was calibrated using 80 image samples that are captured for different coal samples in different angles. All the images were captured in RGB color space and converted into five other color spaces (HSI, CMYK, Lab, xyz, Gray) for feature extraction. The intensity component image of HSI color space was further transformed into four frequency components (discrete cosine transform, discrete wavelet transform, discrete Fourier transform, and Gabor filter) for the texture features extraction. A total of 280 image features was extracted and optimized using a step-wise linear regression-based algorithm for model development. The datasets of the optimized features were used as an input for the model, and their respective coal characteristics (analyzed in the laboratory) were used as outputs of the model. The R-squared values were found to be 0.89, 0.92, 0.92, and 0.84, respectively, for fixed carbon, ash content, volatile matter, and moisture content. The performance of the proposed artificial neural network model was also compared with the performances of performances of Gaussian process regression, support vector regression, and radial basis neural network models. The study demonstrates the potential of the machine vision system in automated coal characterization.

5. the Dataset employed (preferably a publicly available dataset)

-Dataset from kaggle website.

https://www.kaggle.com/datasets/zalando-research/fashionmnist

A) ANN clothes dataset.

-T-shirt

-Trouser

-Pullover

-Dress

-Coat

-Sand

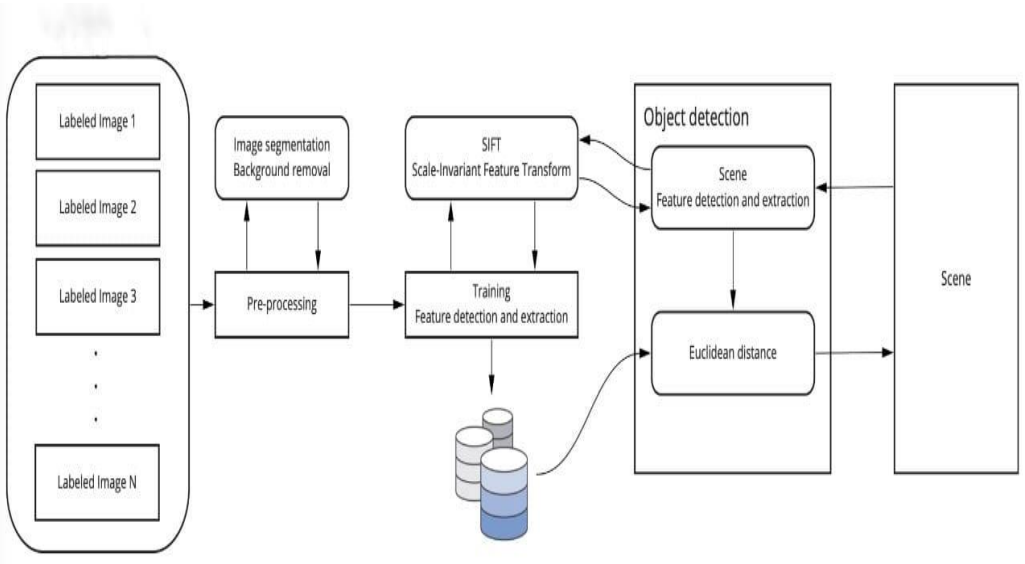
-Shirt

-Sneaker

-Bag

-Ankle boot

6.Details of the algorithm(s)/approach(es) used and the results of the experiments



A) ANN MODEL ALGORITHM.

1-Load fashion images and save each of them into a specific index.

```
def save_image(filename, data_array):
    im = Image.fromarray(data_array.astype('uint8'))
    im_invert = ImageOps.invert(im)
    im_invert.save(filename)

# Load Fashion-MNIST Data
(x_train, y_train), (x_test, y_test) = fashion_mnist.load_data()

DIR_NAME = "data/train/Tshirt"
if os.path.exists(DIR_NAME) == False:
    os.mkdir(DIR_NAME)
DIR_NAME = "data/train/Trouser"
if os.path.exists(DIR_NAME) == False:
    os.mkdir(DIR_NAME)
DIR_NAME = "data/train/Dress"
if os.path.exists(DIR_NAME) == False:
    os.mkdir(DIR_NAME)
DIR_NAME = "data/train/Coat"
if os.path.exists(DIR_NAME) == False:
    os.mkdir(DIR_NAME)
DIR_NAME = "data/train/Sandal"
if os.path.exists(DIR_NAME) == False:
    os.mkdir(DIR_NAME)
DIR_NAME = "data/train/Shirt"
if os.path.exists(DIR_NAME) == False:
    os.mkdir(DIR_NAME)
DIR_NAME = "data/train/Sneaker"
if os.path.exists(DIR_NAME) == False:
    os.mkdir(DIR_NAME)
DIR_NAME = "data/train/Ankle boot"
if os.path.exists(DIR_NAME) == False:
    os.mkdir(DIR_NAME)
DIR_NAME = "data/train/Bag"
if os.path.exists(DIR_NAME) == False:
    os.mkdir(DIR_NAME)
DIR_NAME = "data/train/Pullover"
if os.path.exists(DIR_NAME) == False:
```

```
i = 0
for li in [x_train]:
    print("[------------------------------------------------------------------]")
    for x in li:
        name = getcode(y_train[i])
        if name == "Tshirt" :
            filename = "data/train/Tshirt/{0}.jpg".format(i)
        elif name == "Trouser" :
            filename = "data/train/Trouser/{0}.jpg".format(i)
        elif name == "Dress" :
            filename = "data/train/Dress/{0}.jpg".format(i)
        elif name == "Coat" :
            filename = "data/train/Coat/{0}.jpg".format(i)
        elif name == "Sandal" :
            filename = "data/train/Sandal/{0}.jpg".format(i)
        elif name == "Shirt" :
            filename = "data/train/Shirt/{0}.jpg".format(i)
        elif name == "Sneaker" :
            filename = "data/train/Sneaker/{0}.jpg".format(i)
        elif name == "Bag" :
            filename = "data/train/Bag/{0}.jpg".format(i)
        elif name == "Ankle boot" :
            filename = "data/train/Ankle boot/{0}.jpg".format(i)
        elif name == "Pullover" :
            filename = "data/train/Pullover/{0}.jpg".format(i)
        print(filename)
        save_image(filename, x)
        i += 1
```

***convert each folder to label.**

```
code = {'Tshirt':0 ,'Trouser':1,'Pullover':2,'Dress':3,'Coat':4,'Sandal':5,'Shirt':6,'Sneaker':7,'Bag':8,'Ankle boot':9}
####################
def getcode(n) :
    for x , y in code.items() :
        if n == y :
            return x
```

***Read files of the training dataset.**

```
# Reading files [ training_dataset ]

for folder in  os.listdir(trainpath + 'train') :
    files = gb.glob(pathname= str( trainpath +'train//' + folder + '/*.jpg'))
    print(f'For training data , found {len(files)} in folder {folder}')
```

```
For training data , found 6000 in folder Sneaker
For training data , found 6000 in folder Pullover
For training data , found 6000 in folder Coat
For training data , found 6000 in folder Dress
For training data , found 6000 in folder Tshirt
For training data , found 6000 in folder Sandal
For training data , found 6000 in folder Ankle boot
For training data , found 6000 in folder Bag
For training data , found 6000 in folder Shirt
For training data , found 6000 in folder Trouser
```

*Read files of the test dataset.

```
# Reading files [ test_dataset ]

for folder in  os.listdir(testpath +'test') :
    files = gb.glob(pathname= str( testpath +'test//' + folder + '/*.jpg'))
    print(f'For testing data , found {len(files)} in folder {folder}')
```

```
For testing data , found 1000 in folder Sneaker
For testing data , found 1000 in folder Pullover
For testing data , found 1000 in folder Coat
For testing data , found 1000 in folder Dress
For testing data , found 1000 in folder Tshirt
For testing data , found 1000 in folder Sandal
For testing data , found 1000 in folder Ankle boot
For testing data , found 1000 in folder Bag
For testing data , found 1000 in folder Shirt
For testing data , found 1000 in folder Trouser
```

*make resize on training and test dataset.

n [15]:
```
# to resize images_train to one equalied size
X_train = []
y_train = []

for folder in  os.listdir(trainpath +'train') :
    files = gb.glob(pathname= str( trainpath +'train//' + folder + '/*.jpg'))
    for file in files:
        image = cv2.imread(file)
        imgGray = color.rgb2gray( image)
        X_train.append(list(imgGray))
        y_train.append(code[folder])
```

n [16]:
```
# to resize images_test to one equalied size

X_test = []
y_test = []
for folder in  os.listdir(testpath +'test') :
    files = gb.glob(pathname= str(testpath + 'test//' + folder + '/*.jpg'))
    for file in files:
        image = cv2.imread(file)
        imgGray = color.rgb2gray( image)
        X_test.append(list(imgGray))
        y_test.append(code[folder])
```

*show the shape of images.

In [5]:
```
# Shape of training nad test data
print(f'Shape of train_images: {train_images.shape}')
print(f'Shape of train_labels: {train_labels.shape}')
print(f'Shape of test_images: {test_images.shape}')
print(f'Shape of test_labels: {test_labels.shape}')
```

```
Shape of train_images: (60000, 28, 28)
Shape of train_labels: (60000,)
Shape of test_images: (10000, 28, 28)
Shape of test_labels: (10000,)
```

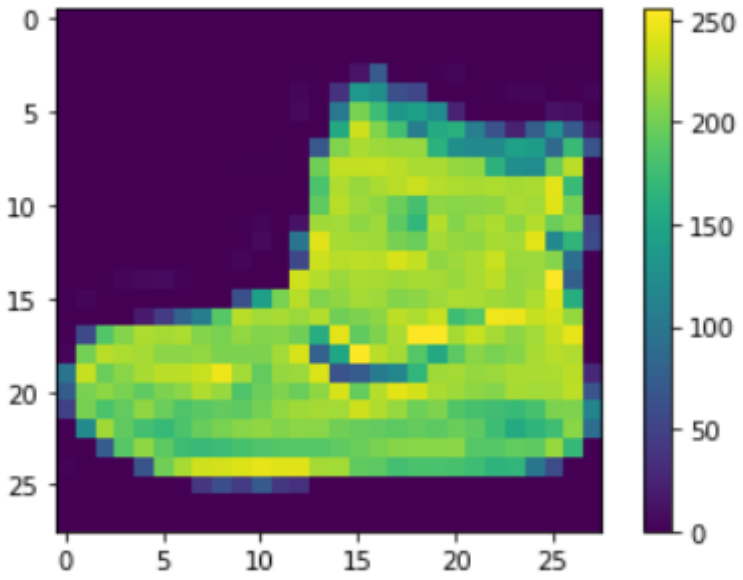
***make labels of images.**

```
In [6]: # There are 10 labels starting from 0 to 9
        print(f'Unique train labels: {np.unique(train_labels)}')
        print(f'Unique test labels: {np.unique(test_labels)}')
```

```
Unique train labels: [0 1 2 3 4 5 6 7 8 9]
Unique test labels: [0 1 2 3 4 5 6 7 8 9]
```

***check if the image is correct or not.**

```
: plt.figure()
  plt.imshow(train_images[0])
  plt.colorbar()
  plt.grid(False)
  plt.show()
```



***make rescalling to the images.**

```
In [11]: train_images = train_images / 255.0
         test_images = test_images / 255.0
```

***convert images to array.**

In [17]:
```
################## preprocessing images to arrays ####################
X_train = np.array(X_train)
X_test = np.array(X_test)
#X_pred_array = np.array(X_pred)
y_train = np.array(y_train)
y_test = np.array(y_test)
```

In [18]:
```
print(X_train)
```

```
[[[1.         1.         1.         ... 1.         1.         1.        ]
  [1.         1.         1.         ... 1.         1.         1.        ]
  [1.         1.         1.         ... 1.         1.         1.        ]
  ...
  [1.         1.         1.         ... 1.         1.         1.        ]
  [1.         1.         1.         ... 1.         1.         1.        ]
  [1.         1.         1.         ... 1.         1.         1.        ]]

 [[1.         1.         1.         ... 1.         1.         1.        ]
  [1.         1.         1.         ... 0.98431373 0.98431373 0.98431373]
  [1.         1.         1.         ... 1.         0.99215686 0.98823529]
  ...
  [1.         1.         1.         ... 0.92156863 0.94901961 0.98823529]
  [0.99607843 1.         1.         ... 1.         1.         1.        ]
  [0.98039216 0.98039216 0.98431373 ... 1.         0.99607843 0.99607843]]

 [[1.         1.         1.         ... 1.         0.94509804 0.97647059]
  [1.         1.         1.         ... 1.         1.         1.        ]
  [1.         1.         1.         ... 0.98823529 0.96470588 0.98431373]
  ...
  [1.         0.99607843 0.99607843 ... 1.         1.         1.        ]
  [0.99607843 1.         1.         ... 1.         1.         1.        ]
  [1.         1.         0.99607843 ... 1.         1.         1.        ]]

 ...
```

***build the ANN model.**

-convert data to 1d array.

-use activation function relu.

-out put layer with ten output.

In [12]:
```
model = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=(28,28)),
    tf.keras.layers.Dense(128, activation= 'relu'),
    tf.keras.layers.Dense(10) # linear activation function
])
```

***make a compile to ANN model.**

-In this step we add all the required settings for the model training.

-Loss Function: To measure models accuracy during training.

-loss=tf.keras.losses.a sparse Categorical Cross entropy(from_logits=true)

-Optimizer: To update the model weights based on the input data and loss function output.

-Metrics: Used to monitor the training and testing steps.

```
[13]: # The from_logits=True attribute inform the loss function that the output values generated by the model are not normalized, a.
      # Since softmax layer is not being added at the last layer, hence we need to have the from_logits=True to indicate the probabi

      model.compile(optimizer= 'adam',
                    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
                    metrics = ['accuracy'])
```

*Training the model with epochs 10.

```
n [14]: %%timeit -n1 -r1 # time required toexecute this cell once

        # To view in TensorBoard
        logdir = os.path.join("logs/adam", datetime.datetime.now().strftime("%Y%m%d-%H%M%S"))
        tensorboard_callback = tf.keras.callbacks.TensorBoard(logdir, histogram_freq=1)

        model.fit(train_images, train_labels, epochs= 10, callbacks = [tensorboard_callback])
```

```
Epoch 1/10
1875/1875 [==============================] - 5s 3ms/step - loss: 0.4979 - accuracy: 0.8250
Epoch 2/10
1875/1875 [==============================] - 5s 3ms/step - loss: 0.3752 - accuracy: 0.8655
Epoch 3/10
1875/1875 [==============================] - 5s 2ms/step - loss: 0.3379 - accuracy: 0.8771
Epoch 4/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.3130 - accuracy: 0.8856
Epoch 5/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.2960 - accuracy: 0.8910
Epoch 6/10
1875/1875 [==============================] - 5s 2ms/step - loss: 0.2823 - accuracy: 0.8962
Epoch 7/10
1875/1875 [==============================] - 6s 3ms/step - loss: 0.2679 - accuracy: 0.9002
Epoch 8/10
1875/1875 [==============================] - 5s 3ms/step - loss: 0.2586 - accuracy: 0.9037
Epoch 9/10
1875/1875 [==============================] - 5s 3ms/step - loss: 0.2485 - accuracy: 0.9087
Epoch 10/10
1875/1875 [==============================] - 5s 3ms/step - loss: 0.2395 - accuracy: 0.9106
1 loop, best of 1: 1min 22s per loop
```

*Accuracy of the model

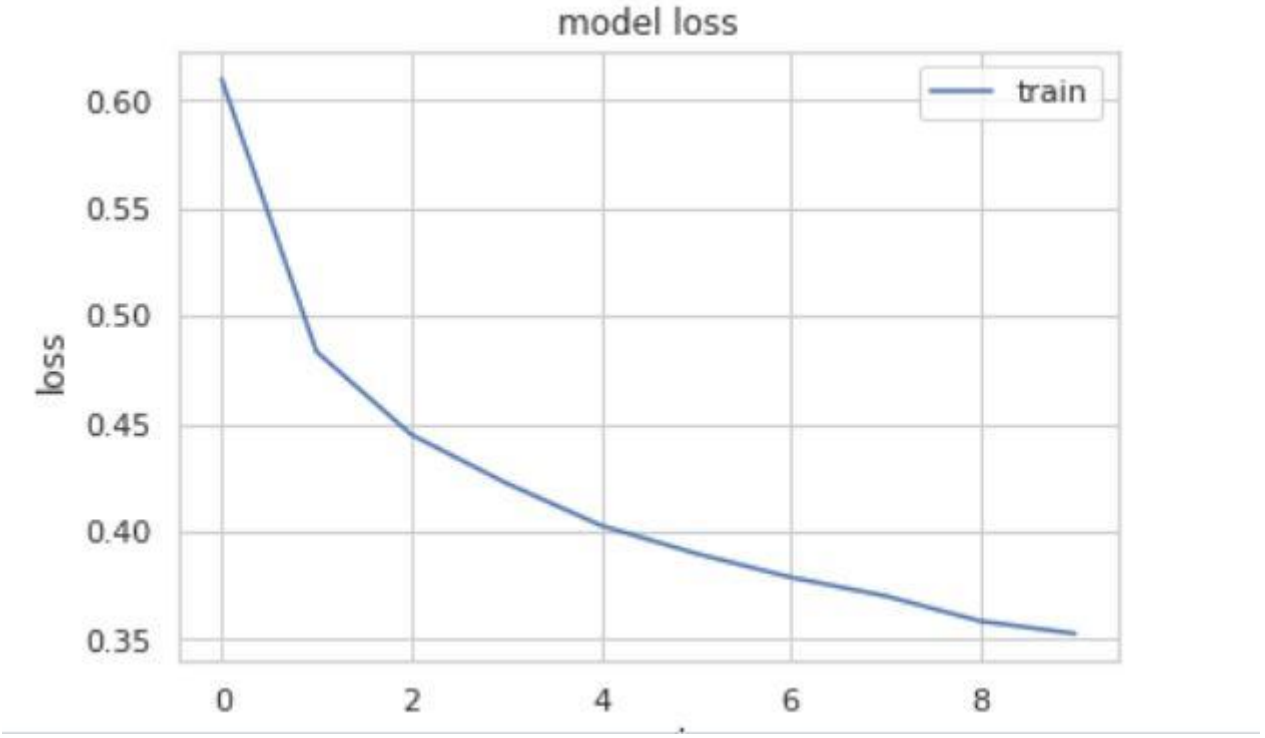
```
test_loss, test_acc = model.evaluate(X_test, y_test, verbose= 2)
print(f'\nTest accuracy: {test_acc}')
```

```
313/313 - 1s - loss: 0.3869 - accuracy: 0.8617 - 828ms/epoch - 3ms/step

Test accuracy: 0.8616999983787537
```

Result & plots

***Loss curve**



*Some testing Result

```
y_pred = c.predict(X_test)
for i in  range(len(y_pred)):
    print([getcode(np.argmax(y_pred[i])),getcode(y_test[i])])
```

```
['Bag', 'Bag']
['Bag', 'Bag']
['Bag', 'Bag']
['Sneaker', 'Bag']
['Bag', 'Bag']
['Bag', 'Bag']
['Bag', 'Bag']
['Bag', 'Bag']
['Bag', 'Bag']
['Sneaker', 'Bag']
['Bag', 'Bag']
['Bag', 'Bag']
['Bag', 'Bag']
```

```
['Sandal', 'Sandal']
['Sandal', 'Sandal']
['Sandal', 'Sandal']
['Sandal', 'Sandal']
['Sneaker', 'Sandal']
['Sandal', 'Sandal']
['Sandal', 'Sandal']
['Sandal', 'Sandal']
['Sandal', 'Sandal']
['Sandal', 'Sandal']
['Sneaker', 'Sandal']
['Sandal', 'Sandal']
['Sandal', 'Sandal']
```

```
['Sneaker', 'Ankle boot']
['Ankle boot', 'Ankle boot']
['Ankle boot', 'Ankle boot']
['Ankle boot', 'Ankle boot']
['Ankle boot', 'Ankle boot']
['Ankle boot', 'Ankle boot']
['Ankle boot', 'Ankle boot']
['Ankle boot', 'Ankle boot']
['Ankle boot', 'Ankle boot']
['Ankle boot', 'Ankle boot']
['Ankle boot', 'Ankle boot']
```

```
['Shirt', 'Shirt']
['Shirt', 'Shirt']
['Shirt', 'Shirt']
['Shirt', 'Shirt']
['Shirt', 'Shirt']
['Coat', 'Shirt']
['Dress', 'Shirt']
['Shirt', 'Shirt']
['Shirt', 'Shirt']
['Shirt', 'Shirt']
['Shirt', 'Shirt']
['Pullover', 'Shirt']
```

*Confusion matrix

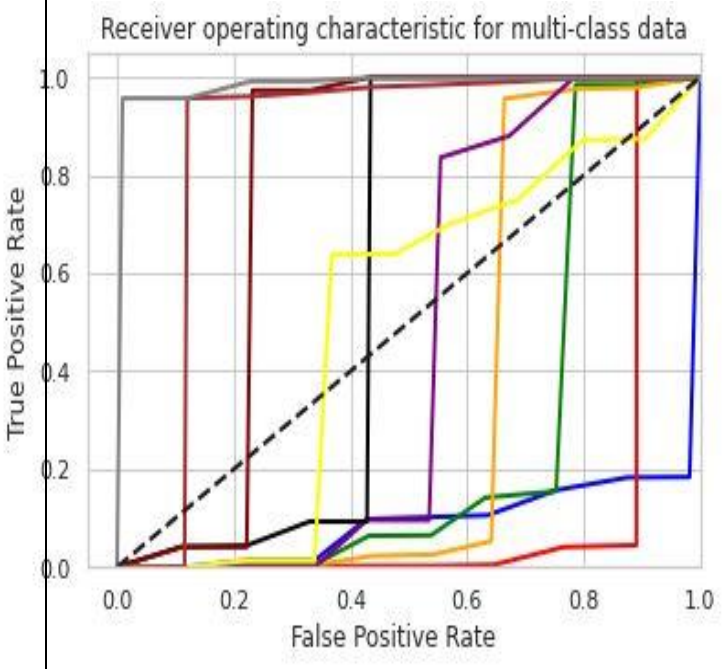
```
from sklearn.metrics import confusion_matrix, accuracy_score
print(np.argmax(y_pred[0]))
predicition=[]
for i in range(len(y_pred)):
  predicition.append(np.argmax(y_pred[i]))
cm = confusion_matrix(y_test, predicition)
print(cm)
accuracy_score(y_test, predicition)
```

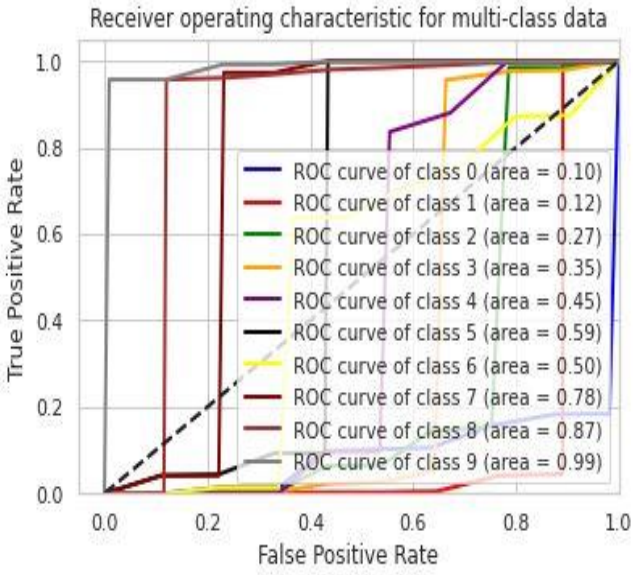
```
7
[[816   1  26  51   4   4  84   0  14   0]
 [  3 952   4  36   2   0   1   0   2   0]
 [ 15   0 830  13  77   1  58   0   6   0]
 [ 21   3  20 903  27   3  18   0   5   0]
 [  0   0 120  44 740   0  95   0   1   0]
 [  0   0   0   0   0 907   0  49   2  42]
 [126   2 122  50  61   1 624   0  14   0]
 [  0   0   0   0   0  27   0 932   0  41]
 [  2   1   7   6   4  11   8   4 956   1]
 [  1   0   0   0   0   7   0  35   0 957]]
0.8617
```

*ROC Curve

```
from sklearn.metrics import roc_curve
from sklearn import metrics
from itertools import cycle
y_pred= c.predict(X_test)
predicition=[]
for i in range(len(y_pred)):
  predicition.append(np.argmax(y_pred[i]))

fpr = dict()
tpr = dict()
roc_auc = dict()
lw=2
for i in range(10):
    fpr[i], tpr[i], _ = roc_curve(y_test, predicition,pos_label=i)
    roc_auc[i] = auc(fpr[i], tpr[i])
colors = cycle(['blue', 'red', 'green','orange','purple','black','yellow','maroon','brown','gray'])
for i, color in zip(range(10), colors):
    plt.plot(fpr[i], tpr[i], color=color, lw=2,
             label='ROC curve of class {0} (area = {1:0.2f})'
             ''.format(i, roc_auc[i]))
plt.plot([0, 1], [0, 1], 'k--', lw=lw)
plt.xlim([-0.05, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic for multi-class data')
plt.legend(loc="lower right")
plt.show()
```





****

7.Development platform.

Tools: - COLAB

Languages: - python

Libraries: - pandas – numpy – matplotlib – os – glob - cv2 – keras- tensorflow – sklearn-seaborn

8.Analysis, Discussion, and Future Work

After running the project, we notice some points

1-from the confusion matrix and the ROC

that the first five labels(T-shirt,Trouser,Pullover,Dress,Coat)

has a low ability for the prediction.

2-we notice two mistakes was made…….

A-the image of the dataset has low quality.

B-the images size is very small to make the model

notice the difference.

3-we notice that if we made epochs more than 10 the model will overfit and give us a bad result in testing.

Advantages

- Detect many types of clothes
- Good overall accuracy 86%
- Less than 1 second to detect object
- Few times training

Disadvantage

- Detect one object only

Conclusion

The disadvantages occur because of bad quality of images and small size in training dataset and in the future I will choose better dataset.