Team 13 - Final Project Report

Deep Learning Based Visual Product Recommendation System

1. Objective

The primary objective of this project is to develop a **Deep Learning-based Visual Product Recommendation System** that utilizes convolutional neural networks (CNNs) to recommend visually similar products based on an input image. The system aims to seamlessly integrate image data with metadata to enhance user experience on e-commerce platforms by providing personalized and contextually relevant product suggestions.

2. Data Summary

The Fashion Product Images Dataset is designed for e-commerce applications, combining **44,441 product images** with detailed metadata, including attributes like id, gender, masterCategory, subCategory, articleType, baseColour, season, and usage. The dataset enables both visual feature extraction using CNNs and contextual analysis through metadata, making it ideal for building a recommendation system that delivers personalized, visually similar, and contextually relevant product suggestions to enhance user experience and drive sales.

2.1 Data Loading

Summary: Since the image data set was significantly big, we could not proceed without uploading the data into the google drive and then directly access the data from there. Hence, providing the link to the Kaggle dataset in case there is any issue in loading the data

– Link (https://www.kaggle.com/datasets/paramaggarwal/fashion-product-images-small/data?select=images)

Steps:

1. **Image Loading:**
    - Authenticate and access Google Drive to extract the image dataset (images.zip) into a specified directory.
    - Verify and list all image files, ensuring availability for further processing.
    - Display a sample image to confirm correct loading.
2. **Metadata Loading:**

- Retrieve metadata from a Google Sheet using the URL and convert it into a pandas DataFrame.
- Verify the number of records and examine the first few rows to understand the dataset structure.

2.2 Data Cleaning

Details:

The data cleaning process ensures the integrity of the dataset by handling missing values, aligning images with metadata, and filtering out incomplete records.

Steps:

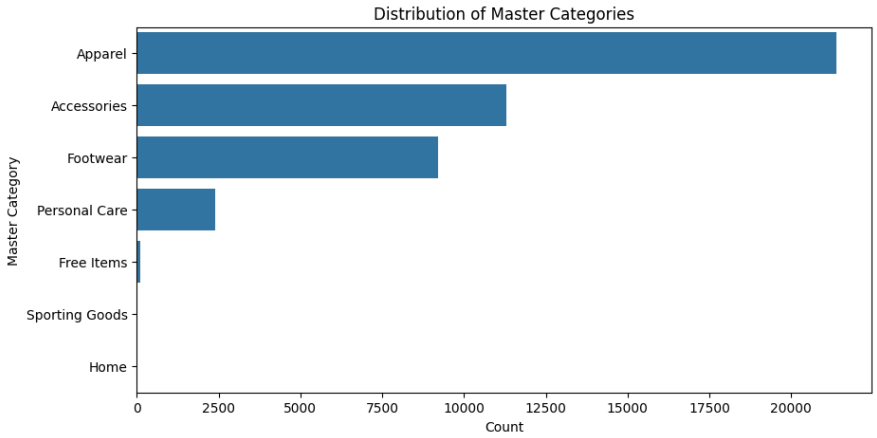
1. **Identify Missing Values:**
   - Check for null values in critical metadata columns (id, masterCategory, subCategory, articleType) and ensure their completeness.
2. **Remove Incomplete Metadata:**
   - Drop records with missing critical fields to maintain dataset consistency.
3. **Map Images to Metadata:**
   - Generate filenames for images (id.jpg) and match them against the loaded image files.
   - Identify and separate records with missing or unmatched image files.
4. **Filter Dataset for Valid Records:**
   - Remove records with missing image files (5 records dropped), ensuring each product has both metadata and an image.
   - Confirm the cleaned dataset size aligns with the available images (44,441 records).

---

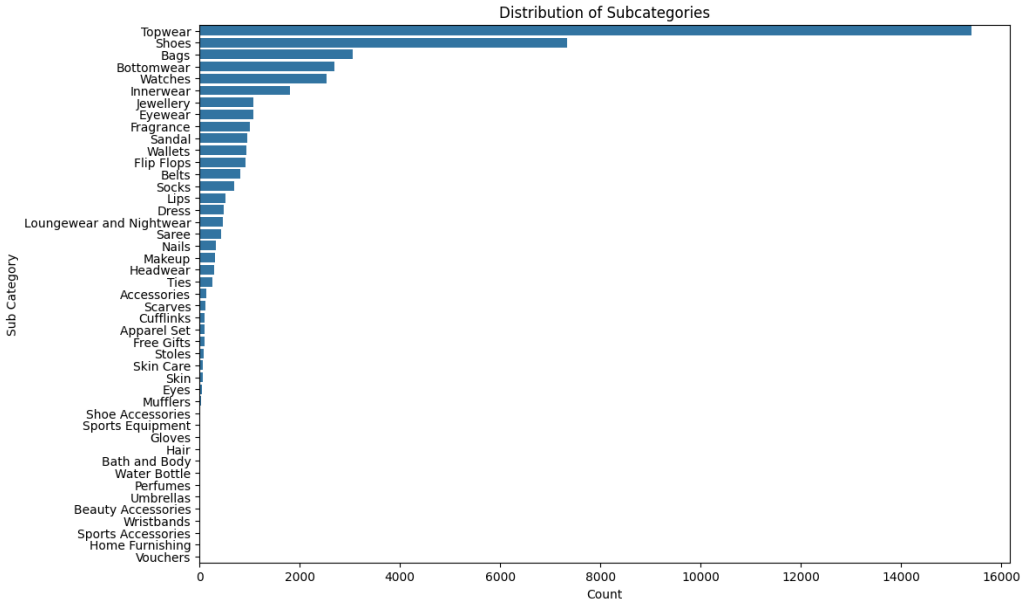
3. Exploratory data analysis

The visualizations highlight the distribution of key categorical variables in the dataset: “masterCategory”, “subcategory”, and “articleType”. Here's the summary:

1. **Master Category Distribution:**
    - The dataset is dominated by **Apparel**, which has the highest count, followed by Accessories and Footwear.
    - Categories like Free Items, Sporting Goods, and Home have minimal representation, indicating less diversity in these groups.

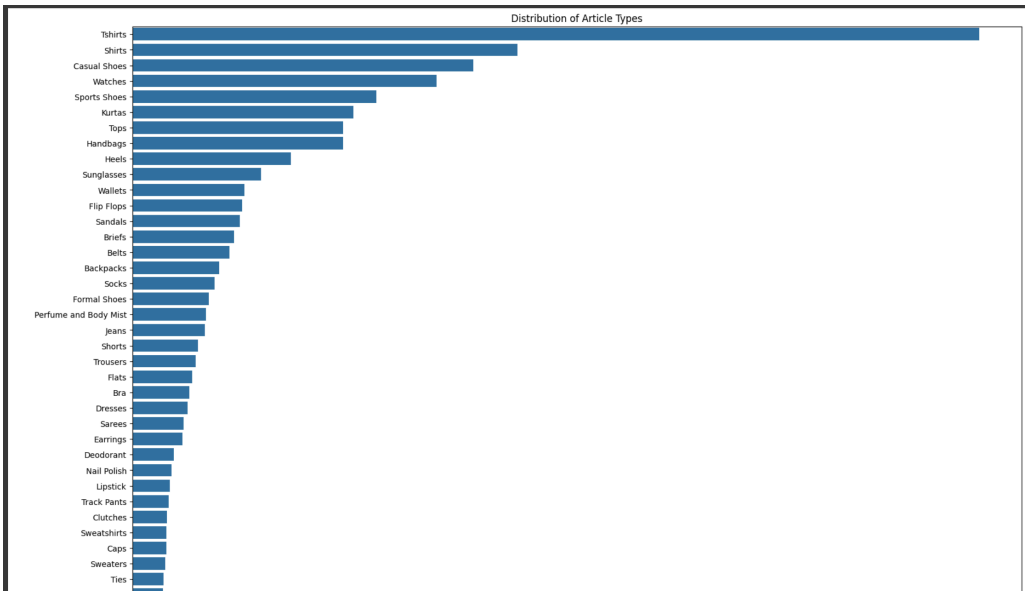


2. **Subcategory Distribution:**
    - Within subcategories, **Topwear**, Shoes, and Bags have the highest frequency, showcasing a higher focus on these product types.
    - Categories such as Headwear, Ties, and Scarves have significantly lower counts, implying limited representation.



3. **Article Type Distribution:**
    - **Tshirts, Shirts**, and Casual Shoes are the most common article types in the dataset, reflecting popular fashion trends.
    - Rarely occurring types like Eye Cream, Key Chain, and Cufflinks have minimal entries, providing opportunities for niche analysis.

Below image is just a snippet of the full picture:



---

4. Data Pre-Processing

1. Sub-setting the Dataset

Since the model training of 44,000 images kept failing in the first time we ran the model with full data set we are sub setting the images data to include the images from the most populated categories as mentioned below.

2. Label Encoding

Categorical columns (masterCategory, subCategory, articleType, baseColour) were encoded numerically using LabelEncoder, with rare classes (fewer than two samples) removed to maintain balance. A category_encodedcolumn was added for modeling.

3. Data Splitting

The subset was split into training (5,511 samples), validation (613 samples), and testing (1,532 samples) using stratified train_test_split to preserve class distribution.

4.Preparing Data Generators
Image preprocessing (rescaling, resizing to 128x128, horizontal flipping) was performed using ImageDataGenerator. Generators were configured to load images in batches of 64 while maintaining class balance across train, validation, and test sets.

---

5. Model

Baseline Model

1. **Architecture:**
    - **Convolutional Layers:** Three layers with increasing filters (32, 64, 128) and ReLU activation.
    - **Pooling:** MaxPooling layers after each convolutional block to reduce spatial dimensions.
    - **Fully Connected Layers:** A dense layer with 128 neurons followed by a dropout of 50%.
    - **Output Layer:** Softmax activation for multi-class classification.
2. **Performance:**
    - **Training Accuracy:** 69.83%
    - **Test Accuracy:** 69.65%
    - The baseline model demonstrated decent performance, but improvements were required to handle class imbalance and boost accuracy.

Optimization Steps

Method 01 - Batch Normalization (Model 1)

1. **Changes Made:**

- Added Batch Normalization layers after each convolutional layer to stabilize learning and improve generalization.
- Replaced the Flatten layer with GlobalAveragePooling2D for better spatial information retention.
- Increased the dense layer size to 256 neurons.

2. **Performance:**
   - **Test Accuracy:** Dropped to 28.98%
   - Despite the architectural changes, the model overfitted to training data and failed to generalize.

Method 01.1 - Further Tuning (Model 2)

1. **Changes Made:**
   - Reverted to the original baseline structure but added:
     - A new dense layer with 256 neurons before the existing 128-neuron layer.
     - Two dropout layers (50%) for regularization.
   - Increased training epochs and adjusted hyperparameters for better convergence.
2. **Performance:**
   - **Training Accuracy:** Gradual improvement over epochs, reaching above 68%.
   - **Validation Accuracy:** Exceeded 78% during training.
   - **Test Accuracy:** Achieved 72.13%, showing improved generalization over both baseline and Model 1.

Final Model (Model 2)

1. **Architecture:**
   - **Convolutional Layers:** Three blocks (32, 64, 128 filters) with MaxPooling.

- **Dense Layers:** Two dense layers (256 and 128 neurons) with ReLU activation and dropout.
- **Output Layer:** Softmax for multi-class classification.

2. **Performance Highlights:**
    - **Test Accuracy:** 72.45% (best among all iterations).
    - Achieved a balance between training and test accuracies, demonstrating better generalization compared to previous versions.
3. **Improvements Over Baseline:**
    - More robust feature extraction with additional dense layers.
    - Effective use of dropout for regularization.
    - Enhanced test accuracy from 69.65% to 72.45%.

---

6. Recommendation System

1. Process Overview

1. **Model Loading:**
    - The pre-trained model is loaded to generate feature embeddings for images. These embeddings represent high-level visual features such as color, texture, and patterns.
2. **Image Preprocessing:**
    - All images (including the target) are resized to 128x128 pixels, normalized (pixel values scaled to [0, 1]), and converted into arrays to match the model's input requirements.
3. **Feature Extraction for the Target Image:**
    - The target image is processed through the model to generate its feature embedding. This embedding serves as a baseline for comparing other images.
4. **Batch Processing for Efficiency:**
    - To handle a large dataset efficiently, images are processed in batches. Each batch is passed through the model to generate feature embeddings.

5. **Cosine Similarity Calculation:**
    - For each image in the dataset, the cosine similarity between its feature embedding and the target image's embedding is calculated.
    - Cosine similarity measures the angular similarity between two vectors, making it ideal for comparing high-dimensional data.
6. **Sorting and Ranking:**
    - All images are ranked based on their cosine similarity scores. The higher the score, the more visually similar the image is to the target.
    - **The top 5 images with the highest scores are selected as recommendations**.

Final Input Image & the top 5 recommendations:

Input **Outputs**

---

2. Business Benefits

- **Enhanced User Experience:**
    - Provides personalized and intuitive recommendations based on visual similarity.
- **Increased Sales:**
    - Encourages cross-selling and upselling by showcasing visually similar alternatives.
- **Competitive Edge:**

- Positions the platform as technologically advanced, leveraging deep learning for enhanced recommendations.

---

7. Deployment & Next Steps

1. Data Integration and Scalability

- **Objective:** Seamlessly integrate the recommendation system into existing data infrastructure while ensuring scalability to handle increasing data volumes as the business expands.
- **Actionable Steps:**
  - Utilize cloud storage solutions for efficient data management.
  - Develop APIs to integrate the recommendation system with e-commerce platforms.

2. Model Generalization and Maintenance

- **Objective:** Adapt to changing user preferences and maintain model relevance over time.
- **Actionable Steps:**
  - Continuously monitor model performance and retrain as necessary using updated datasets.
  - Implement feedback loops to incorporate user interactions and preferences.

3. Optimized Platform to Run the Model

- **Objective:** Ensure reliable and efficient model execution, especially when processing high-resolution images.
- **Actionable Steps:**
  - Deploy the model on optimized platforms such as advanced GPUs or cloud-based services.
  - Use containerized environments like Docker for portability and consistent performance.

4. Integrate Other Factors

- **Objective:** Enhance recommendation accuracy by including additional contextual and transactional data.
- **Actionable Steps:**
    - **Incorporate factors such as purchase history, user reviews, and ratings** into the recommendation algorithm.
    - Develop hybrid models that combine visual similarity with behavioral data for improved relevance.