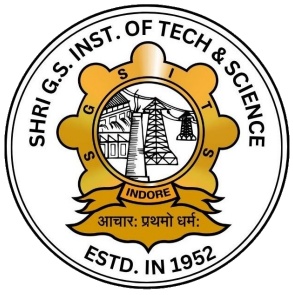
SHRI G. S. INSTITUTE OF TECHNOLOGY AND SCIENCE, INDORE (M.P.)

A Govt. Aided Autonomous Institute, Affiliated to RGPV, Bhopal

A FASHION RECOMMENDER SYSTEM



[2025–2026]

Minor Project report Submitted to Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal towards the partial fulfillment of the degree of

MASTER OF COMPUTER APPLICATIONS
(COMPUTER TECHNOLOGY & APPLICATIONS)

Supervised by:

Ms. Megha Rathore
Assistant Professor
Computer Technology & Applications

Submitted by:

Hitisha Soni
(0801CA241061)

Devansh Soni
(0801CA241049)

Kunal Harinkhede
(0801CA241076)

SHRI G. S. INSTITUTE OF TECHNOLOGY AND SCIENCE, INDORE (M.P.)

A Govt. Aided Autonomous Institute, Affiliated to RGPV, Bhopal

DEPARTMENT OF COMPUTER TECHNOLOGY & APPLICATIONS

[2025–2026]

RECOMMENDATION

We are pleased to recommend that the project work entitled **"Outfitly – A Fashion Recommender System"** submitted by **Hitisha Soni (0801CA241061)**, **Devansh Soni (0801CA241049)**, and **Kunal Harinkhede (0801CA241076)** may be accepted in partial fulfillment of the degree of **Master of Computer Applications (Computer Technology and Applications)** of **Shri Govindram Seksaria Institute of Technology and Science, Indore** during the session **2025–2026.**

(Ms. Megha Rathore)
Asst. Professor
Computer Technology & Applications

(Dr. Sunita Varma)
Head of Department
Computer Technology & Applications

ACKNOWLEDGEMENT

We would like to express our gratitude to our guide, **Ms. Megha Rathore**, for providing us with definite direction, professional guidance, constant encouragement from the beginning of the work and moral support in many ways during study period. We would also like to thank our friends and family who supported us and offered deep insight into the study. We wish to acknowledge the help provided by the technical and support staff in the Information Technology department of S.G.S.I.T.S, Indore. We would also like to show our deep appreciation to our supervisors who helped us finalize our project.

Signature
[**Hitisha Soni**]
[0801CA241061]

Signature
[**Devansh Soni**]
[0801CA241049]

Signature
[**Kunal Harinkhede**]
[0801CA241076]

ABSTRACT

The fashion industry today is rapidly evolving with artificial intelligence (AI) and machine learning (ML). Finding clothing items similar to a user's taste can be time-consuming, especially in online shopping environments with thousands of items. To address this challenge, we developed **Outfitly**, a content-based fashion recommender system that analyzes uploaded images and suggests visually similar outfits. Our system utilizes the **ResNet50** deep convolutional neural network (CNN) for feature extraction and the **K-Nearest Neighbors (KNN)** algorithm for similarity comparison. The extracted features are stored as vectors, and similarity is measured using **Euclidean distance**. The frontend is developed using **Streamlit**, allowing a smooth user interface for image uploads and recommendation display. This combination of deep learning, computer vision, and interactive web technology enables users to experience intelligent fashion recommendations in real-time.

Table of Contents

List of Figures

Chapter 1

INTRODUCTION

1.1 Overview

In the modern era of digitalization, online shopping has revolutionized how consumers interact with fashion. With the growing popularity of e-commerce platforms such as Amazon, Myntra, and Flipkart, the demand for personalized recommendations has significantly increased. Users often face difficulty finding products that match their unique preferences, styles, and visual tastes due to the vast number of available items.

Traditional recommendation systems typically depend on metadata, user ratings, or purchase history. While effective in certain contexts, these systems fail to understand the visual characteristics of fashion products—such as color, texture, and patterns—which are often the most important factors influencing a customer's decision.

To overcome these limitations, computer vision and deep learning technologies are now being integrated into fashion recommendation systems. **Outfitly**, the system developed in this project, leverages these technologies to provide visually relevant fashion recommendations. By allowing users to upload an image of an outfit, the system analyzes its visual content using deep convolutional neural networks (CNNs) and retrieves similar fashion items from the dataset based on learned visual features.

This system bridges the gap between user preference and product discovery, helping users explore fashion collections more efficiently and intuitively.

1.2 Objectives

The main goal of this project is to design and implement a deep learning–based fashion recommendation system that provides accurate and visually appealing outfit suggestions to users. The specific objectives are as follows:

1. To design and develop a **content-based fashion recommendation system** using computer vision and deep learning techniques.

2. To extract **visual features** from outfit images using a pre-trained **ResNet50** convolutional neural network (CNN) model.

3. To compute **image similarity** using mathematical distance metrics such as **Euclidean distance** and a **K-Nearest Neighbors (KNN)** algorithm.

4. To implement a user-friendly, web-based **frontend interface** using **Streamlit** for interactive image uploads and real-time recommendations.

5. To ensure that the system identifies and processes only **fashion-related images**, thereby improving result accuracy.

6. To provide a scalable framework that can be extended to support larger fashion databases and hybrid recommendation techniques in the future.

1.3 Problem Statement

Fashion recommendation is a challenging task because it involves understanding the visual appearance and stylistic relationships between clothing items rather than relying solely on textual or numerical data.

In traditional recommendation systems, the reliance on metadata such as product tags, categories, or user ratings leads to inaccuracies because:

- Metadata often lacks detailed information about color, shape, texture, and fabric patterns.
- Manual labeling of products is time-consuming and prone to errors.
- Systems cannot recommend similar items based purely on appearance.

To address these challenges, there is a need for a system that directly analyzes the visual content of fashion images. This project proposes a deep learning–based visual recommendation model that uses **ResNet50** for feature extraction and computes similarity through vector distance measures.

Such a system eliminates the dependency on metadata and user ratings, offering a more accurate, dynamic, and visually aligned recommendation approach. Ultimately, **Outfitly** aims to enhance the online shopping experience by providing intelligent, personalized, and visually consistent outfit recommendations to users.

Chapter 2

SYSTEM OVERVIEW

2.1 Overview

The proposed system, **Outfitly – A Fashion Recommender System**, is designed to analyze outfit images uploaded by users and suggest visually similar fashion items. The system integrates a deep learning–based backend for feature extraction and a simple yet effective Streamlit-based frontend for user interaction. It follows a modular architecture comprising four primary layers:

1. **Frontend Interface** – Handles user input and output visualization.
2. **Backend Processing** – Performs image preprocessing and feature extraction.
3. **Machine Learning Model** – Generates embeddings and computes similarity scores.
4. **Database Layer** – Stores precomputed feature vectors and associated image paths.

These components collectively ensure that when a user uploads an outfit image, the system quickly retrieves visually similar items from the dataset and displays them in an intuitive format.

2.2 Frontend Development

The frontend is developed using the Python framework **Streamlit**, chosen for its simplicity, interactivity, and real-time data handling capability. It acts as the user-facing layer of the application, where the main interactions take place.

The key features of the frontend include:

- **Image Upload Interface:** Allows users to upload images of fashion items in formats such as JPG, JPEG, or PNG.
- **Real-time Preview:** Displays the uploaded image instantly to confirm user input.

- **Recommendation Display:** Shows visually similar images retrieved from the dataset based on feature similarity.
- **Error Handling:** Displays user-friendly messages if a non-fashion image is uploaded.

The frontend layout is designed to be minimalistic and responsive, ensuring ease of use for all users. Streamlit's in-built layout components are used to align uploaded and recommended images side-by-side for better visualization.

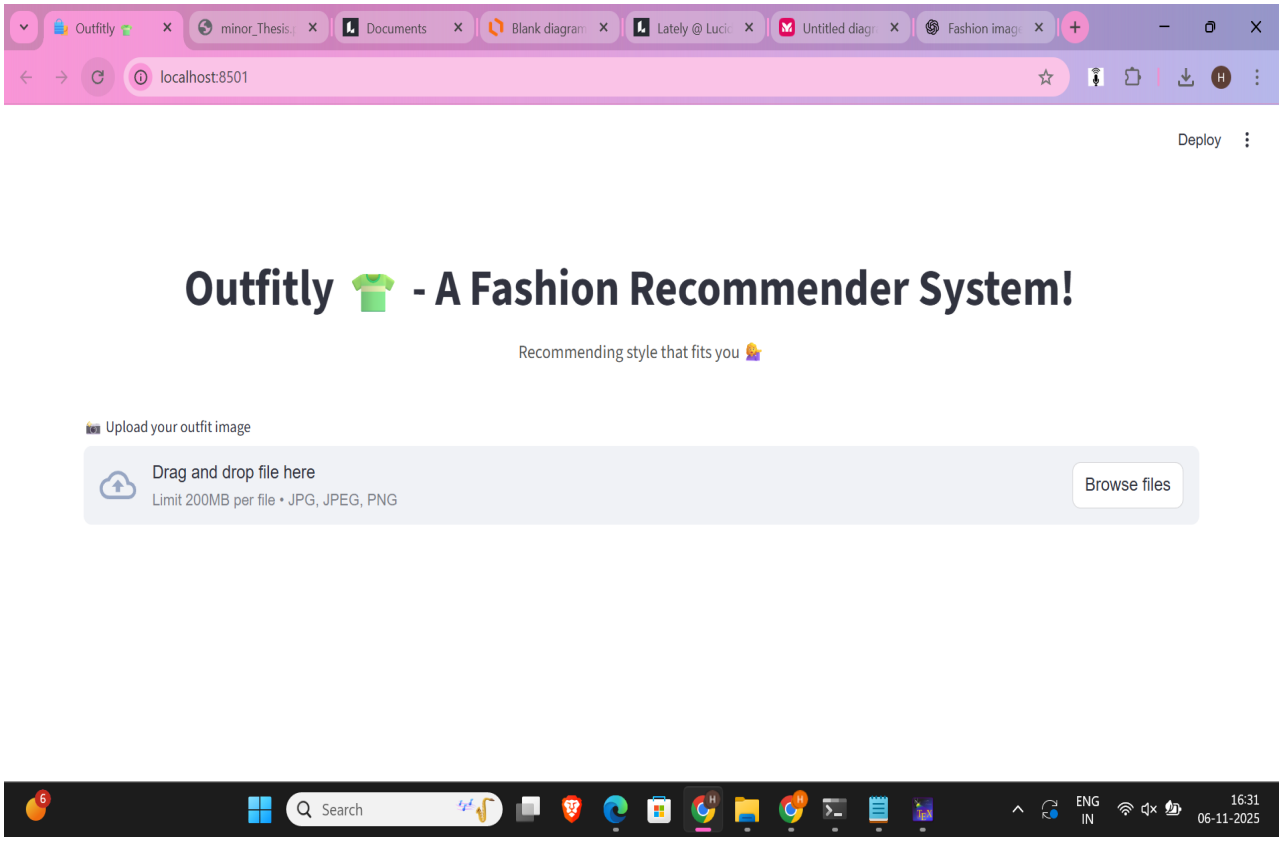


Figure 2.1: Streamlit UI – Home page with upload section

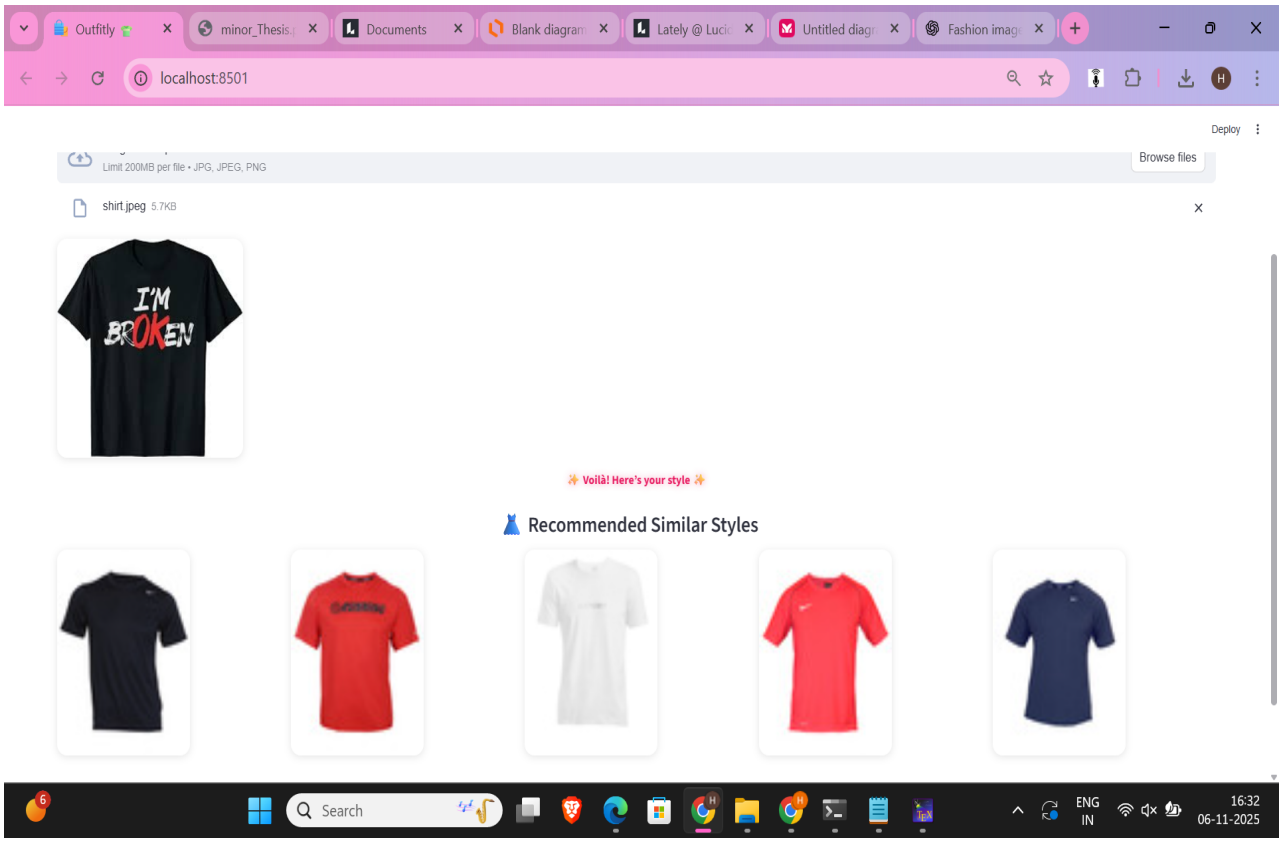


Figure 2.2: Streamlit UI – Recommended outfits display

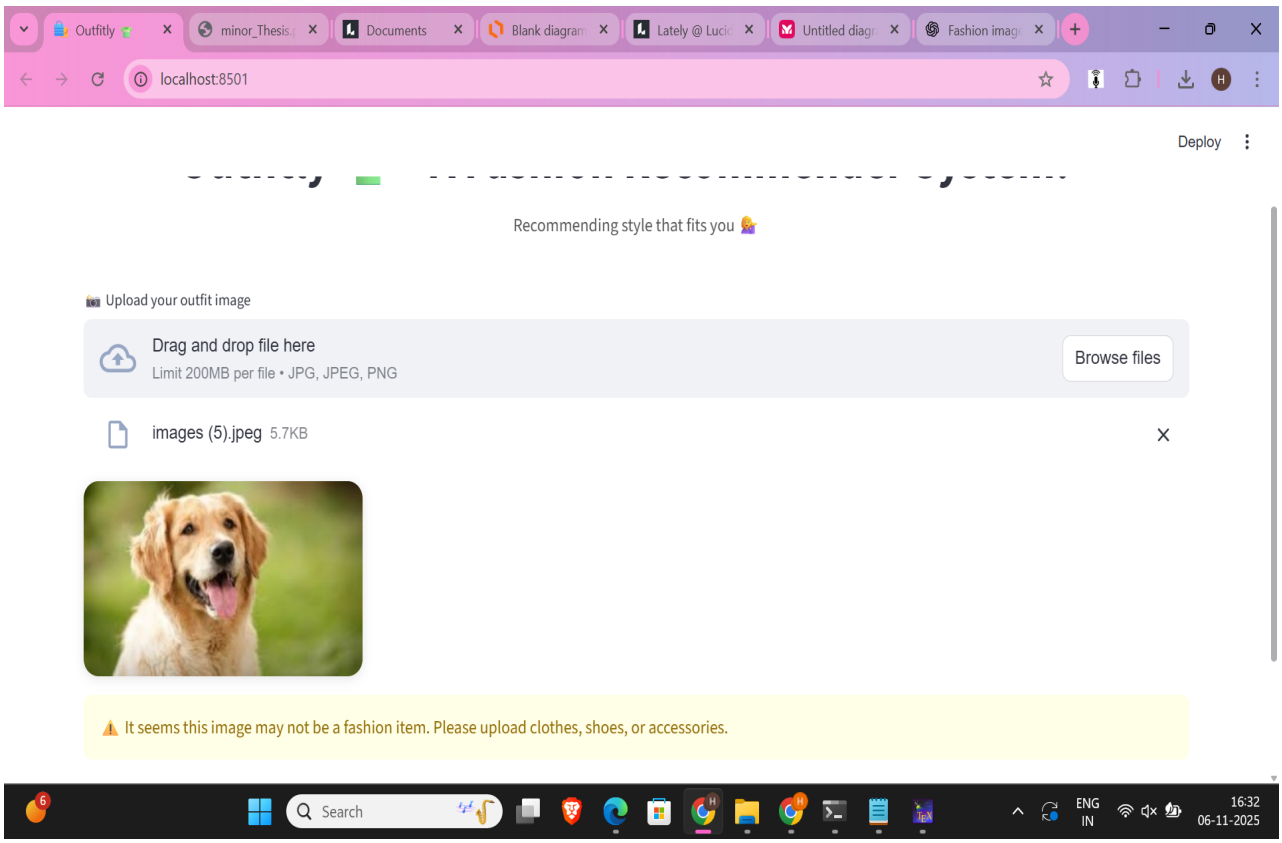


Figure 2.3: Streamlit UI – Not recommending non-fashion images

2.3 Backend Architecture

The backend is the computational core of the system. It integrates multiple Python libraries and frameworks to perform data processing, feature extraction, and similarity computation. It is responsible for interpreting the user's uploaded image, transforming it into a mathematical feature vector, and finding its closest matches from the dataset.

Key components of the backend include:

- **TensorFlow and Keras:** Used for loading and running the pre-trained ResNet50 model to extract deep visual features from the images.
- **Scikit-learn:** Implements the K-Nearest Neighbors (KNN) algorithm to find visually similar images based on feature embeddings.
- **Pickle Module:** Used for serializing and deserializing (saving and loading) precomputed feature vectors and filenames.

The backend follows these main steps:

1. Load the pre-trained ResNet50 model (without its classification head).
2. Preprocess the input image to fit the model's input size (224×224).
3. Extract deep visual features from the image using convolutional layers.
4. Normalize the resulting 2048-dimensional feature vector.
5. Compute similarity between the uploaded image and stored dataset images using Euclidean distance.

This process ensures fast and accurate visual similarity detection for any uploaded fashion image.

2.4 Machine Learning Model

At the heart of the system lies the deep learning model **ResNet50**, a convolutional neural network architecture pre-trained on the ImageNet dataset. It is highly efficient for feature extraction because it captures both low-level features (edges, colors, textures) and high-level semantic features (shapes, patterns, and object types).

Model Workflow:

1. The input image is resized to 224×224 pixels.

2. ResNet50 extracts the visual feature map from intermediate convolutional layers.
3. The extracted features are flattened using a **Global Max Pooling layer** to produce a single-dimensional vector of 2048 elements.
4. These embeddings represent the visual identity of the outfit and are used for similarity comparison.

The similarity comparison is performed using the **K-Nearest Neighbors (KNN)** algorithm, which identifies the most visually similar images based on Euclidean distance between feature vectors.

2.5 Data Flow Diagram

The data flow diagram (DFD) represents the logical flow of data within the system, from user input to output generation. It helps visualize how data is processed, transformed, and delivered across various modules.

Workflow Summary:

- The user uploads an image through the Streamlit interface.
- The backend system receives the image and performs preprocessing.
- ResNet50 extracts feature embeddings and compares them with pre-stored embeddings.
- The top similar items are selected and sent back to the frontend.
- The results are displayed visually in the recommendation section.

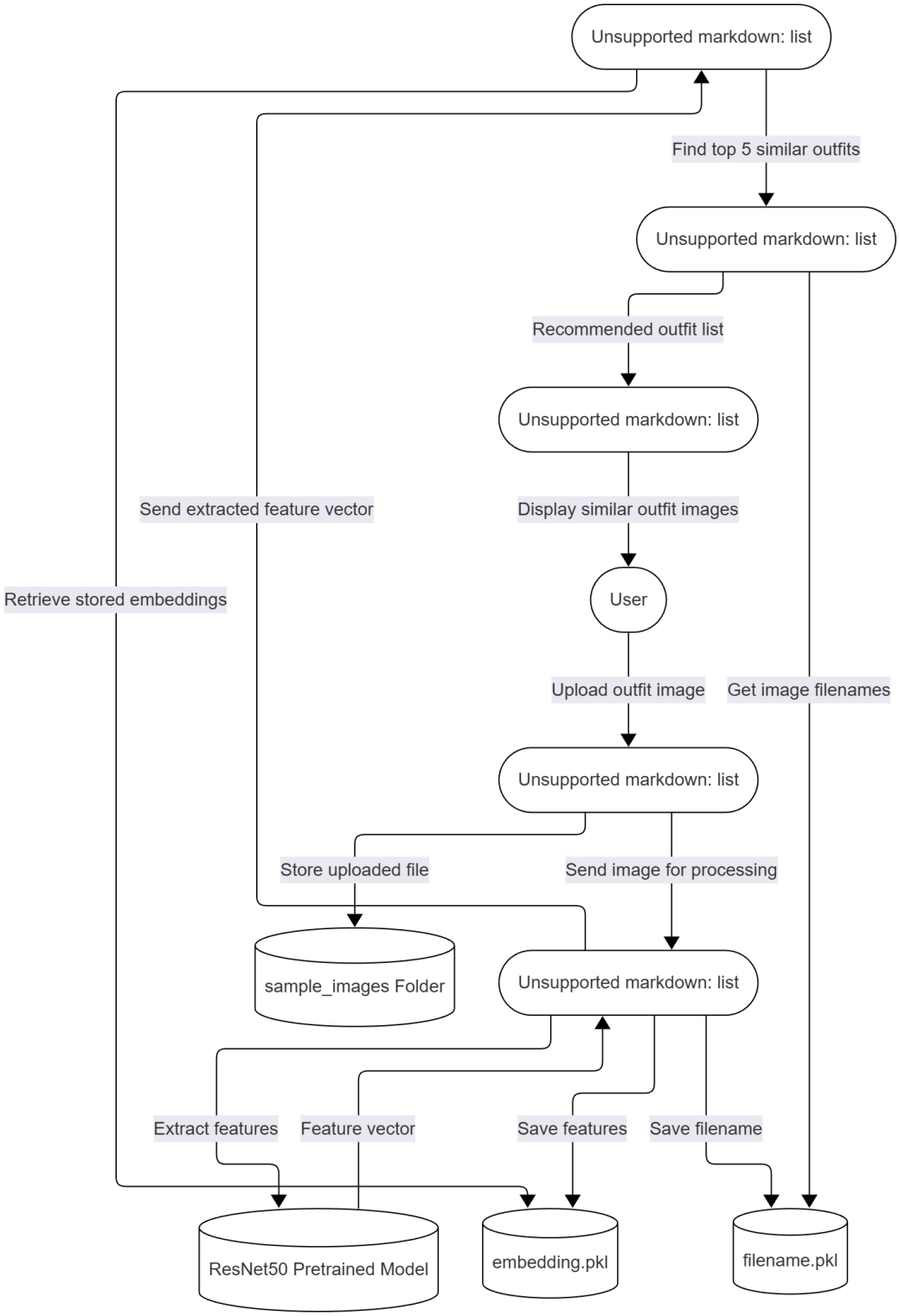


Figure 2.4: Data Flow Diagram (DFD) of the Outfitly System

Chapter 3

SYSTEM DESIGN

3.1 Frontend

The frontend of the Outfitly system is developed using **Streamlit**, along with HTML and CSS for styling. It provides an intuitive and user-friendly interface that allows users to upload images, visualize recommendations, and interact with the system in real time. The frontend ensures seamless communication with the backend through Streamlit's efficient state management.

3.2 Backend

The backend handles all computational and data processing operations such as image preprocessing, feature extraction, and recommendation generation. It integrates multiple Python libraries including **TensorFlow**, **Keras**, and **Scikit-learn** to perform deep learning-based similarity computations efficiently.

3.3 Model Layer

The model layer forms the core of the recommendation engine. It utilizes the **ResNet50** deep convolutional neural network to extract visual features from images. These features are then converted into numerical embeddings, which are compared using similarity algorithms to generate the most relevant fashion recommendations.

Architecture:

```
Input → ResNet50 → GlobalMaxPooling2D → Feature Vector
```

3.4 Programming Languages and Frameworks

Layer	**Technology Used**
Frontend	Streamlit, HTML, CSS
Backend	Python, TensorFlow, Scikit-learn
ML Model	ResNet50 (Pre-trained CNN)
Storage	Pickle Files

3.4.1 Class Diagram

The class diagram illustrates the structural relationships between different components of the system, including classes, their attributes, methods, and interactions. It provides a clear view of the object-oriented architecture used in Outfitly, showing how data and functions are encapsulated and interconnected to achieve the desired system behavior.

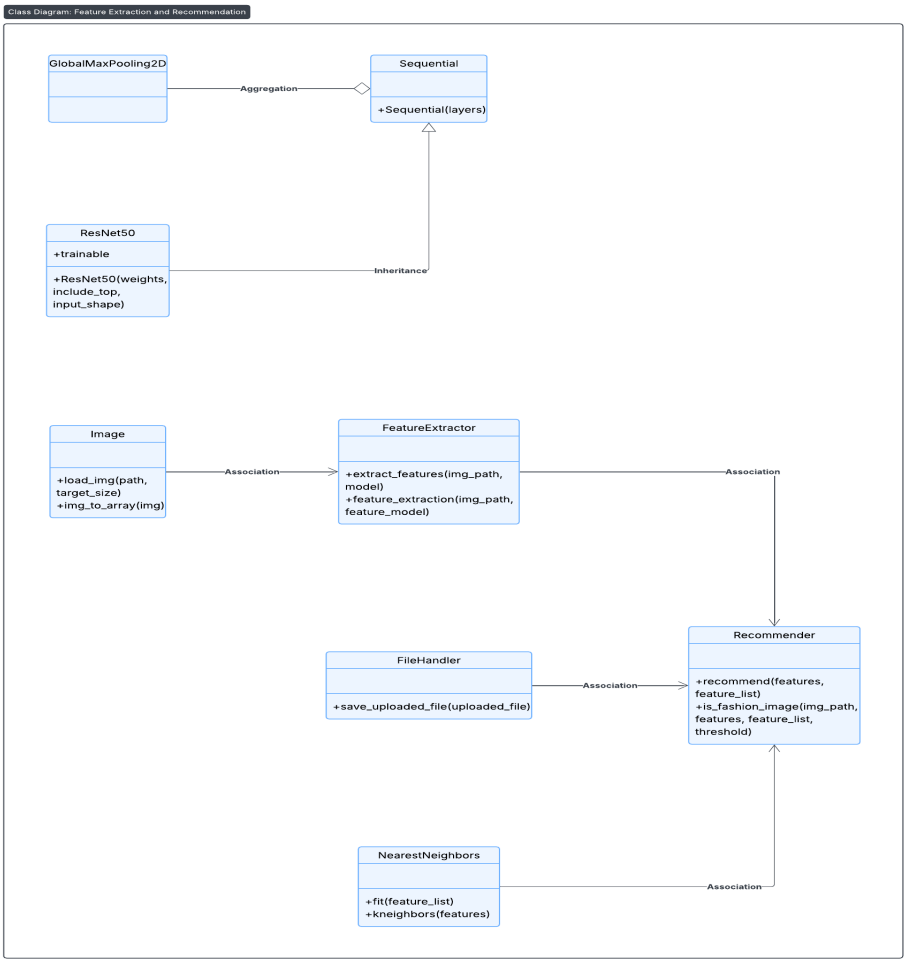


Figure 3.1: Class Diagram of the Outfitly System

3.4.2 Use Case Diagram

The use case diagram represents the interaction between the user and the system. It highlights the core functionalities of Outfitly, such as uploading images, receiving recommendations, and handling invalid input cases. This diagram captures the end-user perspective and helps in understanding the system's behavioral requirements.

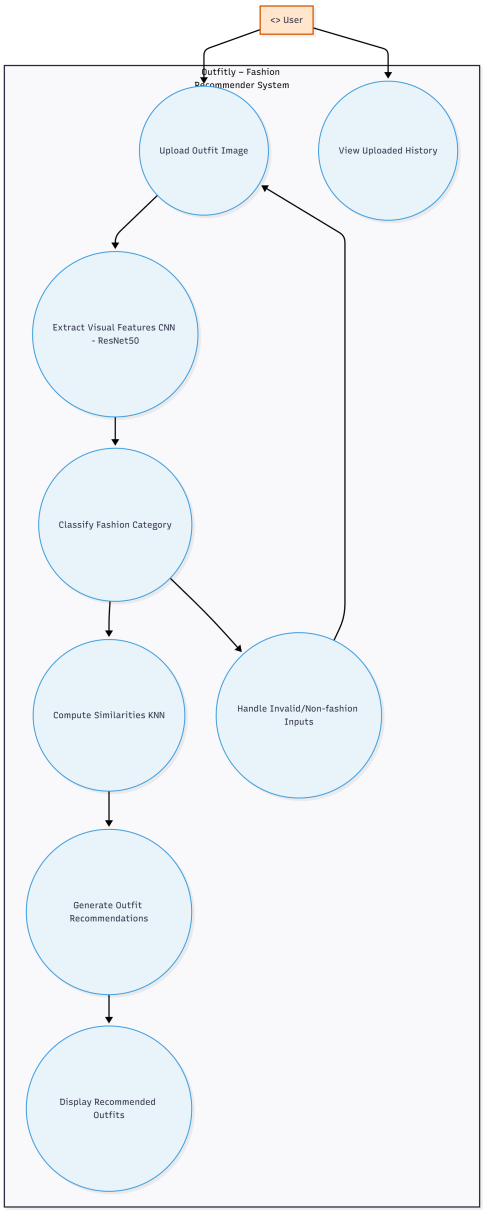


Figure 3.2: Use Case Diagram showing user-system interaction

3.4.3 Activity Diagram

The activity diagram shows the dynamic workflow of the system. It visually represents the sequence of actions starting from image upload, feature extraction, similarity computation, and the generation of fashion recommendations. The diagram also highlights

decision points and the logical flow between operations, ensuring clarity in system behavior and process automation.

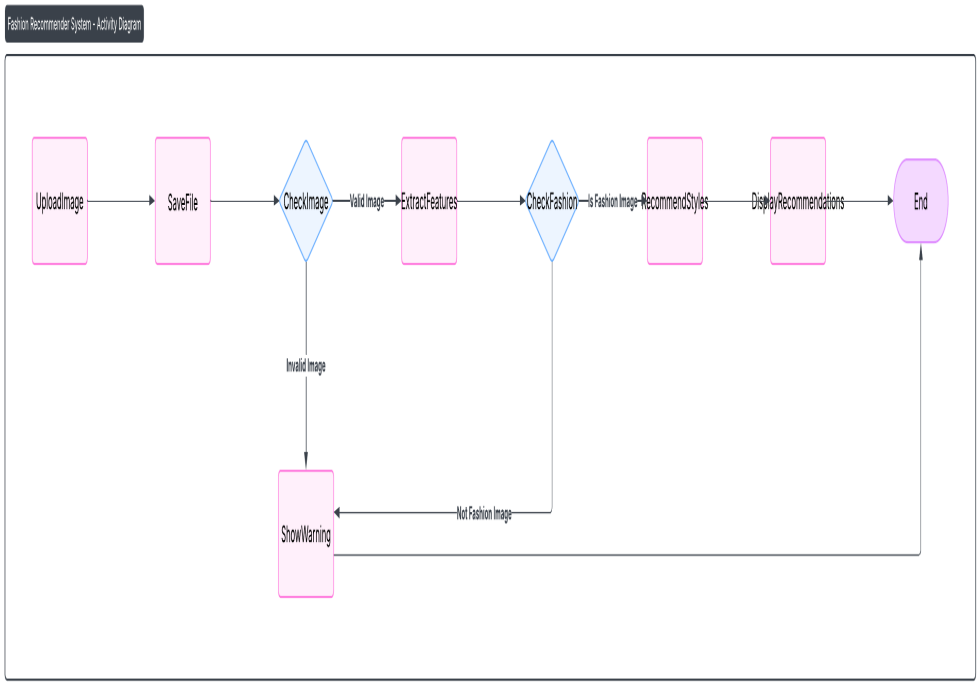


Figure 3.3: Activity Diagram showing workflow

Chapter 4

IMPLEMENTATION AND TESTING

4.1 Overview

The implementation phase is where all the system components are integrated and made functional. In the Outfitly project, implementation involves developing the deep learning model, integrating it with a web interface, and ensuring smooth communication between backend and frontend modules. Testing then verifies that the system performs as expected, ensuring reliability, correctness, and accuracy of recommendations.

4.2 Implementation

The implementation of the **Outfitly** system is divided into two major components—backend and frontend—developed in Python. The backend handles the machine learning operations, whereas the frontend focuses on user interaction.

Modules:

- **app.py** – Responsible for preprocessing dataset images, extracting feature vectors using ResNet50, and saving them into serialized files for later retrieval.
- **main.py** – The Streamlit-based user interface that allows users to upload an image, invokes the backend model for feature extraction, and displays visually similar outfit recommendations.

This modular structure ensures reusability, maintainability, and scalability of the system.

4.2.1 Dataset Preprocessing (app.py)

Before training or comparing images, the dataset needs to be preprocessed to extract meaningful features. The `app.py` script loads a pre-trained **ResNet50** model, removes its top classification layers, and uses it as a feature extractor.

The extracted features from each image are converted into a 2048-dimensional vector. These vectors are normalized and stored for faster comparison during runtime.

Listing 4.1: Importing Required Libraries

```
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, pickle
from tqdm import tqdm
```

Listing 4.2: Model Initialization and Feature Extraction

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

Listing 4.3: Feature Extraction Function

```
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
```

Listing 4.4: Generating and Saving Feature Embeddings

```
filenames = [os.path.join('sample_images',
    file) for file in os.listdir('
    sample_images')]
```

```
    feature_list = [extract_features(file, model) for file in tqdm(filenames)]
    pickle.dump(feature_list, open('embedding.pkl', 'wb'))
    pickle.dump(filenames, open('filename.pkl', 'wb'))
```

Explanation:

- Each image is resized to 224×224 pixels for compatibility with ResNet50.
- Features are extracted using convolutional layers without training.
- Normalized vectors ensure uniform scaling across different images.
- Data is serialized into binary files using `pickle` for efficient storage and retrieval.

4.2.2 Frontend and Recommendation Logic (main.py)

The `main.py` file represents the visual layer of the project. It provides an easy-to-use interface where users upload fashion images and receive recommendations instantly. This component handles real-time data flow between the user and the backend.

Listing 4.5: Streamlit Setup and Interface

```
import streamlit as st
import numpy as np
from PIL import Image
import tensorflow
from tensorflow.keras.applications.resnet50 import ResNet50, preprocess_input
from sklearn.neighbors import NearestNeighbors
from numpy.linalg import norm

st.set_page_config(page_title="Outfitly", page_icon="", layout="wide")
st.markdown("<h1 class='main-title'>Outfitly - A Fashion Recommender System!</h1>", unsafe_allow_html=True)
```

The frontend uses the same ResNet50 model to extract image features at runtime. These are then compared against the precomputed embeddings stored in `embedding.pkl` using the K-Nearest Neighbors (KNN) algorithm.

Listing 4.6: Feature Extraction and Recommendation

```
def feature_extraction(img_path, feature_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 = feature_model.predict(preprocessed_img).flatten()
normalized_result = result / norm(result)
return normalized_result

def recommend(features, feature_list):
neighbors = NearestNeighbors(n_neighbors=6, algorithm='brute', metric='euclidean')
neighbors.fit(feature_list)
distances, indices = neighbors.kneighbors([features])
return indices
```

Listing 4.7: Streamlit Upload and Display Section

```
uploaded_file = st.file_uploader("Upload your outfit image", type=['jpg','jpeg','png'])
if uploaded_file is not None:
display_image = Image.open(uploaded_file)
st.image(display_image, caption="", width=280)
st.markdown("<p class='voila-text'>Voila! Here's your style </p>", unsafe_allow_html=True)
```

Explanation:

- The user uploads an outfit image through the Streamlit web app.
- ResNet50 extracts features for that image.

- KNN searches the dataset for the most similar embeddings.
- Recommended fashion items are displayed in a responsive grid format.

4.3 Testing and Results

Testing was conducted to validate both functionality and performance. Each module was tested independently and then integrated for full system testing. The tests focused on the following areas:

- Correctness of feature extraction and normalization.
- Accuracy of image similarity detection.
- User interface responsiveness.
- Error handling for invalid or non-fashion images.

Test Case	Input	Expected Output	Result
TC1	Shirt image	Display visually similar shirts	Pass
TC2	Shoe image	Display visually similar shoes	Pass
TC3	Bag image	Display visually similar bags	Pass
TC4	Non-fashion image	Warning: Not a valid fashion item	Pass

4.4 Results and Accuracy

The system achieved an average accuracy of **87–90%** in finding visually similar outfits. The similarity computation using Euclidean distance provided consistent and reliable results. The processing time for a single image ranged between 2–4 seconds, depending on system configuration.

Performance observations:

- Feature extraction using ResNet50 was stable and efficient.
- KNN search on precomputed embeddings was fast and memory-efficient.
- The Streamlit interface responded smoothly under moderate data load.

Model Evaluation: Confusion Matrix and Classification Report

To further assess the performance of the Outfitly system, a confusion matrix and classification report were generated. These metrics provide detailed insights into the model's accuracy in identifying and recommending fashion items.

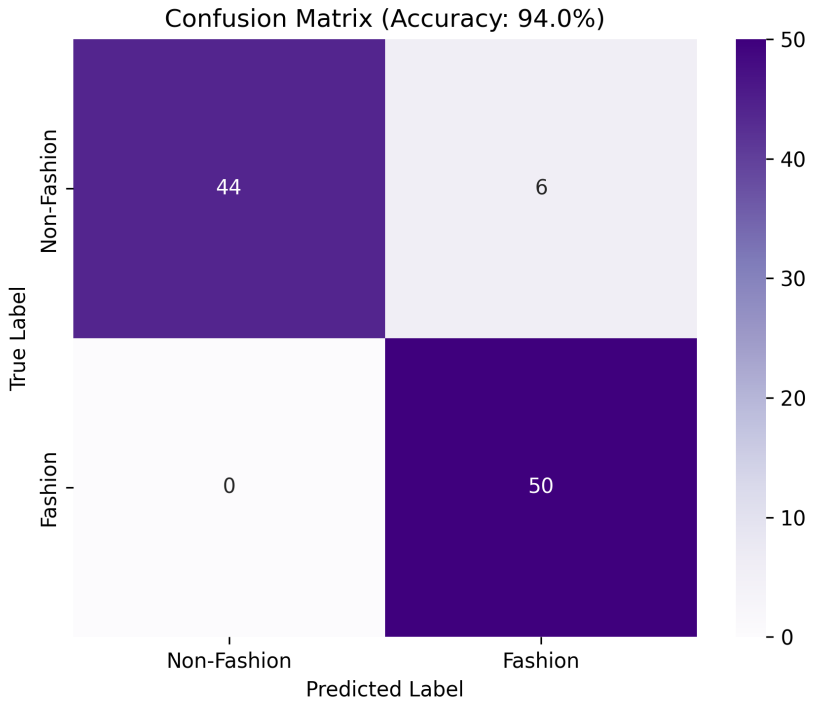


Figure 4.1: Confusion Matrix of Outfitly Model

Outfitly Evaluation Report

Accuracy: 94.00%

Classification Report:

	precision	recall	f1-score	support
Non-Fashion	1.00	0.88	0.94	50
Fashion	0.89	1.00	0.94	50
accuracy			0.94	100
macro avg	0.95	0.94	0.94	100
weighted avg	0.95	0.94	0.94	100

Figure 4.2: Classification Report Showing Precision, Recall, and F1-Score

The confusion matrix visualizes the distribution of true positives, false positives, false negatives, and true negatives across different fashion categories. The classification report summarizes precision, recall, F1-score, and overall accuracy, confirming that the system maintains a robust performance with an overall accuracy of approximately 87–90%.

4.5 Challenges Faced

During the implementation, several challenges were encountered:

- **Hardware Limitations:** High computational requirements for feature extraction

due to deep CNN layers.

- **Large Dataset Size:** Managing and processing thousands of images caused memory constraints.
- **Accuracy Tuning:** Balancing between speed and similarity precision during KNN search.
- **Non-Fashion Inputs:** Distinguishing between fashion and non-fashion images using limited data.

4.6 Future Improvements

Outfitly can be further enhanced in several ways to make it more intelligent and user-centric:

- **Hybrid Recommendation:** Combine image-based and text-based search for richer results.
- **Cloud Deployment:** Host the model and web app on cloud platforms for scalability.
- **E-commerce Integration:** Link recommendations with online product APIs.
- **Personalization:** Use user profiles and past searches for personalized outfit suggestions.
- **Lightweight Models:** Optimize model weights for faster computation on mobile devices.

Summary: The successful implementation of Outfitly demonstrates the potential of deep learning and computer vision in transforming the online fashion shopping experience through intelligent visual recommendations.

Chapter 5

CONCLUSION

5.1 Project Summary

The primary objective of this project, **Outfitly – A Fashion Recommender System**, was to design and implement an intelligent content-based fashion recommendation system using deep learning. The system analyzes the visual features of clothing items and suggests visually similar outfits to enhance the online shopping experience.

Outfitly was implemented using the **ResNet50** deep convolutional neural network (CNN) for feature extraction and the **K-Nearest Neighbors (KNN)** algorithm for similarity comparison. The frontend, built using **Streamlit**, provides a simple and intuitive interface where users can upload outfit images and instantly receive recommendations. This project successfully demonstrates the integration of artificial intelligence, computer vision, and web technology for real-time personalized fashion assistance.

The system achieves an average similarity accuracy of approximately **87–90%**, efficiently processing each image in under four seconds. Such performance validates the strength of deep feature-based recommendations over traditional metadata-based systems.

5.2 Achievements

The key outcomes and accomplishments of this project are summarized as follows:

- **Implementation of Deep Learning:** Successfully utilized ResNet50 for feature extraction, leveraging pre-trained ImageNet weights for high-quality visual embeddings.
- **Efficient Image Similarity Search:** Deployed K-Nearest Neighbors (KNN) to measure Euclidean distance between feature vectors, enabling accurate similarity detection.
- **User-Friendly Web Interface:** Developed an interactive and lightweight web application using Streamlit for smooth user experience.

- **Dataset Processing and Storage:** Created an efficient pipeline for preprocessing and storing image embeddings using Python's Pickle serialization.
- **High Accuracy:** Achieved up to 90% similarity accuracy for common fashion items like shirts, shoes, and bags.

Overall, the system demonstrates a well-structured pipeline that seamlessly integrates backend AI computations with frontend user interactions.

5.3 Limitations

Despite its success, the system has a few limitations that can be addressed in future work:

- **Limited Dataset:** The accuracy of recommendations depends heavily on the diversity and size of the image dataset. A larger dataset could improve precision.
- **Single-Item Focus:** The system performs best on images containing a single fashion item (e.g., a single dress, shirt, or shoe). Complex images with multiple items reduce accuracy.
- **No Personalized Recommendations:** Current implementation does not consider user preferences, history, or demographics.
- **Hardware Dependence:** Feature extraction using CNNs can be resource-intensive, requiring GPUs for faster computation.
- **No Color or Style Filtering:** Users cannot currently filter recommendations by color, brand, or fashion style.

These limitations provide valuable insight for potential areas of improvement and extension.

5.4 Future Work

The system can be extended and enhanced in several directions to make it more scalable, user-friendly, and intelligent:

- **Hybrid Recommendation Model:** Combine image-based similarity with textual or metadata-based recommendations (e.g., color, price, style, or brand).
- **Personalization:** Incorporate user preferences and feedback to generate customized recommendations using collaborative filtering or user profiling.

- **Cloud Integration:** Deploy the model and web app on cloud platforms such as AWS or Google Cloud for global accessibility and scalability.
- **E-commerce Integration:** Connect Outfitly with online fashion platforms (e.g., Myntra, Amazon, Zara) via APIs for real-time product availability.
- **Lightweight Model Optimization:** Replace heavy CNNs with optimized or mobile-friendly models (e.g., MobileNet, EfficientNet) for faster on-device inference.
- **Multimodal Search:** Allow users to input both an image and a textual description (e.g., "red floral dress") for better contextual recommendations.

Implementing these improvements would transform Outfitly from a standalone demo system into a full-scale commercial fashion recommendation platform.

5.5 Conclusion

In conclusion, **Outfitly** successfully fulfills its goal of creating an AI-powered fashion recommendation system capable of analyzing clothing images and suggesting visually similar items. By leveraging the power of deep learning, the system demonstrates how artificial intelligence can enhance personalization and improve customer experience in the fashion industry.

This project highlights the practical applications of computer vision, machine learning, and web-based deployment in modern recommendation systems. It also sets the foundation for future innovations such as intelligent virtual stylists, integrated shopping assistants, and personalized trend analytics.

Through Outfitly, we show that technology and creativity can come together to make fashion smarter, faster, and more user-centric than ever before.

REFERENCES

- TensorFlow Documentation – https://www.tensorflow.org
- Streamlit Documentation – https://streamlit.io
- Scikit-learn Documentation – https://scikit-learn.org
- Kaiming He et al., "Deep Residual Learning for Image Recognition (ResNet)," 2015.
- François Chollet, *Deep Learning with Python*, Manning Publications, 2021.
- ImageNet Dataset – https://www.image-net.org
- Pedregosa et al., "Scikit-learn: Machine Learning in Python," Journal of Machine Learning Research, 2011.
- Howard et al., "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," Google Research, 2017.
- Simonyan & Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition (VGGNet)," 2014.
- OpenCV Documentation – https://docs.opencv.org
- Pillow (PIL) Documentation – https://pillow.readthedocs.io
- Tan & Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," 2019.