**PYTHON PROJECT**

**Final Report**

**TEAM 1**

**MEMBERS:**

VALLURU, ESWAR(27)

SIRASANAGANDLA, BHAVYA SRI(21)

NADIMPELLI, MANIDEEP(16)

**INTRODUCTION**

Visual classification of commercial products is a branch of the wider fields of object detection and feature extraction in computer vision, and, in particular, it is an important step in the creative workflow in fashion industries. Automatically classifying garment features makes both designers and data experts aware of their overall production, which is fundamental in order to organize marketing campaigns, avoid duplicates, categorize apparel products for e-commerce purposes, and so on. There are many different techniques for visual classification, ranging from standard image processing to machine learning approaches: this work, made by using and testing the aforementioned approaches, describes a real-world study aimed at automatically recognizing and classifying logos, stripes, colors, and other features of clothing, solely from final rendering images of their products. Specifically, both deep learning and image processing techniques, such as template matching, were used. The result is a novel system for image recognition and feature extraction that has a high classification accuracy and which is reliable and robust enough.

**Objectives:**

* To train a model in a way, such that it classifies the given fashion products.
* To build a website that takes in the gray scale image as input and communicates with model and displays the predicted product.

**Features:**

* A model which is trained to predict the class of given product
* A website that takes images and displays the predicted results.

**GIT Link:**

* **Model :** <https://github.com/Sbhavyarao/Python_Project>

**Technologies Used:**

* Python
* Flask
* HTML5
* CSS3
* JavaScript

**Python Concepts Used:**

* From Keras
  + Convolutional Neural Networks
  + Dense
  + MaxPooling
  + Early Stopping
* Flask
  + Requests
  + Render Templates
  + Request
* OpenCV2
  + ImRead
  + ImShow
  + ImResize

**DATASET:**

We have used Kaggle’s data set: Fashionmnist

**STEP 1:**

* We are building our model in google COLAB so for this we have uploaded the data set in to google COLAB.
* Code snippet for uploading data using read\_csv( ):

A screenshot of a cell phone

Description automatically generated

* We have loaded the train dataset into train dataframe(fashion\_train\_df) and test data set into test dataframe (fashion\_test\_df)

**STEP 2:**

Visualization of Dataset:

* The input data is given in the form of pixel values.
* The input images given to the model is resized to 28X28 size .
* The head and tail of the train and test data frames are shown in order to understand the input format.

A screenshot of a social media post

Description automatically generated

**STEP 3:**

* To know the size of the input and the classes
* Creating the training and testing numpy arrays

**A screenshot of a cell phone

Description automatically generated**

**STEP 4:**

* Take any random image from the training set and showing it and knowing its class and also showing all the classes.

A screenshot of a cell phone

Description automatically generated

* Also displaying the label for randomly selected images.

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A close up of a device

Description automatically generated

* Randomly selected images and displaying their class

**STEP 5:**

Scaled the training data and test data by dividing each pixel value by 255.0 in order to reduce the complexity and to get better accuracy.

A screenshot of a cell phone

Description automatically generated

**STEP 6:**

Building the model

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a social media post

Description automatically generated

We have got an accuracy of 95% and validation accuracy 92%. As the difference between both of them is small , we can say that the model is a good fit.

**STEP 7:**

Evaluating the model:

A screenshot of a cell phone

Description automatically generated

A picture containing screenshot

Description automatically generated

**STEP 8:**

* Confusion matrix and Heatmap
* Heatmap: A heat map is a data visualization technique that shows magnitude of a phenomenon as color in two dimensions. The variation in color may be by hue or intensity, giving obvious visual cues to the reader about how the phenomenon is clustered or varies over space.
* Confusion matrix: The confusion matrix is a 2 dimensional array comparing predictedcategory labels to the true label.

A screenshot of a cell phone

Description automatically generated

**STEP 9:**

* Classification Report
* A Classification report is used to measure the quality of predictions from a classification algorithm.

A screenshot of a social media post

Description automatically generated

**STEP 10:**

* We have saved the model using pickle library.

A screenshot of a social media post

Description automatically generated

**STEP 11:**

* Flask using Python to render HTML

from werkzeug.utils import secure\_filename  
import joblib  
from flask import Flask, request, render\_template  
import cv2  
import numpy as np  
  
# Define a flask app  
app = Flask(\_\_name\_\_)  
def process\_eval(imk):  
 output1 = cv2.resize(imk, (28,28))  
 output1 = output1.astype('float')  
 output1 /= 255.0  
 print(type(output1))  
 output1 = np.array(output1).reshape(-1, 28, 28, 1)  
 classifer = joblib.load("python.pk2")  
 x = classifer.predict\_classes(output1[[0], :])  
 if x[0] == 0:  
 result = "PREDICTED RESULT IS TSHIRT"  
 if x[0]==1:  
 result = "PREDICTED RESULT IS TROUSERS"  
 if x[0]==2:  
 result = "PREDICTED RESULT IS PULLOVER"  
 if x[0]==3:  
 result = "PREDICTED RESULT IS DRESS"  
 if x[0]==4:  
 result = "PREDICTED RESULT IS COAT"  
 if x[0]==5:  
 result = "PREDICTED RESULT IS SANDAL"  
 if x[0]==6:  
 result = "PREDICTED RESULT IS SHIRT"  
 if x[0]==7:  
 result = "PREDICTED RESULT IS SNEAKER"  
 if x[0]==8:  
 result = "PREDICTED RESULT IS BAG"  
 else:  
 result = "PREDICTED RESULT IS ANKLE SHOE"  
  
 return result  
  
@app.route('/', methods=['GET'])  
def index():  
 return render\_template('index.html')  
  
@app.route('/', methods=['GET', 'POST'])  
def handle\_form():  
 if request.method == 'POST':  
 file = request.files['file']  
 file.save(secure\_filename("save.jpeg"))  
 im=cv2.imread("save.jpeg")  
 result=process\_eval(im)  
 return render\_template('index.html',result=result)  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 app.run()

**STEP 12:**

* The above flask code will render this HTML
* <!DOCTYPE html>  
  <html lang="en">  
  <head>  
    
   <meta charset="utf-8">  
    
   <title>Upload Image</title>  
    
   <!-- Stylesheets -->  
   <link href="{{ url\_for('static', filename='style.css') }}" rel="stylesheet">  
    
   <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.7.2/jquery.min.js"></script>  
   <script src="../js/plupload.full.js"></script>  
   <script src="../js/jquery-progressbar.min.js"></script>  
   <script>  
   var loadFile = function(event) {  
   var image = document.getElementById('output');  
   image.src = URL.createObjectURL(event.target.files[0]);  
  };  
    
   </script>  
    
  <meta name="robots" content="noindex,follow" />  
   <style>  
   .container {  
   position: absolute;  
   top: 0;  
   right: 0;  
   bottom: 0;  
   left: 0;  
   background-image: url('../static/bg.jpeg');  
   background-repeat: no-repeat;  
   background-attachment: fixed;  
   background-size: cover;  
  }  
   </style>  
  </head>  
    
  <body class="container">  
    
  <div class="upload-form" id="uploader">  
    
   <h1>Fashion Image Classifier</h1>  
    
   <!-- Select & Upload Button -->  
   <div>  
   <form action="" method="post" enctype="multipart/form-data">  
   <div><input type="file" name="file" onchange="loadFile(event)"></div>  
   <div class="img-dec"><p><img id="output" width="400" /></p></div>  
   <br>  
   <div class="top-pad">  
   <input type="submit" value="Upload" class="top-pad">  
   </div>  
   <div>  
   {{result}}</div>  
   </form>  
   </div>  
    
   <!-- File List -->  
   <div id="filelist" class="cb"></div>  
    
   <!-- Progress Bar -->  
   <div id="progressbar"></div>  
  </div>  
    
  </body>

</html>

* The CSS code is shown below

.top-pad{  
 margin-top: 35%; !important;  
 margin-left: 72%; !important;  
 position: absolute;  
}  
  
.img-dec{  
 width: 50%;  
 float: left;  
}  
  
.upload-form,  
.upload-form div,  
.upload-form span,  
.upload-form input,  
.upload-form a,  
.upload-form h1,  
.upload-form p {  
 /\*margin: 0;\*/  
 padding: 0;  
 border: none;  
 outline: none;  
}  
  
/\* Form Container \*/  
.upload-form {  
 position: relative;  
 z-index: 100;  
 cursor: default;  
 width: 600px;  
 height: 600px;  
 padding: 25px 25px 65px 25px;  
 margin-left: 26%;  
 -webkit-border-radius: 3px;  
 -moz-border-radius: 3px;  
 border-radius: 3px;  
 background: white;  
}  
  
.container {  
 position: absolute;  
 top: 0;  
 right: 0;  
 bottom: 0;  
 left: 0;  
 background-image: url('bg.jpeg');  
}  
  
input[type=button], input[type=submit]{  
 background-color: black;  
 border: none;  
 color: white;  
 padding: 16px 32px;  
 text-decoration: none;  
 /\*margin: 4px 2px;\*/  
 cursor: pointer;  
  
}  
  
::-webkit-file-upload-button {  
 background: black;  
 border: none;  
 color: white;  
 padding: 16px 32px;  
}  
  
  
/\* Clear Floats \*/  
.upload-form .cb { clear: both; }  
  
/\* Form Title \*/  
.upload-form h1 {  
 font-family: 'Trebuchet MS', sans-serif;  
 font-weight: bold;  
 text-transform: uppercase;  
 font-size: 18px;  
 text-shadow: 0px 1px 1px rgba(0,0,0, .65);  
 padding-bottom: 2px;  
 border-bottom: 1px solid #7a7b80;  
}  
  
/\* Replace Title with Image \*/  
.upload-form h1.replace-text {  
 display: block;  
 width: 100%;  
 height: 17px;  
 padding-bottom: 7px;  
 text-indent: -9999px;  
 background: url(../img/title.png) no-repeat left center;  
}  
  
  
/\* Paragraph \*/  
.upload-form p {  
 padding: 10px 0;  
 font-family: sans-serif;  
 font-size: 12px;  
 color: #a4a5a8;  
}  
  
/\* Select & Upload Buttons \*/  
.upload-form .button {  
 display: inline-block;  
 height: 30px;  
 margin: 15px 0;  
  
 font-family: sans-serif;  
 font-size: 14px;  
 color: #777;  
  
 text-decoration: none;  
 text-transform: uppercase;  
 text-align: center;  
 line-height: 30px;  
  
 background: #ffffff;  
 background: -webkit-linear-gradient(top, #ffffff 0%, #f6f6f6 100%);  
 background: -moz-linear-gradient(top, #ffffff 0%, #f6f6f6 100%);  
 background: -o-linear-gradient(top, #ffffff 0%, #f6f6f6 100%);  
 background: -ms-linear-gradient(top, #ffffff 0%, #f6f6f6 100%);  
 background: linear-gradient(top, #ffffff 0%, #f6f6f6 100%);  
}  
  
.upload-form .button:hover {  
 background: #f2f2f2;  
 background: -webkit-linear-gradient(top, #f2f2f2 0%, #eaeaea 100%);  
 background: -moz-linear-gradient(top, #f2f2f2 0%, #eaeaea 100%);  
 background: -o-linear-gradient(top, #f2f2f2 0%, #eaeaea 100%);  
 background: -ms-linear-gradient(top, #f2f2f2 0%, #eaeaea 100%);  
 background: linear-gradient(top, #f2f2f2 0%, #eaeaea 100%);  
}  
  
.upload-form #pickfiles {  
 width: 99px;  
 float: left;  
 border-right: 1px solid #999;  
  
 -webkit-border-radius: 3px 0 0 3px;  
 -moz-border-radius: 3px 0 0 3px;  
 border-radius: 3px 0 0 3px;  
}  
  
.upload-form #uploadfiles {  
 width: 100px;  
 float: right;  
  
 -webkit-border-radius: 0 3px 3px 0;  
 -moz-border-radius: 0 3px 3px 0;  
 border-radius: 0 3px 3px 0;  
}

**STEP 13:**

* Executing the Python file

A screenshot of a cell phone

Description automatically generated

**STEP 14:**

* Browse the URL from STEP 12
* URL: <http://127.0.0.1:5000>

A close up of a computer

Description automatically generated

* Upload a gray scale image.
* In this case we have taken a bag image.

A screenshot of a computer

Description automatically generated

* The predicted result by our model is also a bag.

A close up of a computer

Description automatically generated

A close up of a computer

Description automatically generated

**CHALLENGES FACED:**

1. We were in dilemma how to compare the actual and predicted values, but finally we ended up plotting Confusion matrix.
2. Selection of Hyperparameters
3. We also faced an issue of overtime in training our model while introducing more layers.
4. We encountered a situation where the epochs were running but there is no significant change in the accuracy, so we used the concept of Early Stopping and resolved the issue.

**Future Scope :**

In future if this application is deployed on any cloud service then it would be useful for classifying any fashion related image.

**References:**

<https://www.kaggle.com/zalando-research/fashionmnist>

<https://keras.io/>

<https://www.tensorflow.org/tutorials/images/classification>

<https://towardsdatascience.com/wtf-is-image-classification-8e78a8235acb>

<https://www.youtube.com/watch?v=j-3vuBynnOE>

<https://www.youtube.com/channel/UCWN3xxRkmTPmbKwht9FuE5A>

<https://www.tensorflow.org/>