INTRODUCTION

1.1 Overview

We have created a webpage which will recommend the users based on their preferences. Our Chatbot will get the info from the user and recommends dress based on their input. We have provided a feature where users can upload their favorite dress and it will be processed by our custom Deep Learning model which is built using CNN. After analyzing the picture, the custom model will pull the clothes from our database which are similar to the clothes which the user uploaded. We didn't stop with that, We even made a virtual try on feature where the user can see the whether the dress will be suitable for them or not.

1.2 Purpose

In online shopping most of the time is wasted in finding the right pair of clothes. So we implemented a visual search where users can search clothes based on their favourite clothes. So by doing visual search users need not have to scroll 2 or 3 pages to find what they are looking for. The clothes they are looking for will be in the top of their search results. Let's be honest everyone in this world is unique. So the dress which is suitable for one person may not be suitable for others or their fashion sense might be different. So we have implemented virtual try-on feature where the users can see how the dress will look like when they wear them. And for the people who are confused with what kind of dress will be suitable for different occasion can ask our chat bot (G root) for suggestions.

2 LITERATURE SURVEY

2.1 Existing problem

It is relatively difficult to express something in words than showing it as a picture. So the search results are inaccurate

In shops user can try on the clothes to know whether the clothes suits their style or not. But in existing online Fashion app it was not possible.

In shops there will be people to suggest some clothes when we were confused with our choice. But in existing online Fashion app it was not available.

2.2 Proposed solution

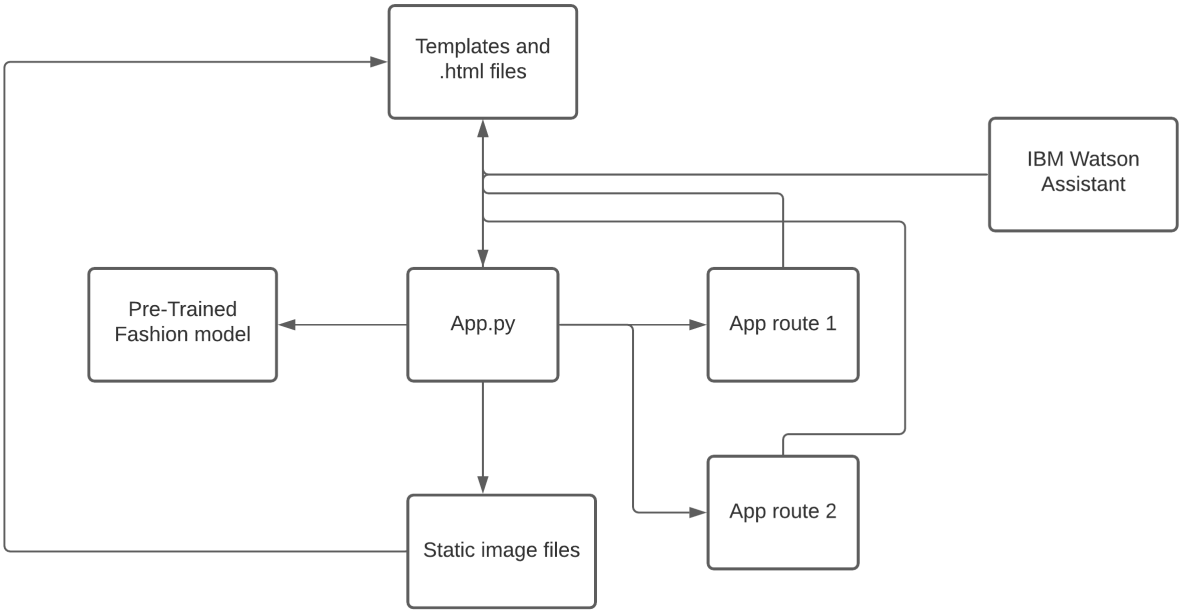
Building a visual search which will use the custom Deep Learning model to show results based on file uploaded by the user.

We have implemented a virtual try-on feature, where users can check whether the dress is suitable from the clothes we suggest.

To help the users on their queries we have implemented a chatbot created using IBM Watson Studio.

3 THEORITICAL ANALYSIS

3.1 Block diagram



Diagrammatic overview of the project

3.2 Hardware / Software designing

Hardware requirements :-

- A 16 threaded processor to train the CNN in 20 minutes of time.

Software requirements :-

- IBM-Watson-machine-learning
- TensorFlow
- Pandas, numpy
- OpenCV
- ngrok
- Flask
- Jupyter Notebook

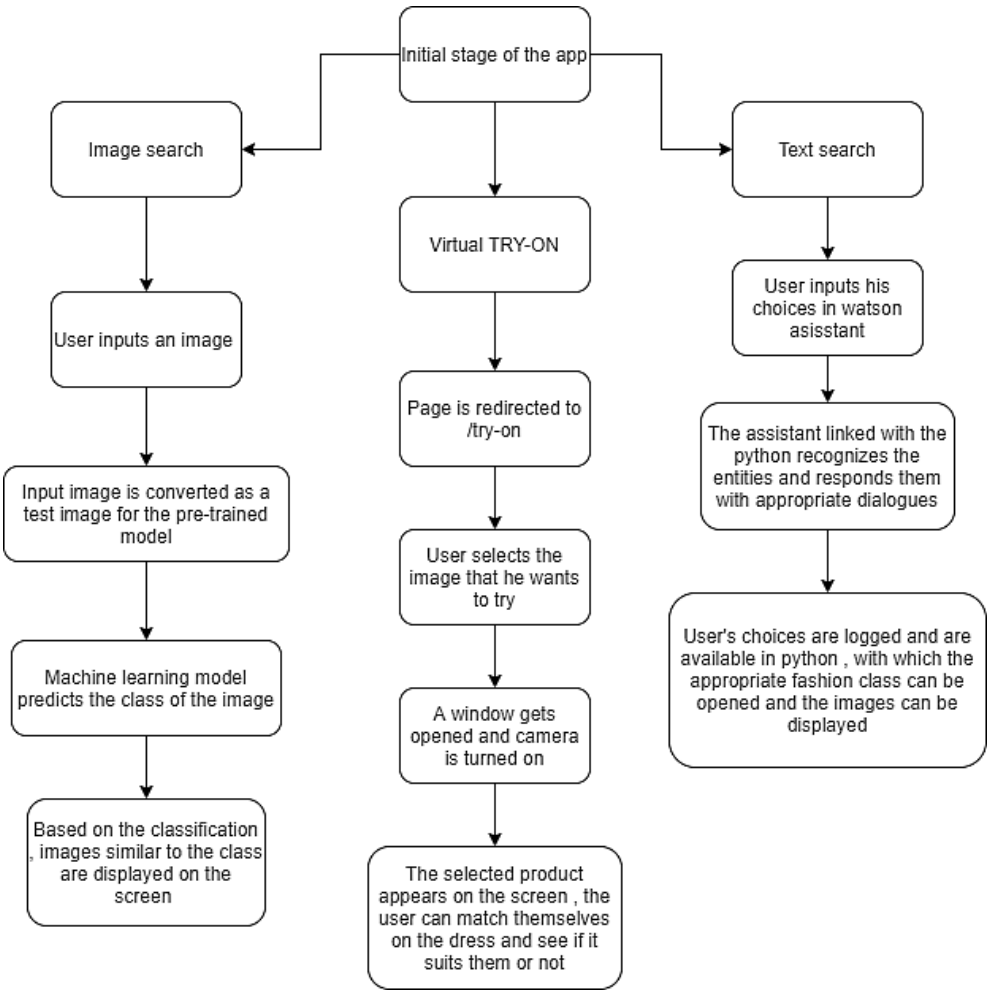
4 data-based INVESTIGATIONS

- When training the model we got to know that the accuracy of the model does not only depend upon the number of classes , but also the number of images in each class, Therefore a dataset has been newly constructed from the given dataset that fits the model.

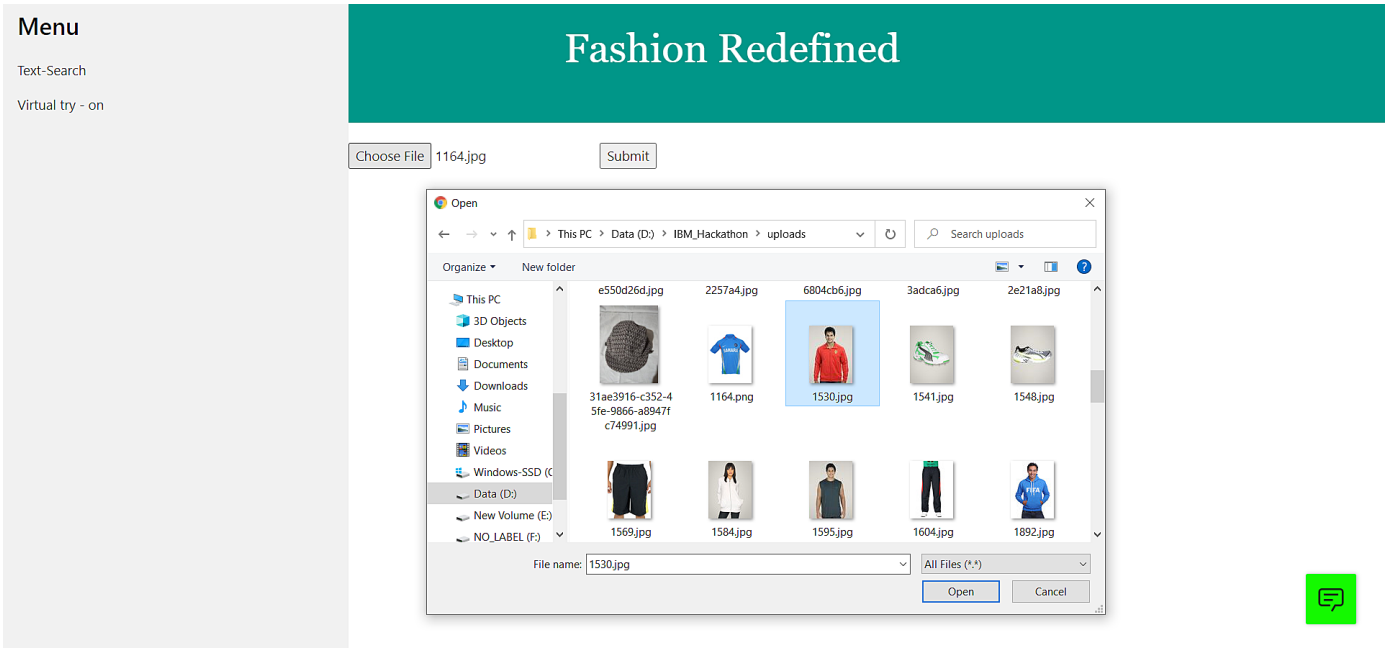
- Another analysis we had is that, Integration of Watson assistant and our main files can be made using a server named ngrok which provides a webhook URL which can be given to the assistant thereby requests to assistant are logged in ngrok and accused through flask.

- Whenever a POST request is made from the assistant , the control of the app directly goes to its default route that is '/' and doesn't get out. Later we realized that all the post requests from Watson assistant alone gets to the main function.So we adapted a different method of displaying products in case of text search using the bot.

5 FLOWCHART


Initial stage of the app
Image search
Virtual TRY-ON
Text search
User inputs an image
Page is redirected to /try-on
User inputs his choices in watson asisstant
Input image is converted as a test image for the pre-trained model
User selects the image that he wants to try
The assistant linked with the python recognizes the entities and responds them with appropriate dialogues
Machine learning model predicts the class of the image
A window gets opened and camera is turned on
User's choices are logged and are available in python , with which the appropriate fashion class can be opened and the images can be displayed
Based on the classification , images similar to the class are displayed on the screen
The selected product appears on the screen , the user can match themselves on the dress and see if it suits them or not

6 RESULT


Menu
Text-Search
Virtual try - on
Fashion Redefined
Choose File
1164.jpg
Submit
Open
This PC
Data (D:)
IBM_Hackathon
uploads
Search uploads
Organize
New folder
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Windows-SSD (C
Data (D:)
New Volume (E:)
NO_LABEL (F:)
e550d26d.jpg
2257a4.jpg
6804cb6.jpg
3adca6.jpg
2e21a8.jpg
31ae3916-c352-45fe-9866-a8947fc74991.jpg
1164.png
1530.jpg
1541.jpg
1548.jpg
1569.jpg
1584.jpg
1595.jpg
1604.jpg
1892.jpg
File name:
1530.jpg
All Files (*.*)
Open
Cancel

Menu

Text-Search

Virtual try - on

Fashion Redefined

Choose File | No file chosen | Submit





Menu

Text-Search

Virtual try - on

Fashion Redefined




Groot at your service !!!

Get me some shorts ..

Shorts it is !!!

Type something...

Built with IBM Watson®


Image
FPS: 30

7 ADVANTAGES

- User's are no longer in need to search for products by typing their names, instead an image containing the product will take them to the place they want to go, thereby saving their time.

- The virtual try-on feature that is implemented makes the trial of cloths easier by just augmenting the image over our body, thereby making it look like a real trial .

- A user friendly bot responds the user's within seconds and clears their queries.

DISADVANTAGES

- As calculating height of a user is not possible even in augmentation , it has to be carefully selected by the users .

- The virtual try on feature needs access to the user's video device if the user does not give appropriate permissions they cannot use the feature.

8 APPLICATIONS

The idea of this application can be applied in any E-Commerce projects that involve products that can virtually tried on (only for the virtual try-on part). All the other components of project can eb applied to any feild of E-Commerce platform Eg; A sweet product can be found by the users by just clicking an image of the product and uploading it and so-on.

9 CONCLUSION

A fashion recommendation app was built using flask, tensorflow and OpenCV which has features such as image search , text search and Virtual try on. The dataset used was clothing dataset(small) from kaggle which contains 10 classes and images in each folder which were used for displaying the results. IBM's Watson assistant was used to create a chatbot in which the user can give his/her inputs and get replies from the bot as well as results in the browser. ngrok server is used to log the post requests from the IBM Chatbot that is been integrated in flask , which gives flask the access to user's inputs and thereby displaying products based on them.

10 FUTURE SCOPE

- The virtual-try on feature can be made even more user-friendly , we can build a feature which detects the body exactly even when the user is in motion, thereby the user no need to align himself with the dress. the dress will be automatically augmented over the detected body.

- A voice search feature can be added using Watson speech to text service thereby making user's product search even more easier.

- An e-kart can be added so that the user can track the products that he liked the most or similar products in the store.

REFERENCES :-

https://www.youtube.com/watch?v=aAT3qALIQXM&t=3s

https://youtu.be/Tr82IMw7BZw

https://www.youtube.com/watch?v=ikoL1FT6i4s

https://www.youtube.com/embed/BzouqMGJ41k

https://www.youtube.com/watch?v=4G2qMhhAg0c&t=41s

https://cloud.ibm.com/apidocs/assistant/assistant-v2?code=python

https://stackoverflow.com/questions/18249949/python-file-object-to-flasks-filestorage

https://note.nkmk.me/en/python-os-basename-dirname-split-splitext/

APPENDIX :-

Source Code :-

App.py :-

```
import numpy as np

import os

from tensorflow.keras.models import load_model
```

```
from tensorflow.keras.pre processing import image

import tensorflow as tf

global graph,preds

graph = tf.Graph()


from flask import Flask, request, render_template, redirect


from werkzeug.utils import secure_filename

from gevent.pywsgi import WSGIServer

from os import listdir

from os.path import isfile, join

import cvzone

import cv2

import json

global payload

import base64




app = Flask(__name__,template_folder='Templates')

model = load_model("FashionF2.h5")



@app.route('/',methods = ["POST","GET"])

def main1():

        onlyfiles = []
```

```
if request.method == "GET":
        return render_template('ex.html',onlyfiles=onlyfiles)
if request.method == "POST":
        payload = request.json
        payload['name'] = 'Hii'
        key1 = 'wish'
        key2 = 'enquire'
        p = 0
        if key2 in payload:
                if payload['enquire'] == 'shirts':
                        p = 5
                        print("User wants shirts !!!")
                elif payload['enquire'] == 'Hoodies':
                        p = 3
                        print("User wants pants !!!")
                elif payload['enquire'] == 'pants':
                        p = 4
                        print("User wants pants !!!")
                elif payload['enquire'] == 'Full hand T-shirts':
                        p = 2
                        print("User wants pants !!!")
                elif payload['enquire'] == 'footwears':
                        p = 6
                        print("User wants pants !!!")
                elif payload['enquire'] == 'cap':
                        p = 1
```

```
                print("User wants pants !!!")
            elif payload['enquire'] == 'Skirts':
                p = 8
                print("User wants pants !!!")
            elif payload['enquire'] == 'T-shirts':
                p = 9
                print("User wants pants !!!")
            elif payload['enquire'] == 'tops':
                p = 0
                print("User wants pants !!!")
            elif payload['enquire'] == 'Half hand T shirts':
                p = 9
                print("User wants pants !!!")
            elif payload['enquire'] == 'Shorts':
                p = 7
                print("User wants pants !!!")
        onlyfiles = [f for f in listdir('dress-1/' + str(p) + '/') if isfile(join('dress-1/' + str(p) + '/', f))]
        print(onlyfiles[0])
        with open('dress.txt', 'w') as f:
            for item in onlyfiles:
                f.write("%s\n" % item)
        return onlyfiles[0]

else:
        print(request.data)
        return "200"
```

```
@app.route('/try-on', methods = ['GET','POST'])
def try_on():
	if request.method == 'GET':
		return render_template('base.html')
	if request.method == 'POST':
		f = request.files['file']
		print("current path")
		basepath = os.path.dirname(__file__)
		print("current path",basepath)
		filepath = os.path.join(basepath,'uploads',f.filename)
		print("uplaod folder is ",filepath)
		f.save(filepath)
		cap = cv2.VideoCapture(0)
		cap.set(3,1920)
		cap.set(4,1080)
		success, img = cap.read()
		print(f)
		imgFront = cv2.imread("uploads/" + f.filename, cv2.IMREAD_UNCHANGED)
		imgFront = cv2.cvtColor(imgFront, cv2.COLOR_BGR2BGRA)
```

```
                print(imgFront.shape)


                hf, wf, cf = imgFront.shape
                hb, wb, cb = img.shape


                fpsReader = cvzone.FPS()


                while True:
                        success, img = cap.read()
                        imgResult = cvzone.overlayPNG(img, imgFront, [350, 200])
                        _, imgResult = fpsReader.update(imgResult)
                        cv2.imshow("Image", imgResult)
                        key = cv2.waitKey(1)
                        if key%256 == 27:
                                break
                cap.release()
                cv2.destroyAllWindows()
                return render_template('ex.html')


if __name__ == '__main__':
        app.run(debug = True)
```