<u>**Autoencoder:**</u>

An Autoencoder is a type of neural network designed for unsupervised learning that compresses input data into a lower-dimensional representation (latent space) and reconstructs it back to the original dimensions. It is widely used for image similarity search because the latent representations learned by the encoder capture essential features of the input image, discarding noise and irrelevant details. These latent vectors are used to compute similarities (e.g., cosine similarity or Euclidean distance) between images efficiently. Compared to traditional methods like SIFT, which focuses on detecting and matching handcrafted keypoints, autoencoders offer a more flexible and data-driven approach, learning features directly from the dataset without manual feature engineering. Unlike CNN-based models like ResNet, which focus on extracting hierarchical features for classification or prediction tasks, autoencoders are explicitly designed for dimensionality reduction and reconstruction, making them lightweight and computationally efficient for tasks like image retrieval.

Advantages of autoencoders include their ability to learn task-specific features when trained on a specific domain, their compact latent representations, and their adaptability to different datasets. They are computationally less intensive compared to deep CNNs and can be tuned for efficient encoding-decoding operations. Autoencoders can reduce high-dimensional data (e.g., images) into compact latent representations, which are computationally efficient for similarity comparisons, clustering, or classification tasks.

However, disadvantages include their dependency on the quality of the training data, as poor training can lead to overfitting or incomplete feature representations. Additionally, autoencoders are not inherently scale or rotation-invariant, unlike SIFT, and they often struggle with semantically rich comparisons since the focus is primarily on reconstruction rather than feature generalization. Autoencoders require a large and diverse dataset to learn meaningful representations. Poor or biased training data can lead to overfitting or unrepresentative feature extraction. Despite these limitations, autoencoders remain a strong candidate for tasks involving unsupervised feature learning and efficient image similarity search.

<u>**Implementation of Autoencoder:**</u>

1. `preprocess_data()`

- Loads the Fashion MNIST dataset using the `keras.datasets.fashion_mnist` module.
- Scales the pixel values of images to the range `[0, 1]` by dividing them by 255.
- Expands the dimensions of the images to add a channel dimension (required for the CNN input).
- 255: The divisor used to normalize pixel values from `[0, 255]` to `[0, 1]`, which helps in faster and stable model training.

2. `build_autoencoder()`

- Encoder: Compresses the input image into a lower-dimensional feature space using `Conv2D` and `MaxPooling2D`.
- Decoder: Reconstructs the original image from the compressed feature space using `UpSampling2D` and `Conv2D`.
- Compiles the autoencoder using the Adam optimizer and `binary_crossentropy` loss.
- `Input(shape=(28, 28, 1))`: Specifies the input size for 28x28 grayscale images.
- `Conv2D(32, (3, 3))`:
    - 32: Number of filters (output feature maps).
    - (3, 3): Size of the filter/kernel applied during convolution.
- `MaxPooling2D((2, 2))`: Downsamples the spatial dimensions by a factor of 2 (height and width).
- `UpSampling2D((2, 2))`: Upsamples the spatial dimensions by a factor of 2.
- `binary_crossentropy`: Loss function to measure the difference between the input and reconstructed output.

3. `train_autoencoder(autoencoder, x_train, x_test)`

- Trains the autoencoder using the training dataset.
- Validates the model during training using the test dataset.
- `epochs=20`: Specifies that the model will make 20 passes over the entire training dataset.
- `batch_size=128`: Divides the training data into batches of 128 samples for each training iteration. Larger batches require more memory but result in faster training.
- `shuffle=True`: Shuffles the training data before each epoch to improve generalization.
- `validation_data=(x_test, x_test)`: Uses the test data as validation during training to monitor performance.

4. `generate_embeddings(encoder, x_test)`

- Extracts feature embeddings from the encoder part of the trained autoencoder for the test data.
- Uses these embeddings to represent the images in a compressed feature space.
- None explicitly, but the dimensionality of the embeddings depends on the encoder architecture.

5. `similar_images(embeddings, selected_indices, top_k)`

- Computes the Euclidean distance between the feature embeddings of the selected query images and all other images.
- Finds the indices of the top-`k` most similar images for each query image (excluding the query image itself).
- `top_k`: Specifies the number of similar images to retrieve for each query image. For example, if `top_k=5`, the function will return the indices of the 5 most similar images.

6. `evaluate_similarity(x_test, selected_indices, similar_images_indices, labels, top_k)`

- For each query image:
  - Computes precision, recall, and retrieval accuracy based on the labels of the query and retrieved images.
- Calculates the average precision, recall, and retrieval accuracy across all query images.
- `top_k`: Number of retrieved images considered for evaluation.
- Precision, recall, and retrieval accuracy:
  - Precision: Fraction of retrieved images that are relevant.
  - Recall: Fraction of relevant images that are retrieved.
  - Retrieval Accuracy: Fraction of queries where the most similar image is correctly identified.

7. `plot_similar_images(x_test, selected_indices, similar_images_indices, top_k)`

- Visualizes the original query images and their most similar counterparts in a grid format.
- Each row corresponds to one query image:
  - The first column shows the query image.
  - The remaining columns show the retrieved similar images.
- `top_k`: Number of similar images to display for each query image.
- Height of the grid = number of query images.
- Width of the grid = `top_k + 1` (one column for the query image, `top_k` columns for similar images).

Summary of the Autoencoder Results

Top-5 Similarity Results:

- **Precision:** `0.8000`
  This indicates that 80% of the retrieved images in the top-5 for each query image are relevant (belong to the same class as the query image).
- **Recall:** `0.0040`
  This suggests that the retrieved images account for only a small fraction of all relevant images for the query class.
- **Retrieval Accuracy:** `1.0000`
  This means that for 100% of the query images, the most similar image retrieved was correctly classified (matches the query image class).

Visualization Observations:

- **Consistency:**
  - The retrieved images for all query images closely match their respective classes (e.g., sweaters are matched with sweaters, and pants are matched with pants).
  - This high accuracy is reflected in the retrieval accuracy of 1.0000.
- **Outliers:**
  - There are no noticeable outliers in the top-5 retrieval, indicating strong performance in identifying visually and structurally similar items.

Top-10 Similarity Results:

- **Precision:** `0.7333`
  This indicates that 73.33% of the retrieved images in the top-10 are relevant.
- **Recall:** `0.0073`
  Similar to the top-5 case, the recall remains low because only a small fraction of all relevant images is retrieved.
- **Retrieval Accuracy:** `1.0000`
  Similar to the top-5 results, for 100% of the queries, the most similar image is correctly classified.

Visualization Observations:

- **Consistency:**
  - The retrieved images remain largely consistent with the query class, even with the broader range of top-10 retrievals.
  - Sweaters and pants are accurately matched with items from the same class.
- **Diversity:**
  - While most images in the top-10 are accurate, a few results appear to have slight variations in patterns or textures.