ResNet (Residual Network):

ResNet (Residual Network) is a deep convolutional neural network that revolutionized deep learning by introducing residual connections, which mitigate the vanishing gradient problem, enabling the effective training of very deep networks. It is widely used for image similarity search because its pre-trained versions (e.g., ResNet-50, ResNet-101) can extract rich, high-dimensional feature embeddings from images, capturing complex patterns and structures. These embeddings are particularly suitable for measuring image similarity using metrics like cosine or Euclidean distance.

ResNet-101 is an extension of ResNet-50, featuring 101 layers compared to ResNet-50's 50 layers. This increased depth allows ResNet-101 to capture more complex and hierarchical features from images, which can improve accuracy, especially for tasks involving fine-grained details or large-scale datasets. Both models use residual connections to prevent vanishing gradient issues and bottleneck blocks to optimize computation. While ResNet-50 is faster and requires less memory, ResNet-101 generally achieves better performance due to its deeper architecture, making it more suitable for tasks where accuracy is critical, such as image classification, similarity search, and segmentation. However, the added depth comes with increased computational cost and latency, which may not be ideal for resource-constrained environments. Thus, ResNet-101 is preferred for accuracy-focused applications, while ResNet-50 is favored for efficiency. The reason for choosing ResNet-101 for this task is because it is important that on fashion and e-commerce platforms the matching is more accurate and precise. Hence, for this task I had chosen ResNet-101 instead of ResNet-50

ResNet (Residual Network) offers several key advantages that make it one of the most popular architectures in deep learning. Its revolutionary residual connections effectively address the vanishing gradient problem, enabling the training of very deep networks with hundreds of layers while maintaining stability and performance. ResNet provides state-of-the-art accuracy on image recognition tasks and serves as a robust backbone for advanced applications such as object detection, image segmentation, and similarity search. The architecture is highly effective for transfer learning, as its pre-trained versions (e.g., on ImageNet) can extract generalizable and hierarchical features, making it adaptable to a wide range of domains. Furthermore, ResNet uses bottleneck layers to optimize computational efficiency despite its depth, ensuring a good balance between performance and resource consumption. Its versatility, accuracy, and ability to handle complex tasks have established ResNet as a cornerstone in modern computer vision.However, it has limitations, such as high computational and memory requirements for deeper variants (e.g., ResNet-152), and it may not always be the best choice for tasks requiring lightweight models. Despite this, ResNet remains a popular and robust choice for image similarity tasks due to its accuracy and pre-trained availability.

Implementation of ResNet-101:

1. `preprocess_fmnist(images, sample_size=10000)`

This function preprocesses the Fashion MNIST dataset to prepare it for feature extraction with the ResNet101 model.

- **Steps:**
  - Limits the dataset to the first `sample_size` images for computational efficiency.
  - Converts the grayscale images (28x28) to include a channel dimension (28x28x1), making them compatible with image processing pipelines.
  - Scales pixel values from `[0, 255]` to `[0, 1]` by dividing each pixel value by 255.
  - Resizes each image to 224x224 pixels using TensorFlow's `tf.image.resize` to meet ResNet101's input size requirement.
  - Converts grayscale images to RGB format (224x224x3) by repeating the single grayscale channel three times.
  - Applies `preprocess_input` to adjust pixel values to a range suitable for the ResNet101 model
  - **`sample_size=10000`:** Controls how many images to preprocess for efficiency.
  - **Resizing to `224x224`:** Ensures compatibility with ResNet101's input size.
  - **Normalization (`/255.0`):** Converts pixel intensity values to a range of `[0, 1]`.

2. `extract_features(images, model)`

This function extracts feature vectors for the input images using the ResNet101 model.

- **Steps:**
  - Passes the preprocessed images through the ResNet101 model to compute feature maps. These feature maps are high-dimensional tensors representing learned patterns from the model.
  - Reshapes the feature maps into a 2D array where each row corresponds to a feature vector for an image.
  - **Batch Size:** The model processes images in batches (default for the model), ensuring efficient computation.
  - **Feature Dimensions:** The output features are flattened for use in downstream tasks like similarity matching.

3. `plot_similar_images(query_image, similar_images, class_names)`

This function visualizes the query image alongside its most similar images.

- **Steps:**
    - The first subplot displays the query image.
    - Subsequent subplots show the most similar images, with their predicted or true class names.
    - **`class_names`:** Provides the class labels for images, enhancing interpretability during visualization.

4. `calculate_metrics(true_labels, predicted_labels)`

This function calculates performance metrics for similarity-based retrieval.

- **Steps:**
    - Compares the true labels with the predicted labels and computes the percentage of correct predictions.
    - Calculates precision, recall and f1-score to evaluate the retrieval quality.
    - **Precision and Recall:** Measure the relevance of retrieved images.
    - **F1 Score:** Balances precision and recall for a comprehensive performance metric.

5. `print_metrics_summary(metrics)`

This function displays the calculated metrics in a clear and concise format.

- **Steps:**
    - Retrieves metrics like accuracy, precision, recall, and F1 score from the input dictionary.
    - Outputs the metrics with labels for interpretability.
    - No direct numerical parameters, but it relies on metrics passed to it (accuracy, precision, etc.).

6. `main()`

This function orchestrates the entire workflow, including data preprocessing, feature extraction, similarity computation, and result visualization.

- **Steps:**
    - Calls `preprocess_fmnist` to prepare the dataset.
    - Extracts feature vectors for all images using ResNet101.
    - Identifies similar images for randomly chosen query images.
    - Computes evaluation metrics using `calculate_metrics`.
    - Displays the query images and their most similar counterparts using `plot_similar_images`.
    - Combines all numerical parameters from the individual functions, such as `sample_size`, resizing dimensions (224x224), and evaluation metrics.

Implementation of ResNet-101:

Summary of ResNet101-Based Similarity Search Analysis

The results of the ResNet101-based similarity search reveal its performance in identifying visually similar images.

Query Image: Ankle boot

Observation:

- All five nearest neighbors belong to the same class (`Ankle boot`), with very low distances (0.08).
- The model effectively identifies images from the same class based on shared visual patterns, such as the shape and texture of the footwear.

Analysis:

- ResNet101 excels in retrieving highly similar images within the same class by leveraging robust feature representations.
- The low distances indicate the effectiveness of the feature extraction process in encoding meaningful visual patterns.

Query Image: Sneaker

Observation:

- While the majority of retrieved images belong to the same class (`Sneaker`), one or more images belong to the `Ankle boot` class with similar distances (0.06-0.07).
- The overlap in visual features (e.g., shoe outline, texture) leads to some degree of misclassification.

Analysis:

- ResNet101 captures low-level visual similarities, which can result in retrieval of items from visually related classes.
- This highlights the limitation of feature-based approaches in distinguishing between semantically different but visually similar items.

Query Image: Pullover

Observation:

- The nearest neighbors include images from `Shirt`, `Coat`, and `Pullover` classes, with distances ranging from 0.08 to 0.09.

- While some retrieved images belong to the correct class, others share only general shape or texture characteristics with the query.

---

4. Precision, Recall, and F1-Score Metrics:

The evaluation metrics indicate the effectiveness of the retrieval process at different values of K (number of nearest neighbors retrieved):

Observations:

- **At K=1:**
  - **Precision:** 0.789 (high)
  - **Recall:** 0.002 (very low)
  - **F1-Score:** 0.003
  - The high precision reflects the model's ability to identify the correct match for the closest neighbor. However, the recall is low as only the top neighbor is considered.
- **At K=5:**
  - **Precision:** 0.754
  - **Recall:** 0.008
  - **F1-Score:** 0.015
  - Precision decreases slightly as more neighbors are included, but recall improves marginally.
- **At K=20:**
  - **Precision:** 0.702
  - **Recall:** 0.052
  - **F1-Score:** 0.098
  - Recall improves significantly with higher K, but precision drops due to inclusion of less relevant neighbors.