22AIE201 Fundamentals of AI

A project - Submitted by

Group 10

AIE - BATCH B

Name	Roll Number
SAI YESHWANTH	(CB.SC.U4AIE23132)
VIJAY SANTHOSH	(CB.SC.U4AIE23157)
YASHWANTH P T	(CB.SC.U4AIE23164)
MANOGNA CHALLA	(CB.SC.U4AIE23175)

PROJECT TOPIC :

FASHION RECOMMENDATION SYSTEM.

As part of the 3rd Semester project
Department of Artificial Intelligence
AMRITA VISHWA VIDYAPEETHAM
COIMBATORE - 641 112 (INDIA)
NOVEMBER 2024

DECLARATION

We hereby declare that the project work entitled, *FASHION RECOMMENDATION SYSTEM* , submitted to Dr. Abhishek S, Assistant Professor, CEN, is our own work, based on our personal study and research. We have acknowledged all materials and sources used in its preparation, whether they be books, articles, reports, lecture notes, or any other kind of document, electronic or personal communication.

We also certify that this project has not previously been submitted for assessment in any academic capacity, and that we have not copied, in part or whole, or otherwise plagiarized the work of other persons. We confirm that we have identified and declared all possible conflicts that we may face.

AIE Batch B

Group 10

Place:Coimbatore, Tamil Nadu, India

Date: 16 November 2024

ACKNOWLEDGEMENT

We would like to express our sincere appreciation to all those who contributed to the successful completion of this project. This endeavour would not have been possible without the collective efforts, dedication, and expertise of the following individuals.

We extend our gratitude to each member of the project team for their hard work, collaboration, and commitment to excellence. Their diverse skills and perspectives greatly enriched the project and ensured its successful execution.

We would like to acknowledge Dr. Abhishek S, Assistant Professor, CEN, for her collaboration and contributions to specific aspects of the project. His expertise and cooperation significantly enhanced the overall quality of our work.

We are grateful for the support received from Amrita Vishwa Vidyapeetham. The resources, facilities, and conducive environment provided by the institution were instrumental in the successful execution of our project.

To our friends and family, who stood by us with patience and encouragement, we extend our heartfelt appreciation. Your understanding and support were invaluable during the demanding phases of this project.

In conclusion, each individual and entity mentioned above played a crucial role in the realization of this project. Our gratitude extends to everyone who contributed in various capacities, and we are proud to have worked with such a dedicated and talented team.

TABLE OF CONTENTS

1. Abstract

The fashion industry increasingly relies on technology to enhance user experience by offering personalized recommendations. In this project, we present a fashion recommendation system designed to analyze user-provided images of fashion materials and suggest complementary fashion items. The system is equipped with a user-friendly web interface developed using Streamlit, enabling users to drag and drop images for analysis conveniently.

At the core of this system lies a machine learning model trained to recognize various patterns, textures, and styles from the uploaded images. Once an image is uploaded, the model processes it to extract key features and identifies similar attributes within a curated dataset. Based on this analysis, the system provides tailored fashion recommendations, such as clothing items, accessories, or style combinations, ensuring relevance to the material uploaded.

Our approach emphasizes accessibility and simplicity, making it suitable for both casual users and professionals in the fashion industry. This project demonstrates how technology can seamlessly integrate into the creative domain, offering innovative solutions for personalization in fashion. With the capability to deliver recommendations in real-time, the system stands out as a powerful tool for enhancing decision-making in fashion choices, catering to diverse user needs and preferences

2. Introduction

Fashion plays a significant role in expressing individual identity and personality through clothing choices. It serves as a non-verbal communication medium that reflects various aspects such as faith, social status, and attitude towards life. In recent years, technological advancements have profoundly influenced consumer behavior, enabling people to track global fashion trends and tailor their clothing preferences. Factors such as demographics, geographic location, individual preferences, and culture significantly influence fashion choices. These preferences, coupled with other attributes like season and age, form a foundation for developing effective fashion recommendation systems (FRSs).

FRSs are innovative tools designed to enhance consumer shopping experiences by providing personalized recommendations based on browsing history, purchase history, or uploaded images. E-commerce platforms like Amazon, eBay, and social media sites like Pinterest and Instagram have integrated such systems to improve user engagement and drive sales. Research has demonstrated that these systems reduce transaction costs for consumers while increasing revenue for retailers. Furthermore, advancements in deep learning, image recognition, and recommendation algorithms have expanded the scope and efficiency of FRSs.

2.1. Project Overview

This project aims to develop a **Fashion Recommendation System** that combines state-of-the-art technologies to recommend visually similar clothing items. The system leverages a pre-trained **ResNet50** deep learning model to extract features from uploaded images. Using the **Nearest Neighbors** algorithm, it identifies and suggests visually similar items from a dataset of fashion images. By employing advanced machine learning techniques, the project emphasizes efficiency and accuracy in providing recommendations.

The system is built with an intuitive interface using **Streamlit**, allowing users to upload images effortlessly. Once an image is uploaded, the system extracts its features, compares them with a pre-trained dataset, and provides recommendations. This approach bridges the gap between consumer preferences and available fashion products, making it an effective tool for modern e-commerce platforms.

2.2. Objectives of the Project

The primary objective of this project is to develop a robust **Fashion Recommendation System** that provides personalized clothing recommendations based on visual similarity. By leveraging advanced deep learning and machine learning techniques, the system aims to achieve the following:

1. **Enhance the Shopping Experience**: Offer consumers tailored recommendations by analyzing uploaded images and matching them with visually similar items in a pre-existing dataset.

2. **Bridge Consumer Preferences with Available Options**: Enable users to find fashion items that align with their unique style and preferences, thus improving accessibility and satisfaction.

3. **Utilize Advanced Technology**: Employ a pre-trained **ResNet50** model for feature extraction and the **Nearest Neighbors** algorithm for similarity matching, ensuring high accuracy and efficiency.

4. **Simplify User Interaction**: Design a user-friendly interface using **Streamlit** that allows seamless uploading of images and quick access to recommendations.

2.3. Significance and Applications

The ability of FRSs to analyze consumer preferences and provide personalized recommendations has revolutionized the fashion industry. By incorporating advanced filtering techniques and machine learning models, these systems address challenges such as understanding diverse consumer preferences and predicting fashion trends. Retailers and designers benefit from these insights, enabling them to cater to global audiences effectively. Moreover, FRSs enhance the accessibility and navigability of e-commerce platforms, contributing significantly to consumer satisfaction.

This project aligns with the ongoing research efforts in the field by utilizing image parsing, feature recognition, and similarity-based recommendations. By combining an efficient backend with a user-friendly frontend, it highlights the potential of technology to transform the fashion industry while offering a roadmap for future innovations in personalized recommendation systems.

3. Methodology

The Fashion Recommendation System employs a systematic approach to deliver accurate and efficient recommendations. Below is the detailed methodology:

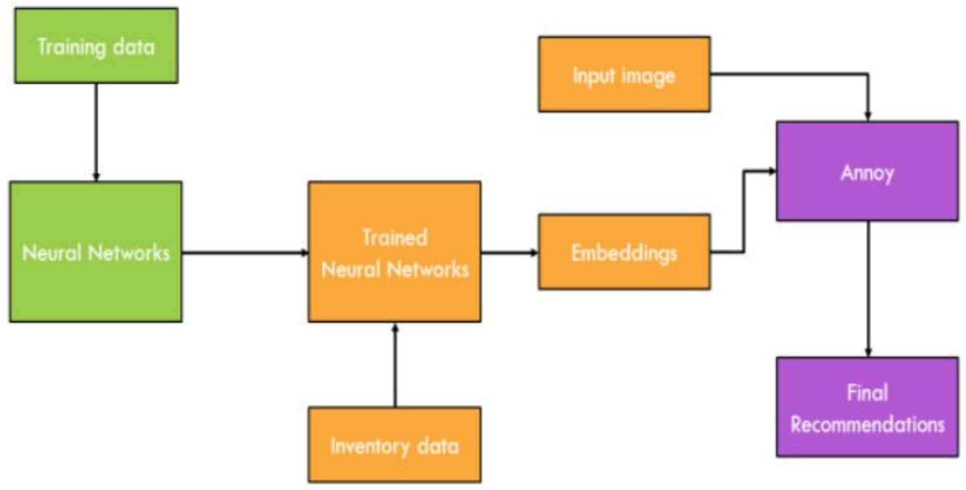


Figure 1. Block diagram of proposed system

3.1. Data Collection and Preprocessing:

 Dataset Preparation:

- A curated dataset of diverse fashion items is gathered from reliable sources to represent various styles, categories, and trends.
- The dataset includes high-resolution images of clothing, accessories, and footwear, ensuring comprehensive coverage of user preferences.

 Image Resizing and Normalization:

- Each image is resized to a standardized resolution of **224x224 pixels**, which is the input size required by the ResNet50 model.
- This resizing ensures uniformity across the dataset, enabling consistent feature extraction while maintaining the visual quality of the images.

□ **Organizing the Dataset**:

- The dataset is systematically categorized into distinct folders or labels based on product types (e.g., shirts, dresses, shoes).
- Proper organization facilitates easy access during training and recommendation processes.

□ **Feature Embedding Generation**:

- Features are extracted from each image using the pre-trained **ResNet50 model**, which is designed to capture rich and meaningful visual representations.
- The model processes the images through its convolutional layers to identify critical features like textures, patterns, and shapes.

□ **Embedding Storage**:

- The extracted features are compressed into high-dimensional vectors using **Global Max Pooling** to reduce data size while retaining essential information.
- These embeddings are normalized for efficient comparison, stored in a serialized format (e.g., .pkl files), and used as a reference for real-time recommendations.

3.2. Feature Extraction

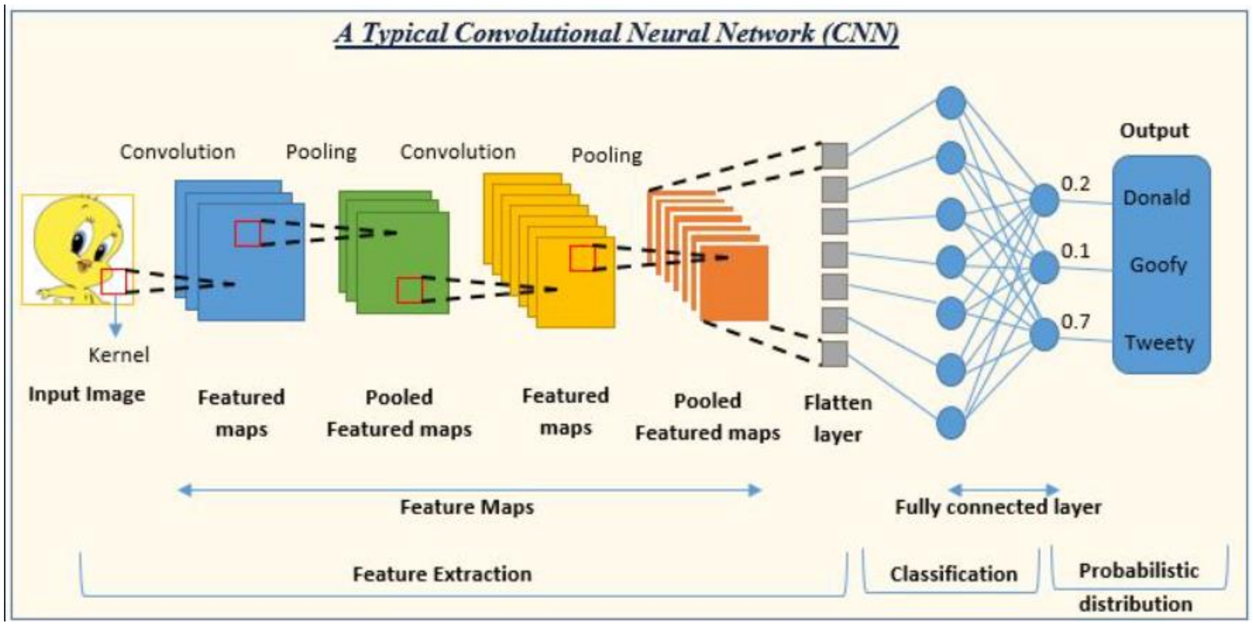


FIGURE:2

Utilizing a Pre-Trained ResNet50 Model:

- A pre-trained **ResNet50 model** is employed as the backbone for feature extraction.
- The fully connected top layers are removed to focus on extracting deep visual features from the convolutional layers, which capture intricate patterns, textures, and shapes present in the images.

Dimensionality Reduction with Global Max Pooling:

- After the convolutional operations, **Global Max Pooling** is applied to condense the feature maps into a lower-dimensional vector representation.
- This step ensures that only the most significant and informative features are retained, reducing computational complexity without compromising the model's ability to differentiate between images.

Feature Normalization for Enhanced Comparison:

- The extracted feature vectors are normalized to a unit norm, ensuring uniformity and scale-invariance.
- This normalization enhances the accuracy of similarity comparisons during the recommendation process by mitigating variations caused by scale differences in raw features.

3.3. Recommendation Algorithm

1. **Identifying Closest Matches with Nearest Neighbors**:
   - The **Nearest Neighbors algorithm** is employed to find fashion items most similar to the input image based on the extracted feature vectors.
   - **Euclidean distance** serves as the similarity metric, ensuring that the closest matches are determined by the minimal distance between feature vectors in the high-dimensional embedding space.
2. **Setting a Fixed Number of Recommendations**:
   - A predetermined number of neighbors (e.g., 5 or 6) is selected to provide consistent and relevant recommendations for every query.
   - This approach ensures a balanced output, giving users a variety of options while maintaining computational efficiency.

3.4. User Interface Development

Building a User-Friendly Interface:

- A sleek and intuitive interface is designed using **Streamlit**, ensuring simplicity and accessibility for users of all backgrounds.
- The platform leverages its interactive capabilities to create a seamless experience for engaging with the recommendation system.

Image Upload and Display:

- Users can easily upload images via a **drag-and-drop feature**, enhancing convenience and usability.
- Once an image is uploaded, it is immediately displayed on the interface for **confirmation and verification**, allowing users to proceed with confidence.

Visual Presentation of Recommendations:

- The recommended fashion items are presented alongside the uploaded image in a **visually appealing grid layout**, making it easy for users to compare and evaluate the suggestions.

- This structured design prioritizes clarity and aesthetics, ensuring an enjoyable and efficient user experience.

3.5. End-to-End Workflow

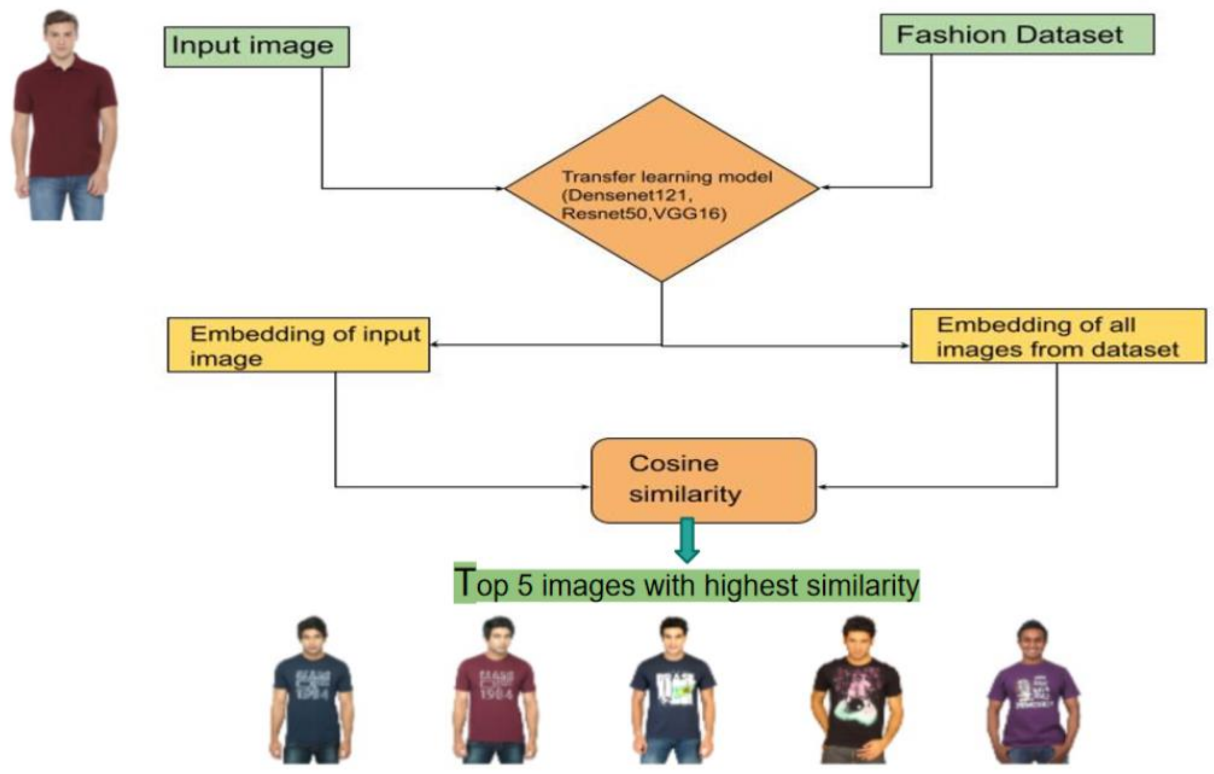

Input image
Fashion Dataset
Transfer learning model (Densenet121, Resnet50,VGG16)
Embedding of input image
Embedding of all images from dataset
Cosine similarity
Top 5 images with highest similarity

FIGURE:3

Image Upload and Storage:

- The user uploads an image through the interface, which is temporarily saved in a designated folder for processing.

Feature Extraction:

- The uploaded image is processed through the **pre-trained ResNet50 model pipeline**, extracting its deep feature representation.

Feature Comparison:

- The extracted features are normalized and compared against the **precomputed feature embeddings** of the dataset using the **Nearest Neighbors algorithm**.

Recommendation Retrieval:

- The system identifies the closest matching fashion items based on feature similarity and retrieves a set of top recommendations.

Display Recommendations:

- The uploaded image is displayed alongside the recommended fashion items, providing a clear and visually engaging result to the user.

3.6. Evaluation and Testing

System Testing with Diverse Inputs:

The system is rigorously tested using a wide range of input images, including different clothing styles, patterns, and categories, to assess its **recommendation accuracy** and **reliability**.

Performance Metrics:

Key performance metrics such as precision, recall, and retrieval time are analyzed to evaluate the system's effectiveness in delivering accurate recommendations.

User Feedback Integration:

Feedback from users is actively collected to understand their experience with the recommendations and interface.

Insights from user feedback are leveraged to **fine-tune the recommendation algorithm**, ensuring better alignment with user preferences and expectations.

Interface Improvements:

Based on testing outcomes, adjustments are made to enhance the **user interface's intuitiveness** and visual appeal, further improving the overall user experience.

4. Implementation

Libraries Used

For this project, several Python libraries were employed to streamline the development of this project:

TensorFlow
Purpose**:** An open-source library for machine learning and AI by Google, supporting deep learning and other AI tasks.
Usage**:** Used to preprocess images and extract features with the ResNet50 model, including dimensionality reduction via GlobalMaxPooling2D**.**
NumPy
Purpose**:** A library for efficient numerical operations and handling arrays.
Usage**:** Used to manipulate image arrays, normalize data, and store embeddings.
scikit-learn
Purpose**:** A machine learning library for classification, regression, clustering, and more.
Usage: Used for image recommendation via the NearestNeighbors algorithm, finding similar items based on embeddings.
OpenCV
Purpose**:** A library for real-time computer vision tasks**.**
Usage**:** Used for image preprocessing like resizing and color space conversion.
Pillow
Purpose**:** A Python imaging library for opening, manipulating, and saving image files.
Usage**:** Used to load, resize, and handle image formats uploaded by users.
tqdm
Purpose**:** A library for creating progress bars in Python loops.
Usage: Used to display progress during image feature extraction.

The provided project consists of two main components: **feature extraction and embedding generation** (`main.py`) and **Streamlit-based user interface for recommendations** (`app.py`). Below is a detailed breakdown of each file and its functionality.

main.py (Feature Extraction and Embedding Generation)

Purpose:

- Precomputes deep features (embeddings) for all images in the images directory using a pre-trained **ResNet50** model.
- Saves these embeddings and the corresponding filenames for efficient retrieval during recommendation.

Steps:

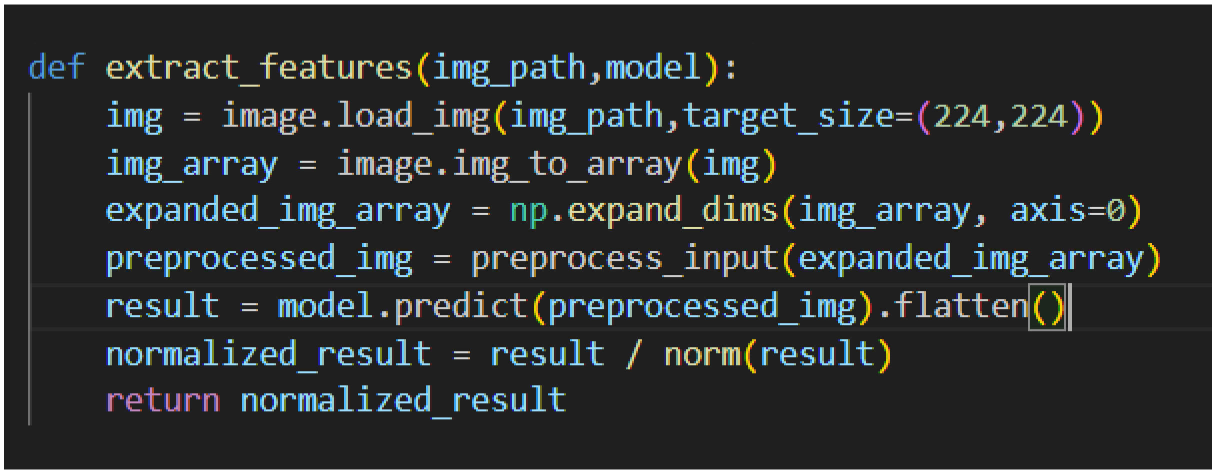
1. **Initialize the ResNet50 Model**:
   - The ResNet50 model is loaded with pre-trained weights from ImageNet, excluding the fully connected layers (include_top=False).
   - **GlobalMaxPooling2D** is added to reduce feature dimensions while retaining

```
model =
ResNet50(weights='imagenet',include_top=False,input_shape=(224,224,3))
model.trainable = False

model = tensorflow.keras.Sequential([
    model,
    GlobalMaxPooling2D()
])
```

2. Feature Extraction Function:

Reads an image, preprocesses it for ResNet50, and extracts normalized features using the model.



```
def extract_features(img_path,model):
    img = image.load_img(img_path,target_size=(224,224))
    img_array = image.img_to_array(img)
    expanded_img_array = np.expand_dims(img_array, axis=0)
    preprocessed_img = preprocess_input(expanded_img_array)
    result = model.predict(preprocessed_img).flatten()
    normalized_result = result / norm(result)
    return normalized_result
```

3. **Generate Features for All Images**:

- Loops through all images in the images directory, extracting features and storing them in a list (feature_list).
- Filenames are also saved in a list for later use.

```
for file in tqdm(filenames):
    feature_list.append(extract_features(file, model))
```

4. **Save Features and Filenames**:

- Both the feature list and filenames are serialized using pickle for efficient reuse in the recommendation system.

```
pickle.dump(feature_list, open('embeddings.pkl', 'wb'))
pickle.dump(filenames, open('filenames.pkl', 'wb'))
```

app.py (Streamlit User Interface)

Purpose:

- Provides a web-based interface for users to upload images and receive fashion recommendations.
- Uses the embeddings generated in main.py to find similar items based on uploaded images.

Key Components:

1. **Load Precomputed Data**:
    - The precomputed embeddings (embeddings.pkl) and filenames (filenames.pkl) are loaded into memory.

```
feature_list = np.array(pickle.load(open('embeddings.pkl', 'rb')))
filenames = pickle.load(open('filenames.pkl', 'rb'))
```

2. **Sidebar Navigation**:

- A collapsible sidebar with various fashion categories for enhanced user experience.

```
with st.sidebar.expander("Men's Fashion"):
    page = st.radio("Select Page", ["T-shirts", "Shirts", "Jeans"])
```

Image Upload:

- Allows users to upload an image file (jpg, png, or jpeg).
- The uploaded file is saved temporarily in the uploads directory for processing

```
uploaded_file = st.file_uploader("Upload an image to get fashion recommendations"
```

Feature Extraction and Recommendation:

- Extracts features from the uploaded image using the same ResNet50 pipeline.
- Finds the nearest neighbors in the precomputed feature space using the **NearestNeighbors** algorithm.

```
features = feature_extraction(file_path, model)
indices = recommend(features, feature_list)
```

Display Recommendations:

- Displays the top recommended images in a grid-like format.

```
for idx in indices[0]:
    recommended_image = Image.open(filenames[idx])
    st.image(recommended_image, caption=f"Recommended Item {idx + 1}", use_column_widt
```

.

Header Image:

Displays a static header image (image.png) at the top for branding or design consistency.

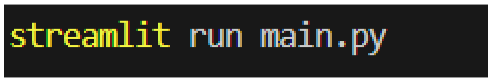
End-to-End Workflow

Precompute Features (app.py):

- Extract and save features for all images in the dataset.

Run Streamlit App (main.py):

- Start the app locally or deploy it online using:



```
streamlit run main.py
```

User Interaction:

- Users upload an image, and the system extracts its features.
- The system finds and displays visually similar items from the dataset.

5. Results:

MAIN PAGE OF THE USER INTERFACE :

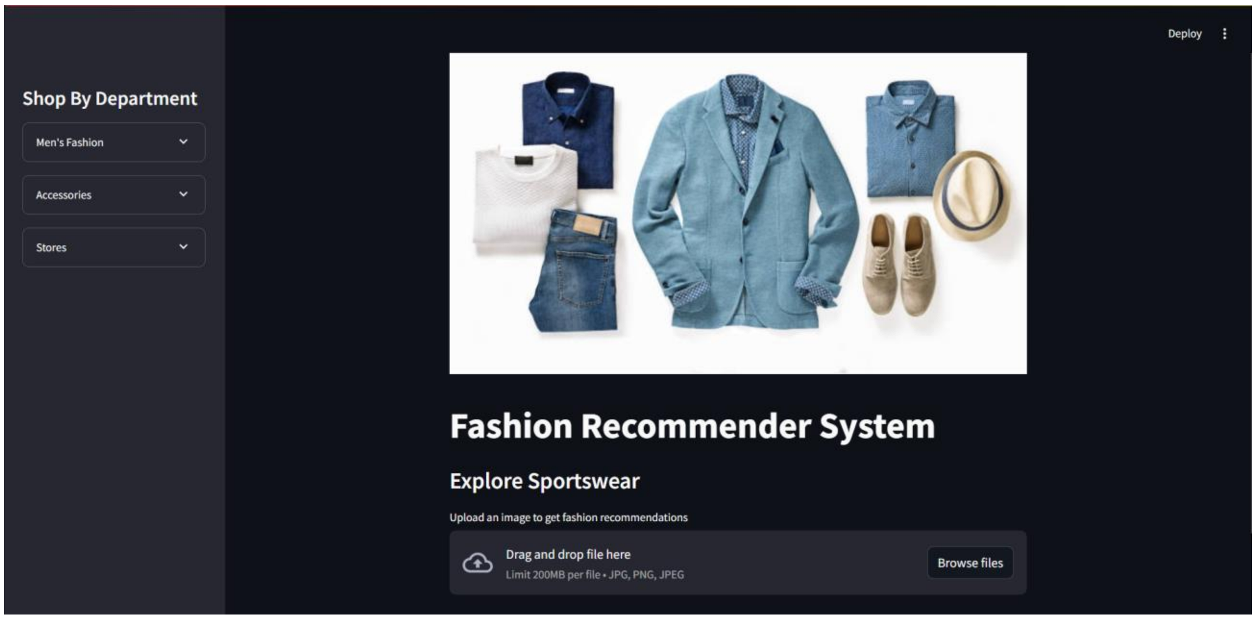

Deploy
Shop By Department
Men's Fashion
Accessories
Stores
Fashion Recommender System
Explore Sportswear
Upload an image to get fashion recommendations
Drag and drop file here
Limit 200MB per file • JPG, PNG, JPEG
Browse files

Figure 3 :- An Output Screenshot of Graph implementation of a Computer network

DRAG AND DROP OPTION FOR UPLOADING THE IMAGE:

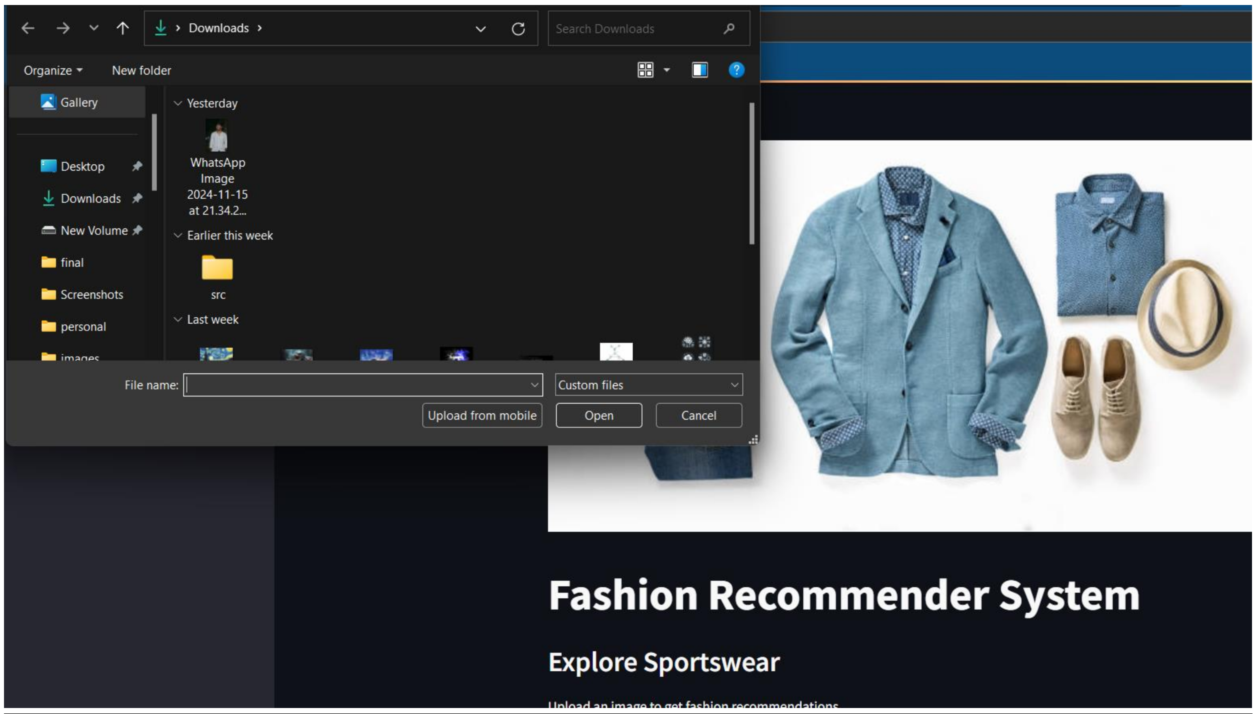


Figure 4 :- An Output Screenshot of the MST of the Network graph resulting in least weight

<u>NAVIGATION BAR FOR DIFFERENT SECTIONS:</u>

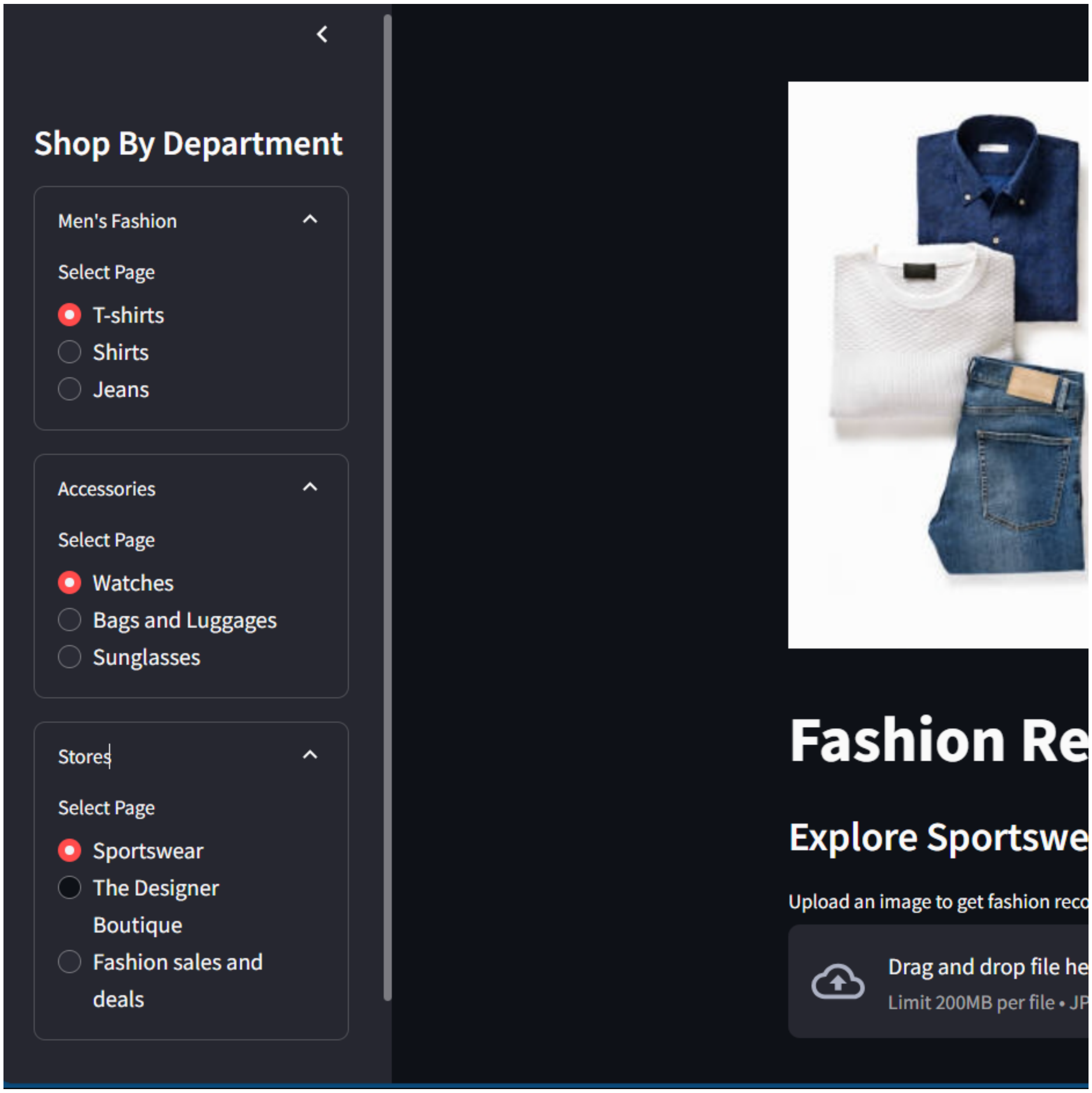
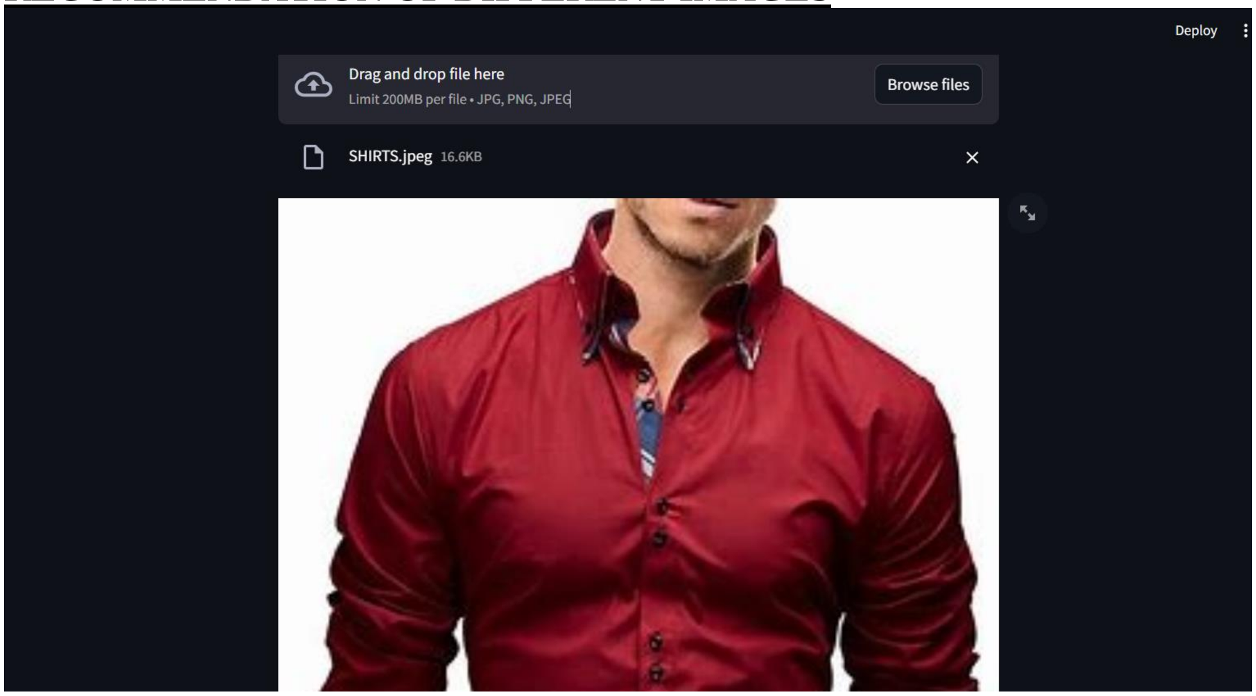
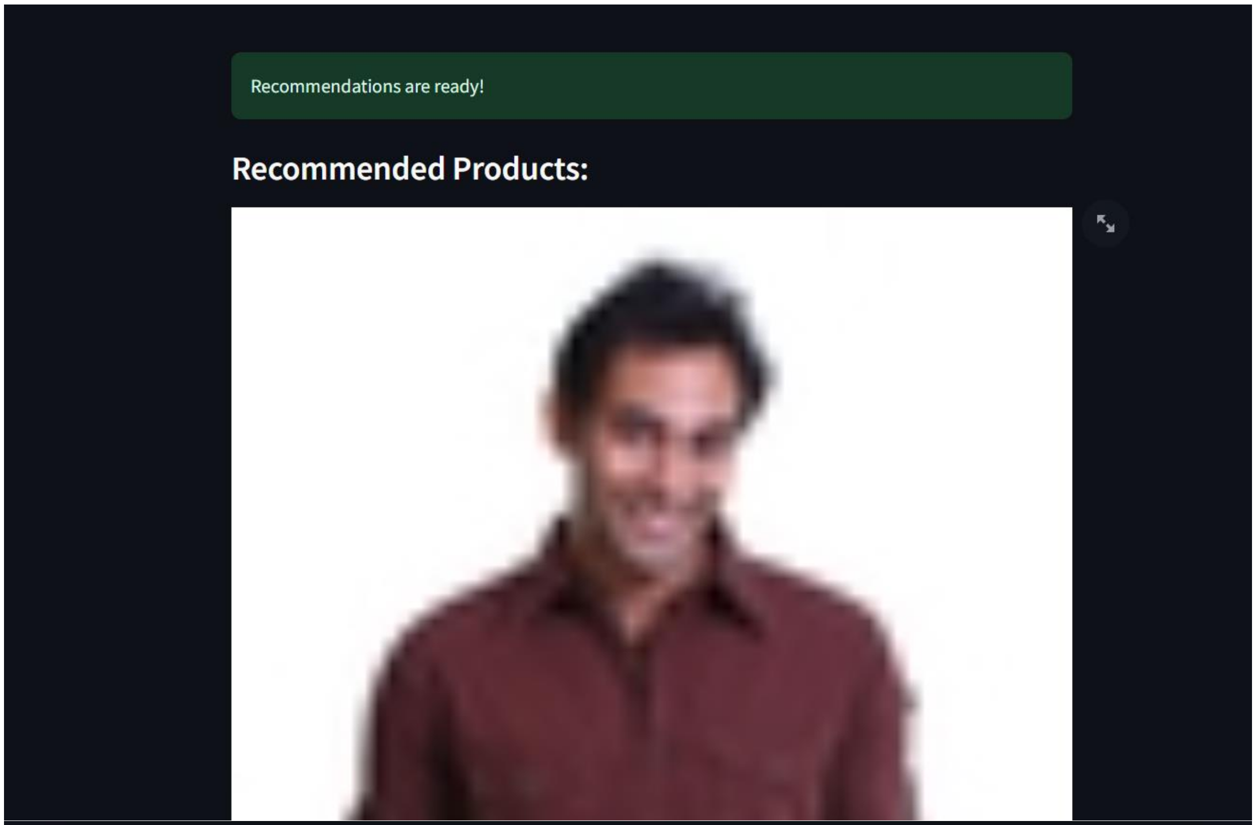


Figure 5 :- An Output Screenshot of the Load Balancing Output of the Network graph along with the Flow Distribution and Load Balancing Visualization

Figure 6 :- An Output Screenshot of the Network Traffic Analysis with Bandwidth and Utilization Details

RECOMMENDATION OF DIFFERENT IMAGES




Recommendations are ready!

Recommended Products:

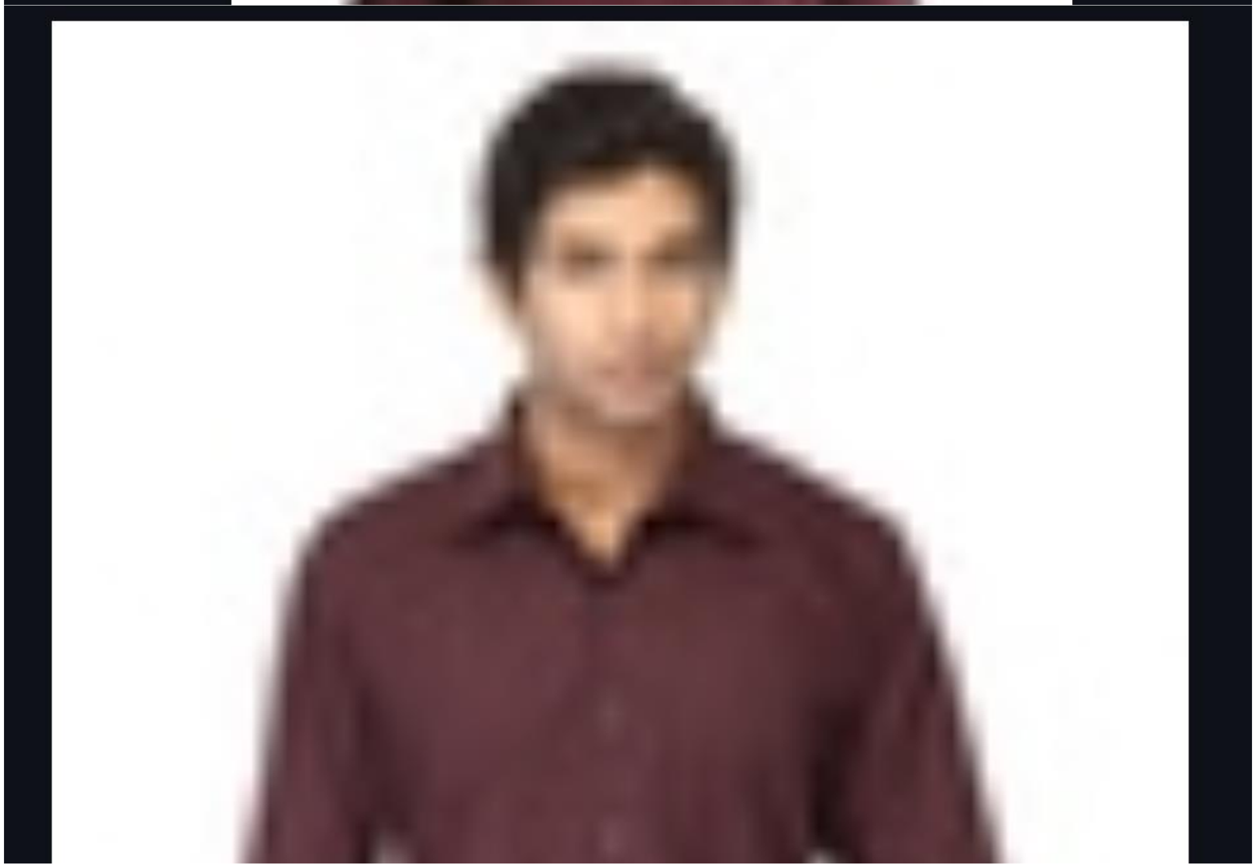


Figure 7 :- An Output Screenshot of the Primary and Redundant Paths Visualization

5. Inferences

The Fashion Recommendation System leverages a deep learning-based approach to suggest similar fashion items based on an input image. By utilizing the ResNet50 pre-trained model, the system extracts deep features from fashion images and uses these features to recommend similar items using the Nearest Neighbors algorithm. The system works by first extracting the features of an uploaded image, comparing them to precomputed embeddings of a catalog of fashion items, and then returning the closest matches based on Euclidean distance.

The model's ability to generalize to different fashion categories and its accuracy in producing relevant recommendations depends on the quality and variety of images in the dataset. The system also utilizes a user-friendly interface built with Streamlit, which allows users to easily interact with the model, upload images, and view the recommended items.

6. Conclusion

This Fashion Recommendation System demonstrates the power of combining deep learning with classical machine learning techniques to create a functional and intuitive product recommendation engine. By using a pre-trained convolutional neural network (ResNet50) for feature extraction and Nearest Neighbors for recommendation, the system provides accurate and relevant suggestions based on visual similarity.

The user interface in Streamlit enhances the user experience, making it easy for non-technical users to upload their images and receive recommendations. While the system performs well with the current dataset, further improvements can be made by incorporating more diverse fashion categories, improving the feature extraction pipeline, and adding more advanced recommendation algorithms. Additionally, user feedback can help refine the model and enhance its performance in real-world scenarios.

Overall, the project successfully demonstrates how AI can be applied to fashion e-commerce to assist consumers in discovering products that match their tastes and preferences.

7. References

1. He, K., et al. (2016). *Deep Residual Learning for Image Recognition*. CVPR. Link
2. Cover, T. M., & Hart, P. E. (1967). *Nearest Neighbor Pattern Classification*. IEEE Trans. Information Theory. Link
3. Mnih, A., et al. (2007). *Recommender Systems with Matrix Factorization Techniques*. KDD.
4. Streamlit Documentation. (n.d.). *Streamlit: The fastest way to build data apps in Python*. Link
5. Xiao, T., et al. (2016). *Fashion-MNIST: A Dataset for Fashion Item Classification*. Link
6. Abadi, M., et al. (2016). *TensorFlow: A system for large-scale machine learning*. OSDI.

8. Source Code :

App.py -

```
import tensorflow

from tensorflow.keras.preprocessing import image

from tensorflow.keras.layers import GlobalMaxPooling2D

from tensorflow.keras.applications.resnet50 import ResNet50,preprocess_input

import numpy as np

from numpy.linalg import norm

import os

from tqdm import tqdm

import pickle


model = ResNet50(weights='imagenet',include_top=False,input_shape=(224,224,3))

model.trainable = False


model = tensorflow.keras.Sequential([

    model,

    GlobalMaxPooling2D()

])


#print(model.summary())


def extract_features(img_path,model):

    img = image.load_img(img_path,target_size=(224,224))

    img_array = image.img_to_array(img)
    expanded_img_array = np.expand_dims(img_array, axis=0)

    preprocessed_img = preprocess_input(expanded_img_array)

    result = model.predict(preprocessed_img).flatten()

    normalized_result = result / norm(result)

    return normalized_result


filenames = []


for file in os.listdir('images'):

    filenames.append(os.path.join('images',file))


feature_list = []


for file in tqdm(filenames):

    feature_list.append(extract_features(file,model))


pickle.dump(feature_list,open('embeddings.pkl','wb'))

pickle.dump(filenames,open('filenames.pkl','wb'))
```

main.py -

```
import streamlit as st
import os
from PIL import Image
```

```python
import numpy as np
import pickle
import tensorflow
from tensorflow.keras.preprocessing import image
from tensorflow.keras.layers import GlobalMaxPooling2D
from tensorflow.keras.applications.resnet50 import ResNet50, preprocess_input
from sklearn.neighbors import NearestNeighbors
from numpy.linalg import norm

# Load pre-computed features and filenames
feature_list = np.array(pickle.load(open('embeddings.pkl', 'rb')))
filenames = pickle.load(open('filenames.pkl', 'rb'))

# Load the ResNet50 model
model = ResNet50(weights='imagenet', include_top=False, input_shape=(224, 224,
3))
model.trainable = False
model = tensorflow.keras.Sequential([
    model,
    GlobalMaxPooling2D()
])

# Save uploaded file
def save_uploaded_file(uploaded_file):
    try:
        upload_dir = 'uploads'
        if not os.path.exists(upload_dir):
            os.makedirs(upload_dir)
        with open(os.path.join(upload_dir, uploaded_file.name), 'wb') as f:
            f.write(uploaded_file.getbuffer())
        return os.path.join(upload_dir, uploaded_file.name)
    except:
        return None

# Extract features from an image
def feature_extraction(img_path, model):
    img = image.load_img(img_path, target_size=(224, 224))
    img_array = image.img_to_array(img)
    expanded_img_array = np.expand_dims(img_array, axis=0)
    preprocessed_img = preprocess_input(expanded_img_array)
    result = model.predict(preprocessed_img).flatten()
    normalized_result = result / norm(result)
    return normalized_result

# Recommend similar images
```

```
def recommend(features, feature_list):
    n_neighbors = min(6, len(feature_list))
    if len(feature_list) < 1:
        raise ValueError("Feature list is empty. Ensure embeddings.pkl is
properly generated.")

    neighbors = NearestNeighbors(n_neighbors=n_neighbors, algorithm='brute',
metric='euclidean')
    neighbors.fit(feature_list)
    distances, indices = neighbors.kneighbors([features])
    return indices

# Sidebar for navigation with collapsible sections
st.sidebar.title("Shop By Department")

# Sections in the sidebar
with st.sidebar.expander("Men's Fashion"):
    page = st.radio("Select Page", ["T-shirts", "Shirts", "Jeans"])

with st.sidebar.expander("Accessories"):
    page = st.radio("Select Page", ["Watches", "Bags and Luggages",
"Sunglasses"])

with st.sidebar.expander("Stores"):
    page = st.radio("Select Page", ["Sportswear", "The Designer Boutique",
"Fashion sales and deals"])

# Display the header image
header_image_path = 'image.png'  # Use the path of your uploaded image
header_image = Image.open(header_image_path)
st.image(header_image, use_column_width=True)

# Main workflow
st.title('Fashion Recommender System')
st.subheader(f"Explore {page}")

# Image upload component
uploaded_file = st.file_uploader("Upload an image to get fashion
recommendations", type=['jpg', 'png', 'jpeg'])
if uploaded_file is not None:
    file_path = save_uploaded_file(uploaded_file)
    if file_path:
        display_image = Image.open(file_path)
        st.image(display_image, caption='Uploaded Image', use_column_width=True)
        with st.spinner("Extracting features and finding recommendations..."):
            features = feature_extraction(file_path, model)
            indices = recommend(features, feature_list)

        st.success("Recommendations are ready!")
        st.subheader("Recommended Products:")

        # Display recommended images with larger size
        for idx in indices[0]:
            recommended_image = Image.open(filenames[idx])
            st.image(recommended_image, caption=f"Recommended Item {idx + 1}",
use_column_width=True)
    else:
        st.error("An error occurred while uploading the file. Please try again.")
```