FASHION CLOTHING MACHINE LEARNING

PROJECT

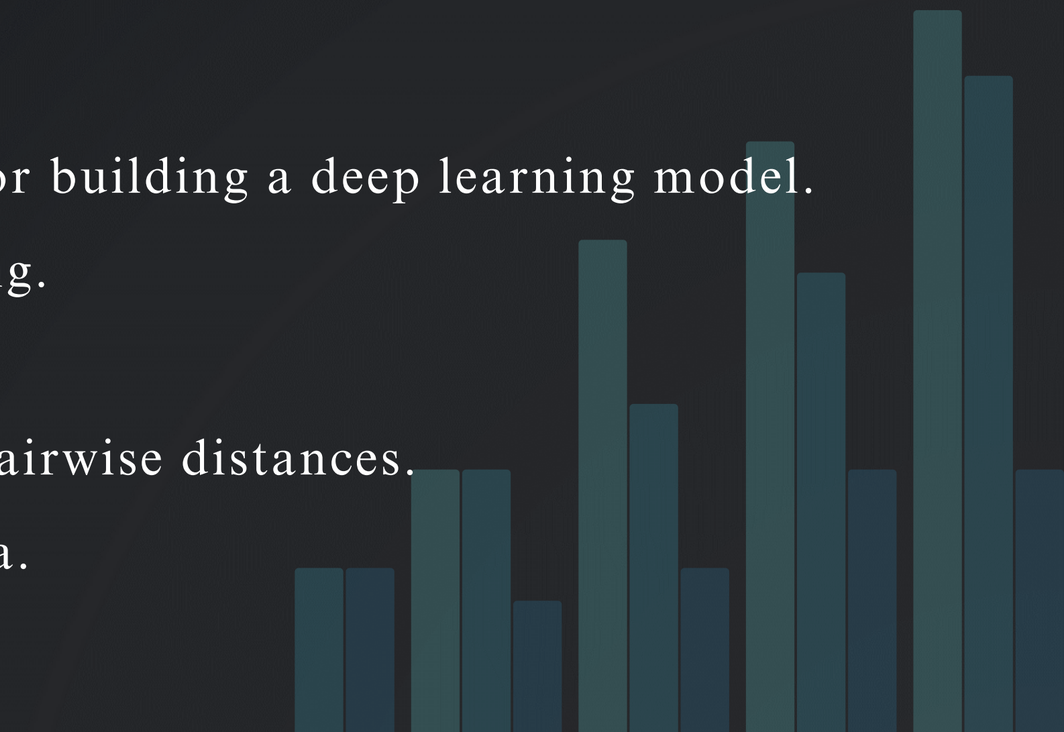
Ata Güneş & Mertcan Duran

CONTENTS

1.1 Importing The Necessary Libraries & Loading The Data

1. Libraries Used:

1. **Data Manipulation and Visualization**
   - NumPy and Pandas: For efficient data manipulation.
   - Matplotlib and Seaborn: Visualization of data and results.
2. **Deep Learning with Keras**
   - ImageDataGenerator: Augmentation of image data.
   - Sequential, Dropout, Flatten, Dense: Components for building a deep learning model.
   - Applications: Pre-trained models for transfer learning.
3. **Pairwise Distances and HTTP Requests**
   - Sklearn.metrics.pairwise_distances: Calculation of pairwise distances.
   - Requests: Making HTTP requests for additional data.



4. Image Processing

- **PIL (Python Imaging Library): Handling image processing tasks.**

5. Serialization and Deserialization

- Pickle: Serialization and deserialization of Python objects.

6. Date and Time Handling

- Datetime: Working with dates and times.

7. Streamlit (Commented Out)

- Streamlit: Library for creating web applications (commented out in this script).

8. Interactive Visualizations with Plotly

- Plotly Figure Factory, Graph Objects, Express: Creating interactive visualizations.

9. Displaying Images

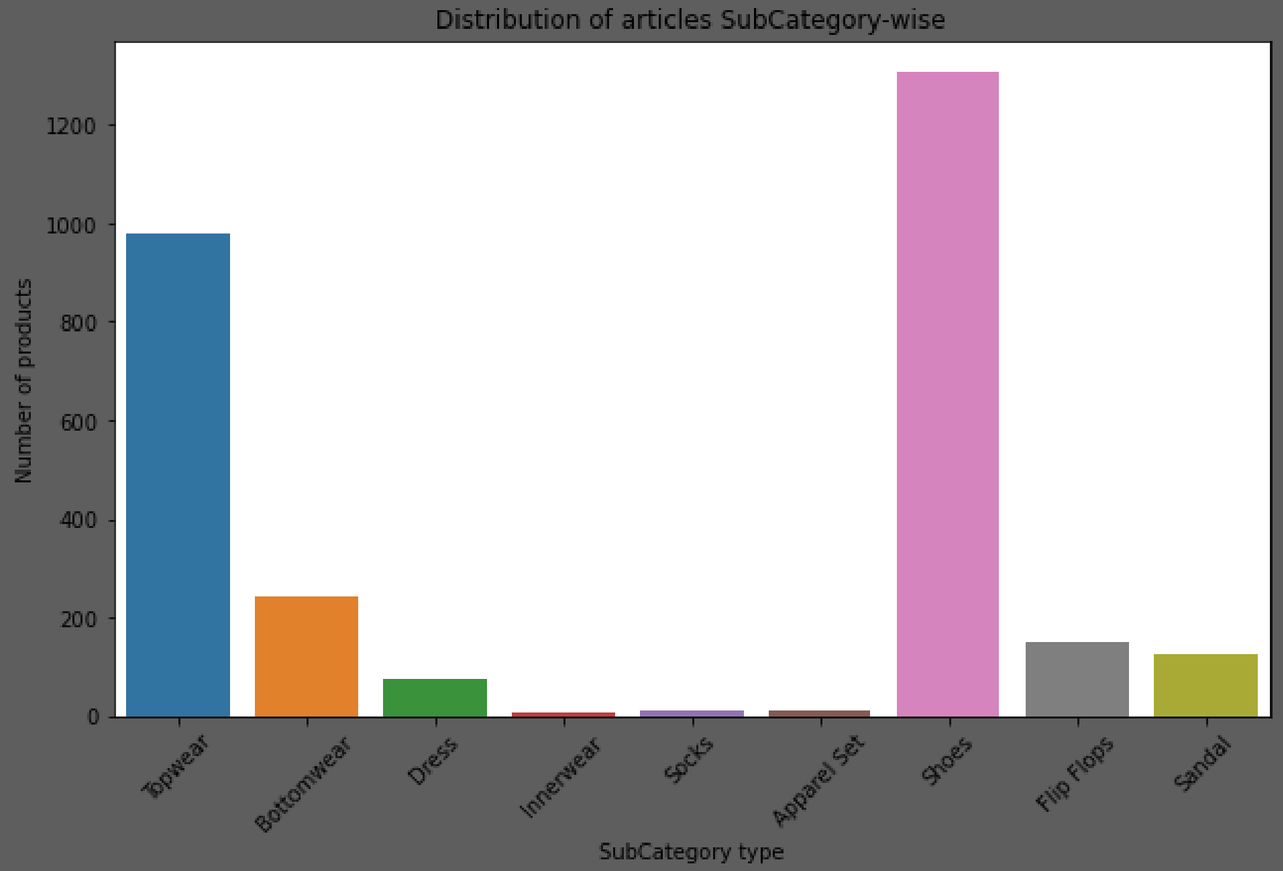
- IPython.display: Displaying images in Jupyter notebooks.

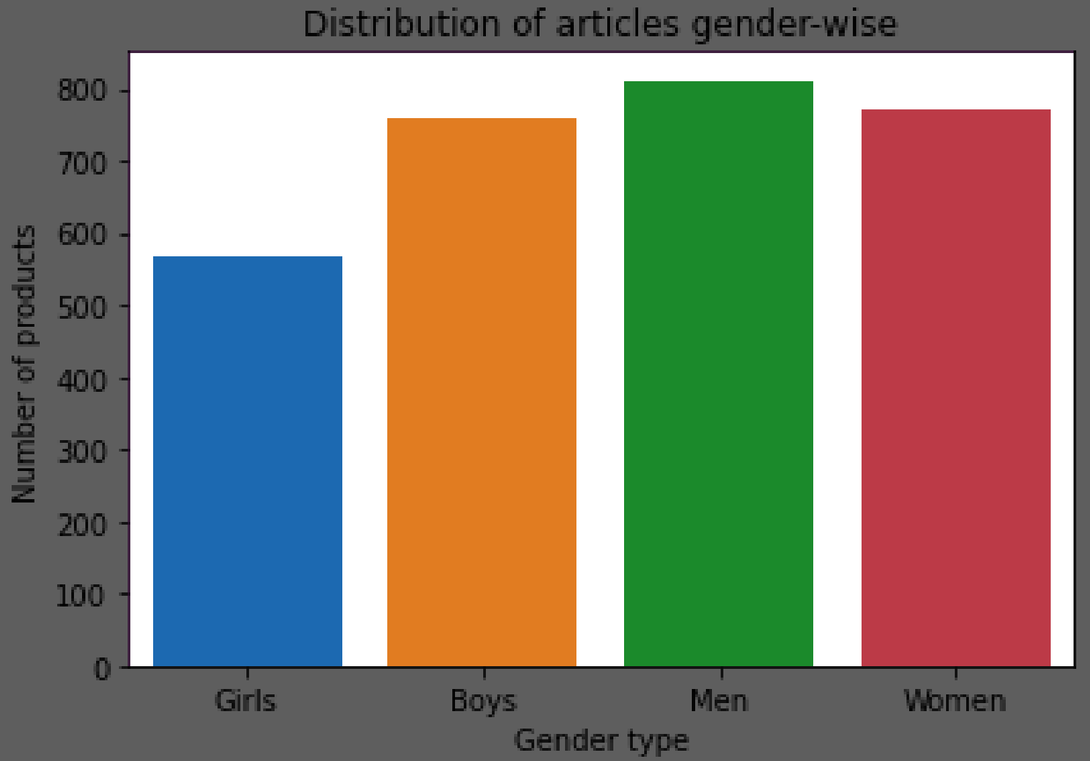
1.2 Basic statistics - Number of products, subcategories & gender

- **Total number of products : 2906**
- **Total number of unique subcategories : 9**
- **Total number of unique gender types : 4**

	ProductId	Gender	Category	SubCategory	ProductType	Colour	Usage	ProductTitle	Image	ImageURL
0	42419	Girls	Apparel	Topwear	Tops	White	Casual	Gini and Jony Girls Knit White Top	42419.jpg	http://assets.myntassets.com/v1/images/style/p...
1	34009	Girls	Apparel	Topwear	Tops	Black	Casual	Gini and Jony Girls Black Top	34009.jpg	http://assets.myntassets.com/v1/images/style/p...
2	40143	Girls	Apparel	Topwear	Tops	Blue	Casual	Gini and Jony Girls Pretty Blossom Blue Top	40143.jpg	http://assets.myntassets.com/v1/images/style/p...
3	23623	Girls	Apparel	Topwear	Tops	Pink	Casual	Doodle Kids Girls Pink I love Shopping Top	23623.jpg	http://assets.myntassets.com/v1/images/style/p...
4	47154	Girls	Apparel	Bottomwear	Capris	Black	Casual	Gini and Jony Girls Black Capris	47154.jpg	http://assets.myntassets.com/v1/images/style/p...
...	...	...	...	...	...	...	...	...	...	...
2901	51755	Women	Footwear	Shoes	Casual Shoes	Black	Casual	Catwalk Women Black Shoes	51755.jpg	http://assets.myntassets.com/v1/images/style/p...
2902	47630	Women	Footwear	Shoes	Flats	Blue	Casual	Carlton London Women Blue Shoes	47630.jpg	http://assets.myntassets.com/v1/images/style/p...
2903	32836	Women	Footwear	Shoes	Flats	Pink	Casual	Grendha Women Flori Pink Sandals	32836.jpg	http://assets.myntassets.com/v1/images/style/p...
2904	35821	Women	Footwear	Shoes	Heels	Black	Casual	Enroute Women Black Heels	35821.jpg	http://assets.myntassets.com/v1/images/style/p...
2905	18553	Women	Footwear	Shoes	Heels	Blue	Casual	Catwalk Women Mary Janes Blue Flats	18553.jpg	http://assets.myntassets.com/v1/images/style/p...

2906 rows × 10 columns


Distribution of articles SubCategory-wise
Number of products
0
200
400
600
800
1000
1200
Topwear
Bottomwear
Dress
Innerwear
Socks
Apparel Set
Shoes
Flip Flops
Sandal
SubCategory type


Distribution of articles gender-wise
Number of products
0
100
200
300
400
500
600
700
800
Girls
Boys
Men
Women
Gender type

2. Data Preparation

1. Boys' Apparel

- The script creates a subset apparel_boys containing fashion items designed for boys. The condition "Gender=="Boys" filters the DataFrame accordingly.

2. Girls' Apparel

- Similarly, a subset apparel_girls is formed, including fashion items labeled for girls using the condition "Gender=="Girls".

3. Men's Footwear

- The subset footwear_men is generated for men's footwear, filtering items with the condition "Gender=="Men".

4. Women's Footwear

- Lastly, footwear_women is created to encompass women's footwear based on the condition "Gender=="Women".

```
apparel_boys = fashion_df[fashion_df["Gender"]=="Boys"]
apparel_girls = fashion_df[fashion_df["Gender"]=="Girls"]
footwear_men = fashion_df[fashion_df["Gender"]=="Men"]
footwear_women = fashion_df[fashion_df["Gender"]=="Women"]
```

3. Feature extraction using ResNet

Objectives

- **Feature Extraction**: Utilize a pre-trained ResNet50 model to extract meaningful features from Men's Footwear images.
- **Data Preprocessing**: Employ an ImageDataGenerator to preprocess and augment the input images.
- **Data Directory**: Set up the directory path for the training data, which contains a diverse range of Men's Footwear images.
- **Save Results**: Save the extracted features and corresponding item codes as NumPy arrays for future use.

---

1. Image Dimensions and Directory

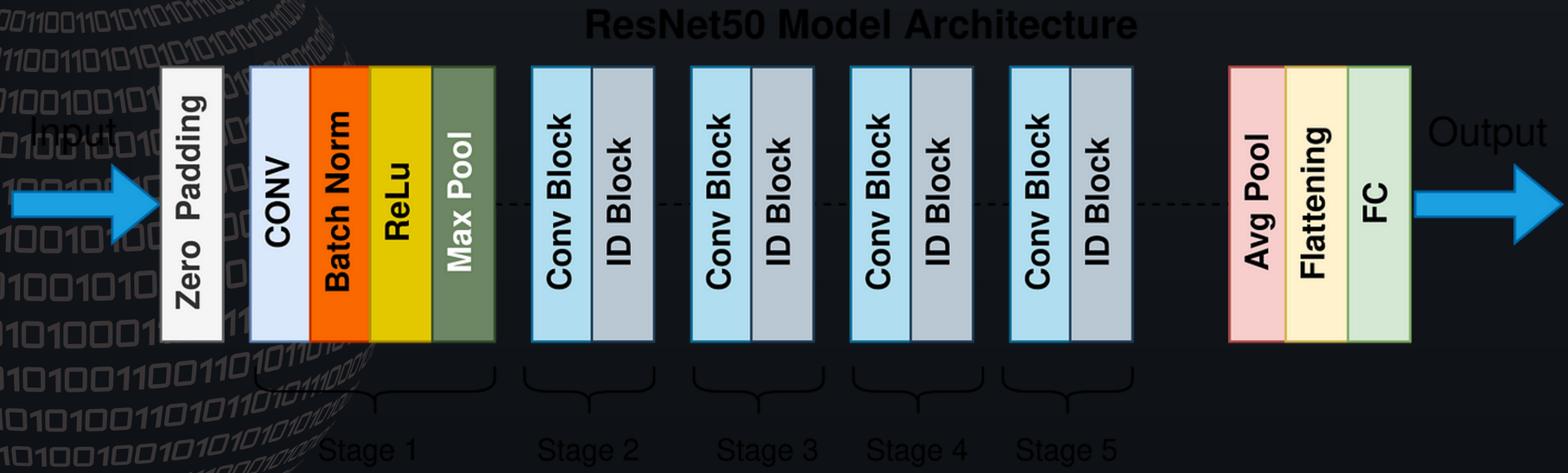
- **Dimensions**: The input images are resized to 224x224 pixels to fit the ResNet50 model's requirements.
- **Directory**: The directory path for the training data is specified.

2. Feature Extraction Function

- **ImageDataGenerator**: Images are preprocessed using an ImageDataGenerator with rescaling.
- **ResNet50 Model**: The pre-trained ResNet50 model is loaded without its top layers for feature extraction.
- **Flow from Directory**: A flow of images is generated from the specified directory for processing.
- **Item Codes Extraction**: Item codes are extracted from file names for later reference.
- **Prediction**: Features are predicted using the ResNet50 model.
- **Reshaping**: The extracted features are reshaped for compatibility.
- **Saving**: Features and item codes are saved as numpy arrays for future use.

RESNET50

- ResNet-50 has been widely used for image classification tasks and is pre-trained on large datasets such as ImageNet. It has 48 convolutional layers, in addition to a global average pooling layer and a fully connected layer for classification. The architecture has been successful in various computer vision tasks and is a popular choice for transfer learning due to its ability to capture hierarchical features in images.



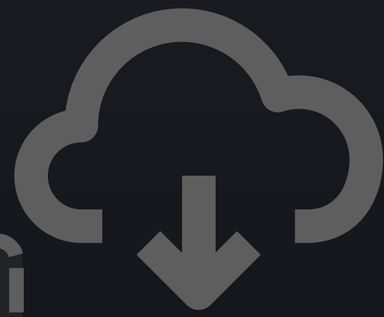
- Loading ResNet50 without the top layers is a common practice in transfer learning. The top layers of a deep neural network, often including fully connected layers, are responsible for high-level tasks like classification. In the case of a pre-trained model like ResNet50 on ImageNet, these top layers have learned to recognize a broad range of objects in the ImageNet dataset.

- The weights='imagenet' argument in the applications.ResNet50 function indicates that the model should be initialized with weights pre-trained on the ImageNet dataset. Here's why this is commonly done:

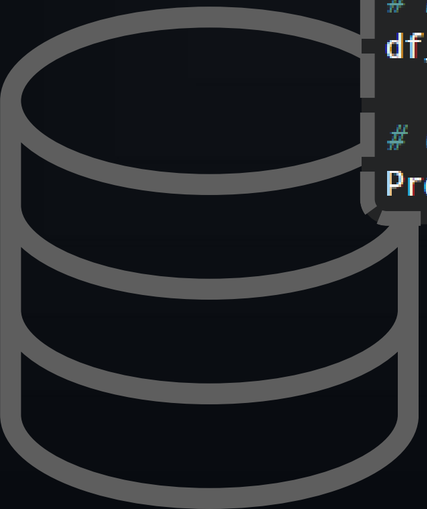
```python
# Setting the dimensions for input images
img_width, img_height = 224, 224

# Directory path for training data
train_data_dir = "/kaggle/input/fashion-images/data/Footwear/Men/Images/"

# Number of training samples
nb_train_samples = 811

# Number of training epochs
epochs = 50

# Batch size for training
batch_size = 1

# Function to extract features using a pre-trained ResNet50 model
def extract_features():
    # ... (see the code for details)

# Recording the start time
a = datetime.now()

# Calling the feature extraction function
extract_features()

# Printing the time taken for feature extraction
print("Time taken in feature extraction", datetime.now()-a)
```

4. Computing the Euclidean distance and recommending similar products

4.1 Loading the extracted features



```
# Loading previously saved extracted features and corresponding product IDs from numpy files
extracted_features = np.load('/kaggle/working/Men_ResNet_features.npy')
Productids = np.load('/kaggle/working/Men_ResNet_feature_product_ids.npy')

# Creating a copy of the original 'footwear_men' DataFrame
men = footwear_men.copy()

# Alternatively, reading a CSV file into the 'men' DataFrame
# men = pd.read_csv('./footwear_men.csv')

# Extracting the 'ProductId' column values from the 'men' DataFrame and converting to a list
df_Productids = list(men['ProductId'])

# Converting the 'Productids' numpy array to a list
Productids = list(Productids)
```



4.2 Distance computation and Recommendation

Overview

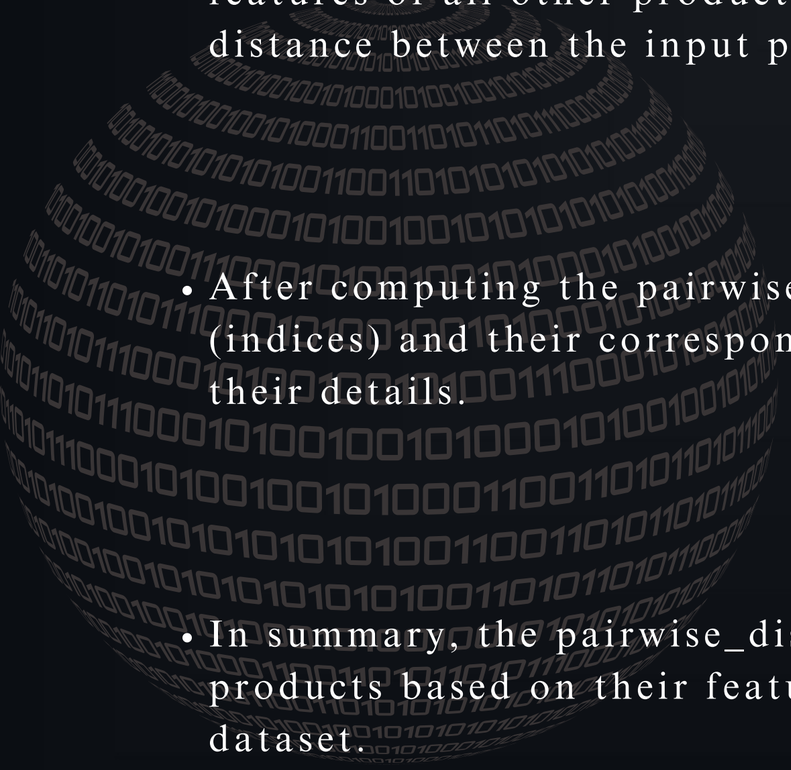
- **Function**: get_similar_products_cnn(product_id, num_results)
- **Input**: Product ID and the desired number of similar products (num_results)
- **Process**:
  - Finds the index of the input product ID in the list of product IDs.
  - Calculates pairwise distances between the input product and all other products using CNN-based feature extraction.
  - Retrieves the indices of the most similar products.
  - Displays the input product image and details.
  - Displays the recommended products along with their details and Euclidean distances from the input image.

Code Breakdown

- **Explanation of key components**:
  - **pairwise_distances**: Utilized to calculate distances between the input product and all other products based on CNN features.
  - **Sorting and selecting**: Using NumPy to identify the most similar products based on calculated distances.
  - **Image and details display**: Leveraging pandas DataFrame to showcase product images and titles.

pairwise_distances()

- pairwise_distances then calculates the pairwise distances between the input product's features and the features of all other products. The result (pairwise_dist) is a 1D array where each element represents the distance between the input product and a corresponding product in the dataset.



- After computing the pairwise distances, the code proceeds to find the indices of the most similar products (indices) and their corresponding distances (pdists). The recommendations are then displayed along with their details.

- In summary, the pairwise_distances function is crucial for comparing the input product with all other products based on their feature representations, and it helps identify the most similar products in the dataset.

Example Usage

- **Example input**: Retrieving 5 similar products for the input product ID '13683'.
- Showcase the output with the input product image and details followed by the recommended products and their details along with Euclidean distances.

Visual Representation

- Provide visual representations of the input product and recommended products, emphasizing the role of CNN-based feature extraction in determining similarity.
- Use images of products along with their titles to illustrate the concept.

Key Takeaways

- **Recap the main points**:
  - CNN-based feature extraction for product similarity.
  - Pairwise distances as a measure of similarity.
  - Output format showcasing input and recommended products.

```python
# Function to retrieve similar products using CNN-based feature extraction
def get_similar_products_cnn(product_id, num_results):
    # Finding the index of the input product ID in the list of product IDs
    doc_id = Productids.index(product_id)

    # Calculating pairwise distances between the input product and all other products
    pairwise_dist = pairwise_distances(extracted_features, extracted_features[doc_id].reshape(1,-1))

    # Finding the indices of the most similar products
    indices = np.argsort(pairwise_dist.flatten())[0:num_results]
    pdists  = np.sort(pairwise_dist.flatten())[0:num_results]
    # Extracting and displaying the input product image and details
    print("="*20, "input product image", "="*20)
    ip_row = men[['ImageURL','ProductTitle']].loc[men['ProductId']==int(Productids[indices[0]])]
    for indx, row in ip_row.iterrows():
        display(Image(url=row['ImageURL'], width = 224, height = 224, embed=True))
        print('Product Title: ', row['ProductTitle'])

    # Displaying the recommended products and their details
    print("\n","="*20, "Recommended products", "="*20)
    for i in range(1,len(indices)):
        rows = men[['ImageURL','ProductTitle']].loc[men['ProductId']==int(Productids[indices[i]])]
        for indx, row in rows.iterrows():
            display(Image(url=row['ImageURL'], width = 224, height = 224, embed=True))
            print('Product Title: ', row['ProductTitle'])
            print('Euclidean Distance from input image:', pdists[i])

# Example: Retrieving 5 similar products for the input product ID '13683'
get_similar_products_cnn('13683', 5)
```

```
==================== input product image ====================
```

```
Product Title:  ADIDAS Men Adi Quest Blue Sports Shoes

==================== Recommended products ====================
```

```
Product Title:  Fila Men Destiny Grey Sports Shoes
Euclidean Distance from input image: 27.668856
```

```
Product Title:  ADIDAS Men Silver Phaedra Shoes
Euclidean Distance from input image: 29.393417
```

Data Loading

- Present the loading of pre-extracted features and corresponding product IDs for different gender categories.
- Emphasize the separation of data for boys' apparel, girls' apparel, men's footwear, and women's footwear.

Data Preparation

- Showcase the conversion of the 'ProductId' column in the 'fashion_df' DataFrame to string type.
- Ensure consistency in data types for effective processing.

```
# Loading previously saved extracted features and corresponding product IDs for boys' apparel
boys_extracted_features = np.load('/kaggle/working/Boys_ResNet_features.npy')
boys_Productids = np.load('/kaggle/working/Boys_ResNet_feature_product_ids.npy')

# Loading previously saved extracted features and corresponding product IDs for girls' apparel
girls_extracted_features = np.load('/kaggle/working/Girls_ResNet_features.npy')
girls_Productids = np.load('/kaggle/working/Girls_ResNet_feature_product_ids.npy')

# Loading previously saved extracted features and corresponding product IDs for men's footwear
men_extracted_features = np.load('/kaggle/working/Men_ResNet_features.npy')
men_Productids = np.load('/kaggle/working/Men_ResNet_feature_product_ids.npy')

# Loading previously saved extracted features and corresponding product IDs for women's footwear
women_extracted_features = np.load('/kaggle/working/Women_ResNet_features.npy')
women_Productids = np.load('/kaggle/working/Women_ResNet_feature_product_ids.npy')

# Converting the 'ProductId' column in the 'fashion_df' DataFrame to string type
fashion_df["ProductId"] = fashion_df["ProductId"].astype(str)
```

Recommender Function

- Present the function get_similar_products_cnn that retrieves similar products based on CNN-based feature extraction.
- Explain how the gender category of the input product determines the choice of pre-extracted features and product IDs.

Function Execution

- Provide an example of how the function is executed.
- Emphasize the determination of the gender category and the subsequent selection of relevant features.

Results

- Illustrate the results with images and details of the input product and recommended products.
- Highlight the calculated Euclidean distance as a measure of similarity.

```python
# Function to retrieve similar products using CNN-based feature extraction
def get_similar_products_cnn(product_id, num_results):
    # Determining the gender category of the input product
    gender_category = fashion_df[fashion_df['ProductId'] == product_id]['Gender'].values[0]

    # Selecting the appropriate set of pre-extracted features and product IDs based on the gender category
    if gender_category == "Boys":
        extracted_features = boys_extracted_features
        Productids = boys_Productids
    elif gender_category == "Girls":
        extracted_features = girls_extracted_features
        Productids = girls_Productids
    elif gender_category == "Men":
        extracted_features = men_extracted_features
        Productids = men_Productids
    elif gender_category == "Women":
        extracted_features = women_extracted_features
        Productids = women_Productids

    # Converting the product IDs to a list
    Productids = list(Productids)

    # Finding the index of the input product ID in the list of product IDs
    doc_id = Productids.index(product_id)

    # Calculating pairwise distances between the input product and all other products
    pairwise_dist = pairwise_distances(extracted_features, extracted_features[doc_id].reshape(1,-1))

    # Finding the indices of the most similar products
    indices = np.argsort(pairwise_dist.flatten())[0:num_results]
    pdists  = np.sort(pairwise_dist.flatten())[0:num_results]
    # Extracting and displaying the input product image and details
    print("="*20, "input product details", "="*20)
    ip_row = fashion_df[['ImageURL','ProductTitle']].loc[fashion_df['ProductId']==Productids[indices[0]]]
    for indx, row in ip_row.iterrows():
        display(Image(url=row['ImageURL'], width = 224, height = 224, embed=True))
        print('Product Title: ', row['ProductTitle'])

    # Displaying the recommended products and their details
    print("\n","="*20, "Recommended products", "="*20)
    for i in range(1,len(indices)):
        rows = fashion_df[['ImageURL','ProductTitle']].loc[fashion_df['ProductId']==Productids[indices[i]]]
        for indx, row in rows.iterrows():
            display(Image(url=row['ImageURL'], width = 224, height = 224, embed=True))
            print('Product Title: ', row['ProductTitle'])
            print('Euclidean Distance from input image:', pdists[i])
```

THANK YOU