
Project Title

Complementary Outfit Recommender Using Convolutional Neural Networks (CNNs)

Table of Contents

1. Introduction
2. Problem Statement
3. Objectives
4. Dataset Description
5. Methodology
 - 5.1 Feature Extraction Using CNNs
 - 5.2 Similarity Matching
 - 5.3 Complementary Outfit Retrieval
6. Implementation
 - 6.1 Tools and Libraries
 - 6.2 Workflow
7. Results
8. Challenges and Solutions
9. Conclusion
10. References

1. Introduction

In the era of fast fashion and e-commerce, finding complementary outfits that match a specific clothing item (e.g., a shirt) is a challenging task. This project aims to simplify this process by building a **Complementary Outfit Recommender System** using **Convolutional Neural Networks (CNNs)**. The system takes a query image (e.g., a shirt) as input and retrieves visually similar items along with their complementary outfits (e.g., pants, shoes, and accessories) from a dataset. By leveraging the power of CNNs, the system provides accurate and efficient fashion recommendations, enhancing the user experience in online shopping and personal styling.

2. Problem Statement

Fashion enthusiasts and online shoppers often struggle to find clothing items that complement their existing wardrobe. Manually searching for matching outfits is time-consuming and often ineffective. This project addresses this problem by automating the process of finding complementary outfits using **deep learning**. Given a query image of a clothing item (e.g., a shirt), the system retrieves:

1. Visually similar items from the dataset.
2. Complementary outfits (e.g., pants, shoes, and accessories) associated with those items.

3. Objectives

To build a system that can extract high-level visual features from fashion images using CNNs.

To retrieve visually similar items for a given query image.

To recommend complementary outfits based on the retrieved items.

To provide an efficient and user-friendly solution for fashion recommendation.

4. Dataset Description

Polyvore (www.polyvore.com) is a popular fashion website, where users create and upload outfit data as shown in These fashion outfits contain rich multimodal information like images and descriptions of fashion items, number of likes of the outfit, hash tags of the outfit, etc.

We constructed own dataset from Ployvore.

The dataset consists of folders, each representing a unique outfit. Each folder contains several parts of a full outfit.



Figure 3: A sample outfit from the Polyvore website. A typical outfit contains a fashion item list, *i.e.*, pairs of fashion images and their corresponding descriptions.

5. Methodology

5.1 Feature Extraction Using CNNs

A **Convolutional Neural Network (CNN)** is used to extract high-level features from the images.

The CNN analyzes patterns, textures, and shapes in the images and converts them into numerical feature vectors.

These feature vectors represent the visual content of the images and are used for similarity matching.

5.2 Similarity Matching

The feature vector of the query image is compared with the feature vectors of all images in the dataset using **cosine similarity**.

The system retrieves the top-k most similar images based on the similarity scores.

5.3 Complementary Outfit Retrieval

For each retrieved item, the system identifies the corresponding folder and retrieves the complementary outfit images.

6. Implementation

6.1 Tools and Libraries

Python: Primary programming language.

TensorFlow/Keras: For building and using the CNN model.

OpenCV: For image preprocessing.

scikit-learn: For similarity computation (cosine similarity).

Matplotlib: For visualizing results.

6.2 Workflow

1. **Data Preparation:**
 - Load and preprocess images (resize, normalize).
2. **Feature Extraction:**
 - Use a pre-trained CNN to extract feature vectors from the images.
3. **Similarity Matching:**
 - Compute cosine similarity between the query image and all images in the dataset.
4. **Result Retrieval:**
 - Retrieve the top-k similar items and their complementary outfits.
5. **Visualization:**
 - Display the query image, similar items, and complementary outfits.

7. Results

The system successfully retrieves visually similar shirts for a given query image.

It also provides corresponding complementary outfits, allowing users to visualize complete looks.

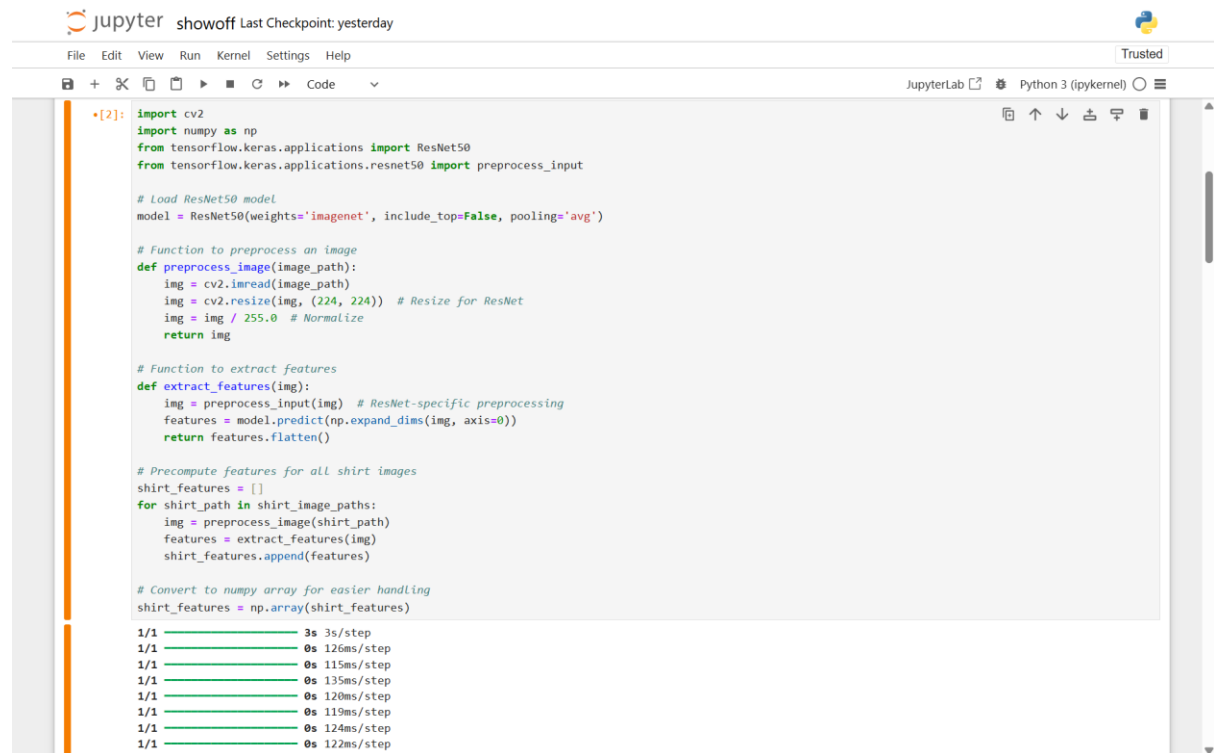
Example:

Query Image: A blue striped shirt.

Retrieved Items: Similar shirts (e.g., different shades of blue striped shirts).

Complementary Outfits: Images of models wearing the retrieved shirts with matching pants, shoes, and accessories.

Current Works:



The image shows a JupyterLab interface with a Python 3 (ipykernel) environment. The code in the cell is as follows:

```
[2]: import cv2
import numpy as np
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.applications.resnet50 import preprocess_input

# Load ResNet50 model
model = ResNet50(weights='imagenet', include_top=False, pooling='avg')

# Function to preprocess an image
def preprocess_image(image_path):
    img = cv2.imread(image_path)
    img = cv2.resize(img, (224, 224)) # Resize for ResNet
    img = img / 255.0 # Normalize
    return img

# Function to extract features
def extract_features(img):
    img = preprocess_input(img) # ResNet-specific preprocessing
    features = model.predict(np.expand_dims(img, axis=0))
    return features.flatten()

# Precompute features for all shirt images
shirt_features = []
for shirt_path in shirt_image_paths:
    img = preprocess_image(shirt_path)
    features = extract_features(img)
    shirt_features.append(features)

# Convert to numpy array for easier handling
shirt_features = np.array(shirt_features)
```

The output of the code is a series of progress bars indicating the execution time for each image path:

```
1/1 ----- 3s 3s/step
1/1 ----- 0s 126ms/step
1/1 ----- 0s 115ms/step
1/1 ----- 0s 135ms/step
1/1 ----- 0s 120ms/step
1/1 ----- 0s 119ms/step
1/1 ----- 0s 124ms/step
1/1 ----- 0s 122ms/step
...
```

```
[3]: from sklearn.metrics.pairwise import cosine_similarity

def find_similar_shirts(query_image_path, shirt_features, shirt_image_paths, model_image_paths, top_k=5):
    # Extract features from query image
    query_img = preprocess_image(query_image_path)
    query_features = extract_features(query_img)

    # Compute similarities
    similarities = cosine_similarity([query_features], shirt_features)[0]

    # Get indices of top matches
    top_indices = np.argsort(similarities)[-top_k:][::-1]

    # Return paths of similar shirts and corresponding model images
    similar_shirts = [shirt_image_paths[idx] for idx in top_indices]
    similar_models = [model_image_paths[idx] for idx in top_indices]

    return similar_shirts, similar_models

# Example usage
query_shirt = "shirt.jpg" # Replace with your query shirt image path
similar_shirts, similar_models = find_similar_shirts(query_shirt, shirt_features, shirt_image_paths, model_image_paths, top_k=5)

# Print the paths of similar shirts and corresponding model images
print("Top similar shirts:")
for shirt in similar_shirts:
    print(shirt)

print("\nCorresponding model images:")
for model in similar_models:
    print(model)
```



8. Challenges and Solutions

Challenge: Limited dataset size.

Solution: Use transfer learning with a pre-trained CNN to leverage learned features from large datasets like ImageNet.

Challenge: Variability in image quality and lighting.

Solution: Apply image preprocessing techniques (e.g., resizing, normalization) to standardize the input.

9. Conclusion

This project demonstrates the effectiveness of **Convolutional Neural Networks (CNNs)** in building a **Complementary Outfit Recommender System**. By extracting high-level visual features and performing similarity matching, the system provides accurate and efficient fashion recommendations. The integration of CNNs ensures that the system can handle complex visual patterns, making it a valuable tool for online shoppers and fashion enthusiasts. Future enhancements, such as fine-tuning

and user interface development, can further improve the system's performance and usability.

10. References

1. (Fashion Outfit Complementary Item Retrieval Yen-Liang Lin, Son Tran, Larry S. Davis)
<https://arxiv.org/abs/1912.08967>
2. (Learning Fashion Compatibility with Bidirectional LSTMs)
<https://arxiv.org/abs/1707.05691>
3. (Image Based Fashion Product Recommendation with Deep Learning Hessel Tuinhof¹, Clemens Pirker², and Markus Haltmeier³)
<https://arxiv.org/abs/1805.08694>