Supervised and Unsupervised Similarity Methods in Deep Learning

Project 3A – Mines Nancy – Théa Chaduteau

February 12th, 2025

Table of contents

Context of the project

The ability to estimate similarity between two objects is a cornerstone of numerous data analysis algorithms

Application in diverse domains

- Facial recognition
- Customer classifications
- Recommendation systems

The power of similarity learning

- Adress this challenge by leveraging models that evaluate similarity **based on the data itself**
- Supervised models: Neural architectures
- Unsupervised methods ?

Goal definition of the project

- Investigating **state-of-the-art similarity learning methods**, such as embeddings and clustering-based approaches;
- Establishing **robust definitions of similarity** that can generalize to real-world applications, such as classification or recommendation systems;
- Testing and developing neural architectures capable of **handling 1D, 2D, and 3D data** modalities;
- Developing **unsupervised models** from scratch capable of handling 1D and 2D data modalities.

Timeline of the project

1. **Litterature Review and Exploration**
   - In-depth review of existing methods
2. **Prototyping and Initial Testing**
   - Develop and test initital supervised models
3. **Advanced Implementation and Analysis**
   - Refine the models and explore unsupervised methods
4. **Final Evaluation and Reporting**
   - Prepare a comprehensive reports and present the results

Table of contents

Defining similarity: a challenge

Feature-based similarity

Based on the **shared features** or attributes of the objects

Transformation-based similarity

Based on the lens of **transformational efforts**

Semantic similarity

Based on the **meanings** or **context** of the objects

Relationnal similarity

Based on their **relationship to other objects**

Different measures of similarity

Distance metrics

- Euclidean
- Cosine
- Manhattan
- Jaccard
- Minkowski
- Chebyshev

Edit-based distance measures

- Hamming
- Levenshtein
- Damerau Levenshtein
- Jaro-Winkler

Statistics-based measures

- Pearson's correlation
- Spearman's rank correlation coefficient

Types of similarity learning

Regression similarity learning

- (x_i^1, x_i^2) a pair of objects / $y_i \in \mathcal{R}$ measure of their similarity
- Goal: Learn a function that approximates $f(x_i^1, x_i^2) \sim y_i$

Classification similarity learning

- (x_i, x_i^+) similar objects / (x_i, x_i^-) non similar objects / $y_i \in \{0; 1\}$ label
- Goal: Again, learn a classifier that can decide if a new pair of objects is similar or not

Ranking similarity learning

- (x_i, x_i^+, x_i^-) triplets with ordered similarity
- Goal: Learn a function where $f(x_i, x_i^+) > f(x_i, x_i^-)$ (contrastive learning)

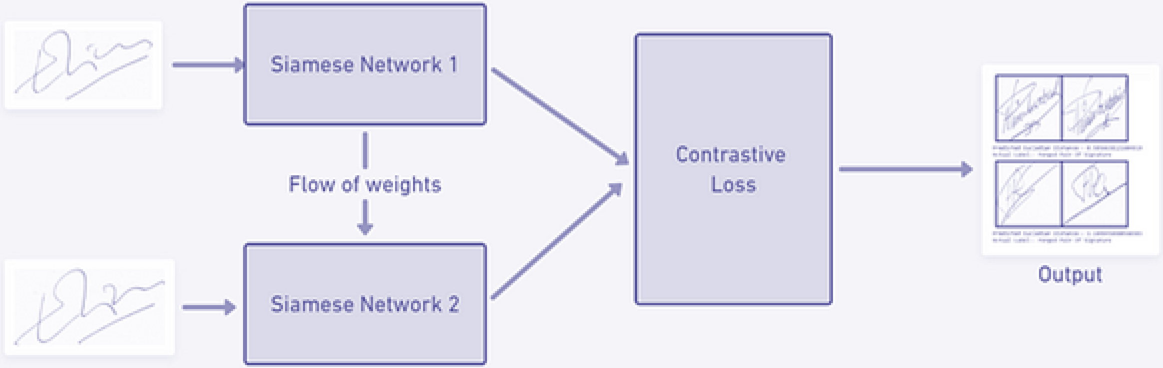
Supervised methods

Structure of a Siamese Network

- Two **identical subnetworks.**
- **Shared weights.**
- **Feature extraction** to represent the content or characteristics of the input in a lower-dimensional space.
- Use of a **distance metric** to measure how close or far apart the representations are.

How it works in practice

- **Training**: Learns from pairs of input example either "similar" or "dissimilar“ aiming to minimize the difference between similar input and reciprocately for dissimilar ones.
- **Contrastive Loss Function**: Penalizes the network if the distance between similar inputs is too large or the distance between dissimilar inputs is too small.

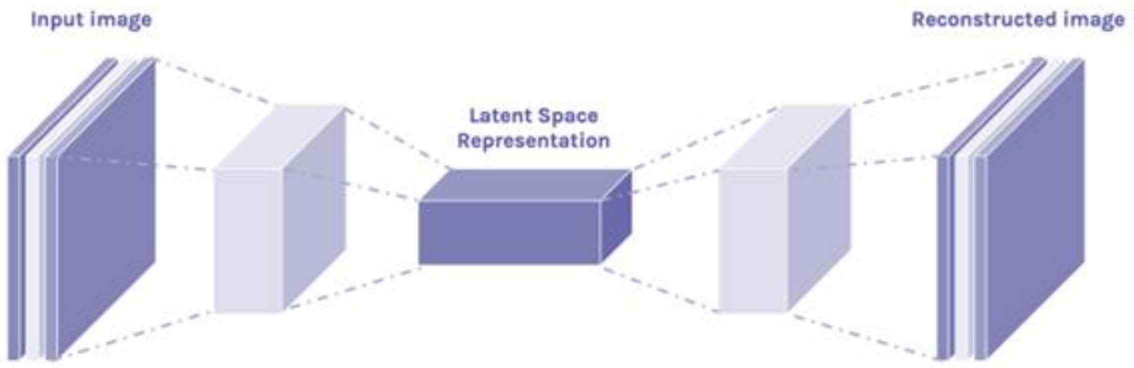


Not all supervised methods are highlighted today, we only display the most common one

Unsupervised methods

Autoencoders

- Unsupervised neural network for learning compact representations.
- Encoder maps input data to a lower-dimensional latent space.
- Decoder reconstructs the original input from the latent representation.
- Similar inputs map to closer representations in latent space.



Partially ordered sets

- A partially ordered set (poset) is a set with a rule for ordering elements, but not every pair must be comparable.
- Visualized using a Hasse diagram, where higher elements are "greater" in the order.
- Found in hierarchies, task scheduling, and dependencies (e.g., prerequisite courses).

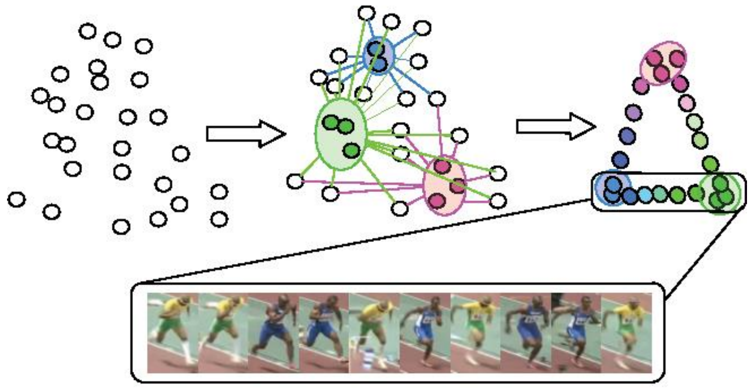
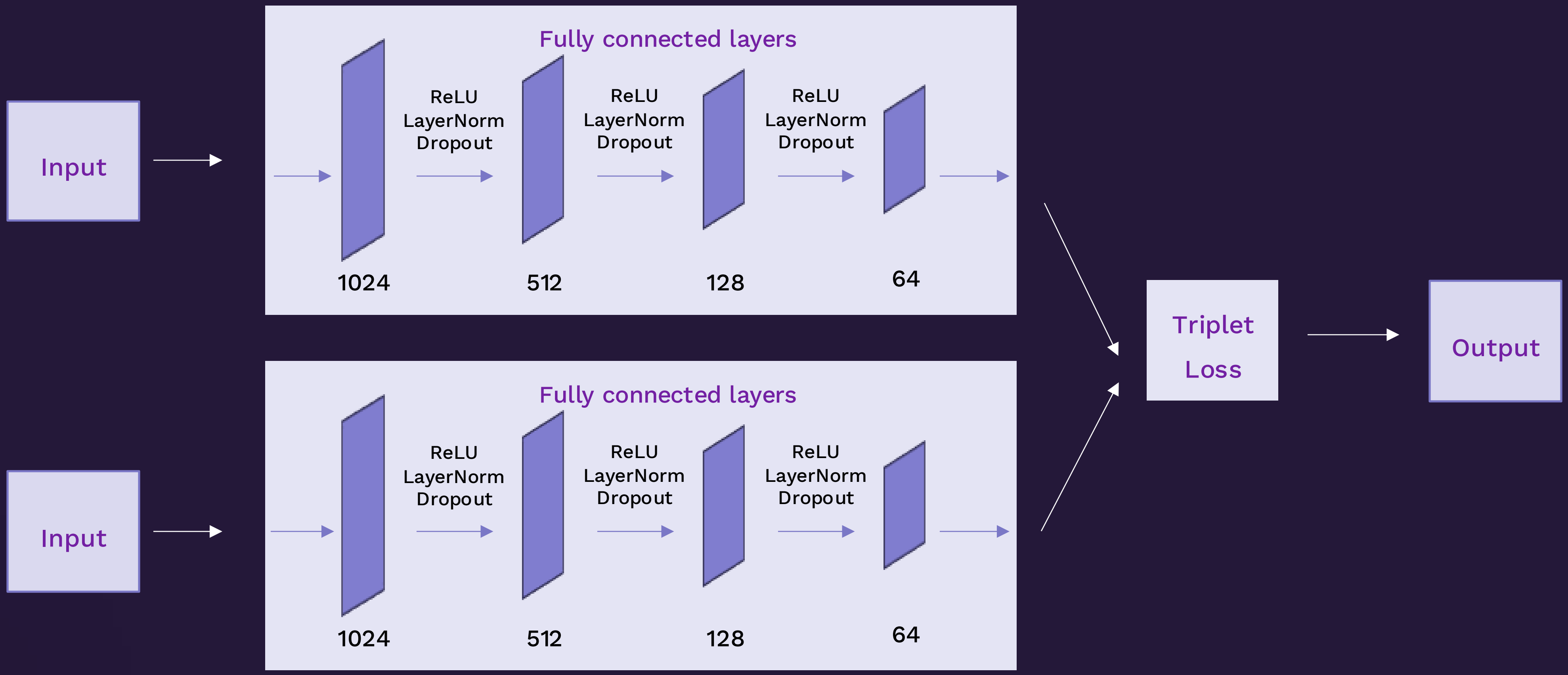
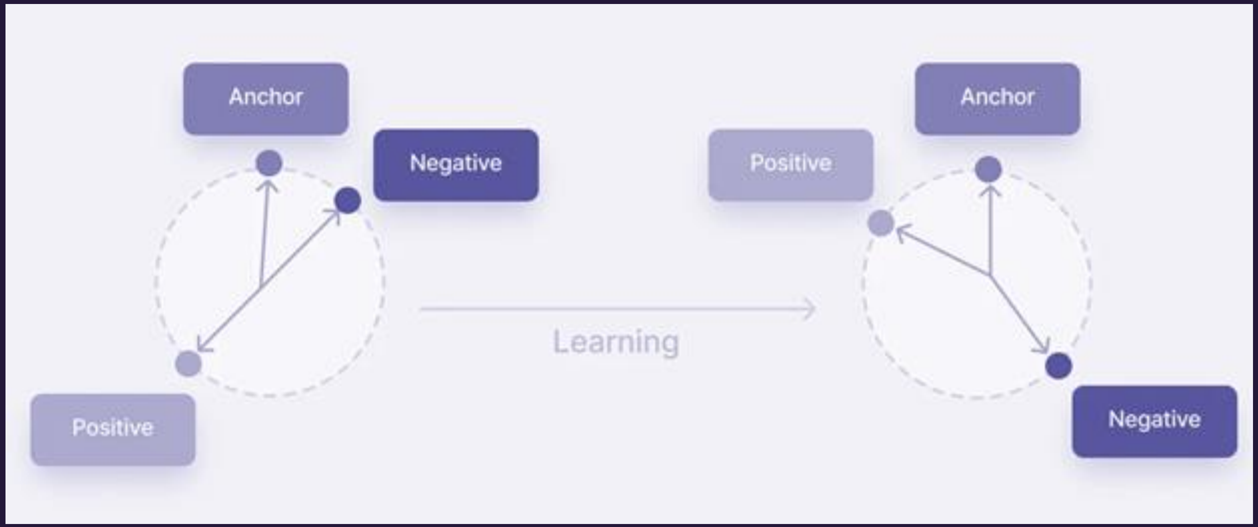


Table of contents

Architecture of our Siamese Network (1/3)



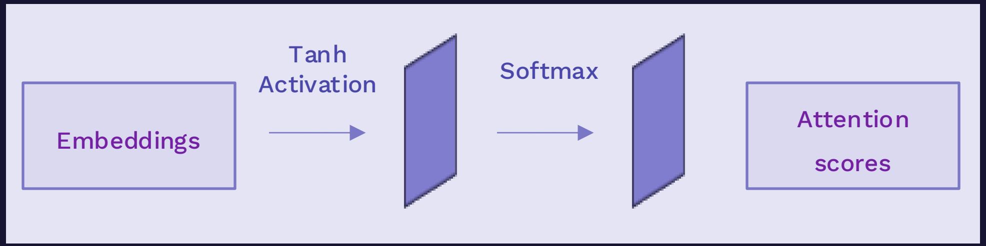
Architecture of our Siamese Network (2/3)



- Triplet loss extends contrastive loss by using three data points: anchor, positive, and negative.
- Objective: Ensure the anchor is closer to the positive than the negative by a margin.
- Distance metric used: Euclidean, cosine and a combination of both.

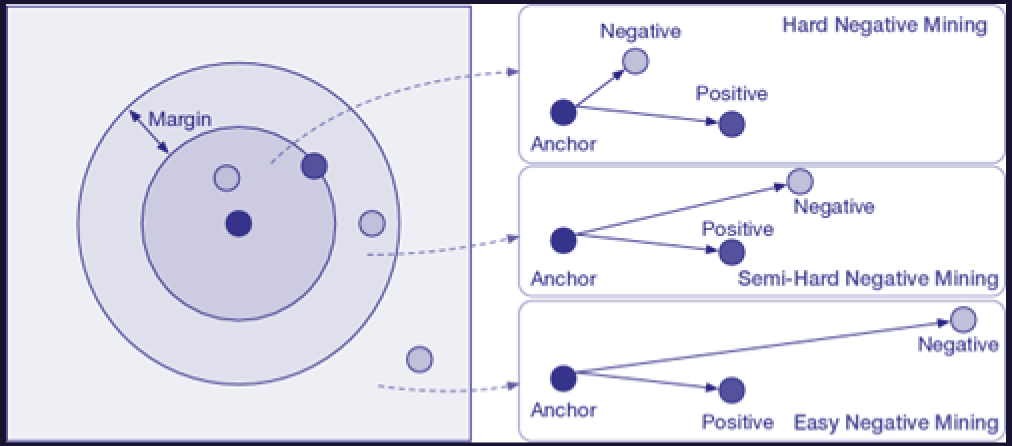
Architecture of our Siamese Network (3/3)

Dynamic margin



- Enhances the network by **dynamically weighting embedding dimensions.**
- Focuses on relevant features for similarity computation.
- Scores sum to 1 and weight embeddings dynamically.
- Stability measures:
  - Attention weights clamped between 0 and 1.
  - L2 normalization applied to weighted embeddings.

Hard Negative Mining

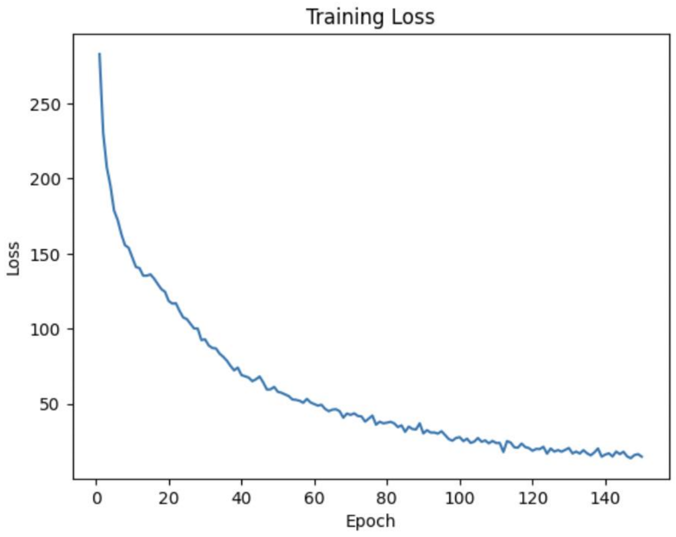


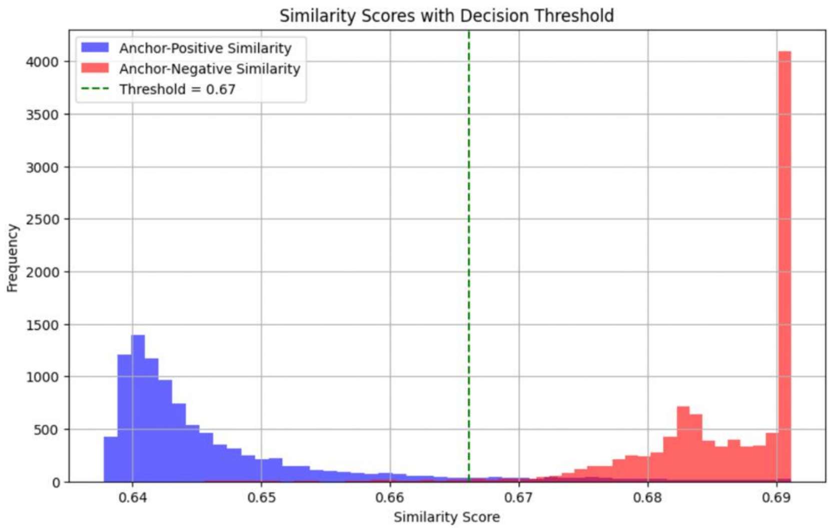
- **Hard negative mining** selects negatives that are difficult to distinguish from positives.
- Prevents wasted training on easy negatives that provide little optimization value.
- **Dynamic triplet batching** improves efficiency and diversity in training.

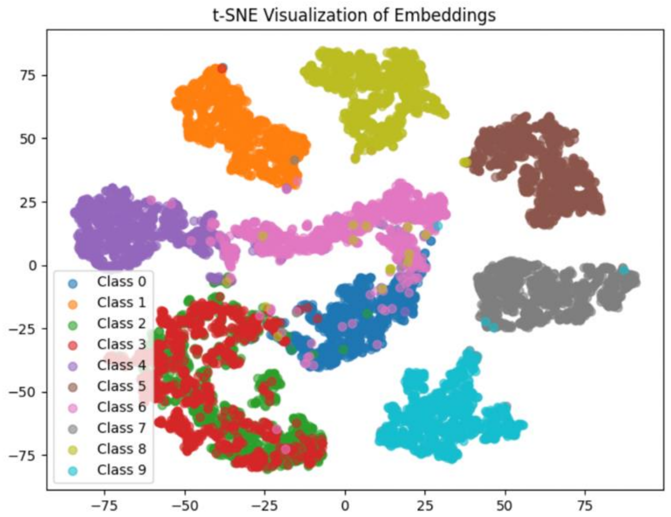
Training results (1/3)

Euclidean distance

Test Accuracy: 0.0102



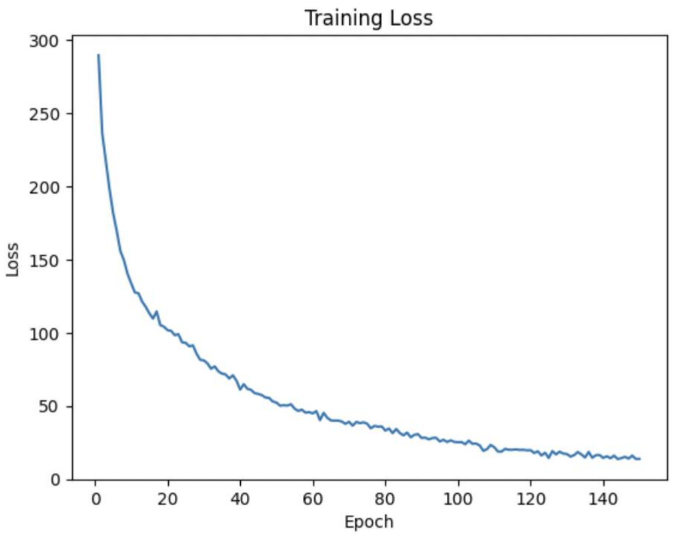


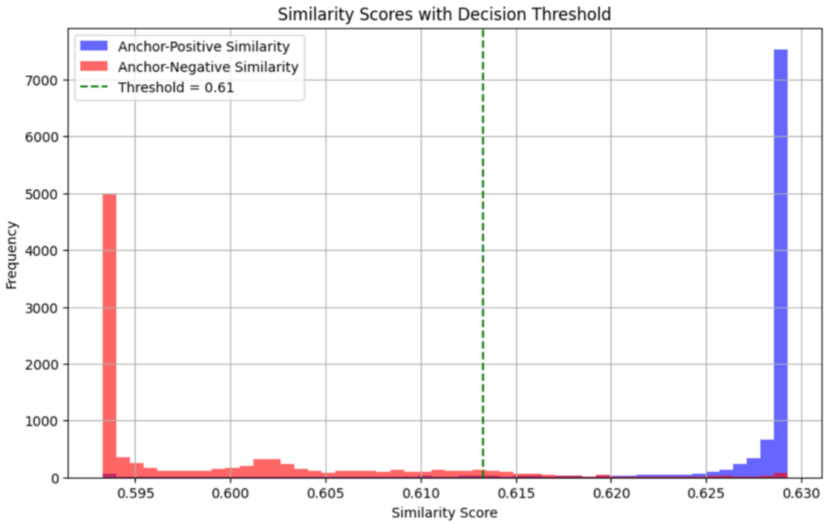


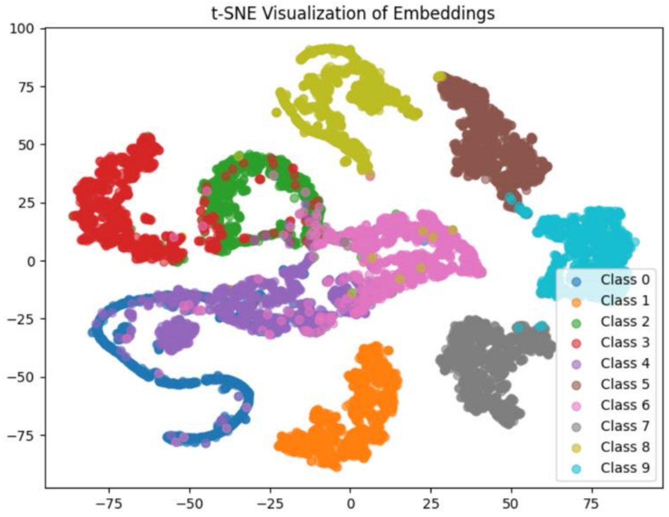
Training results (2/3)

Cosine distance

Test Accuracy: 0.9827



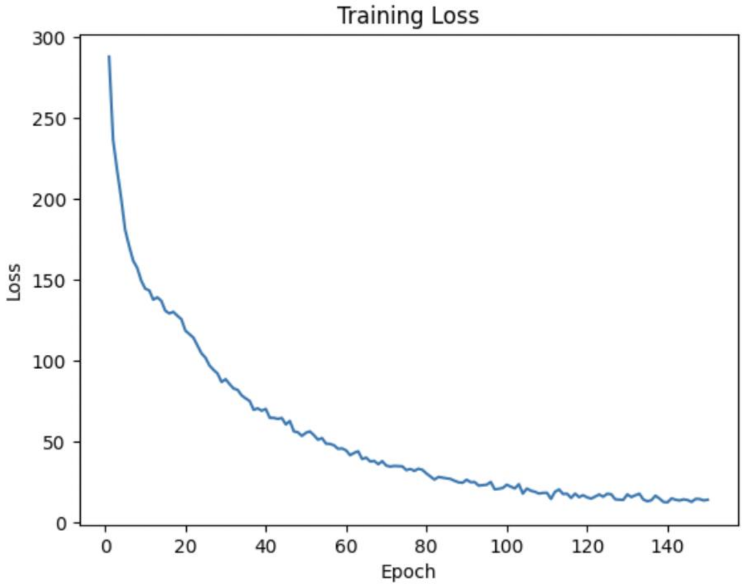


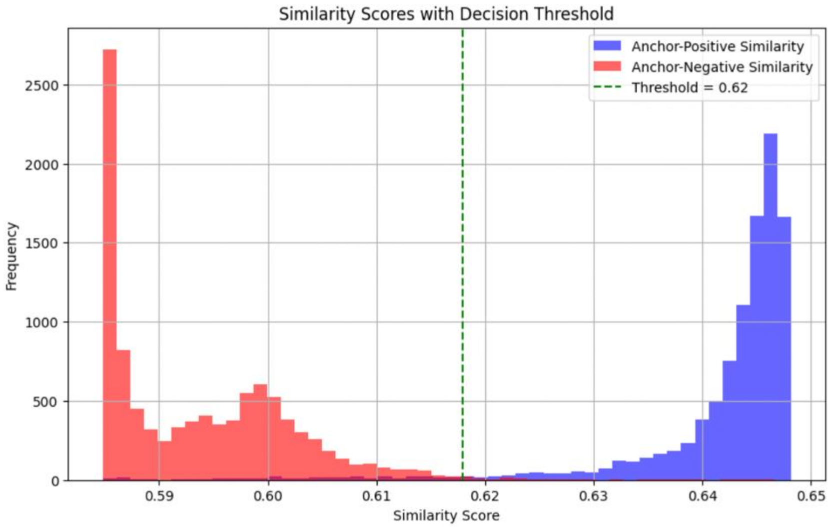


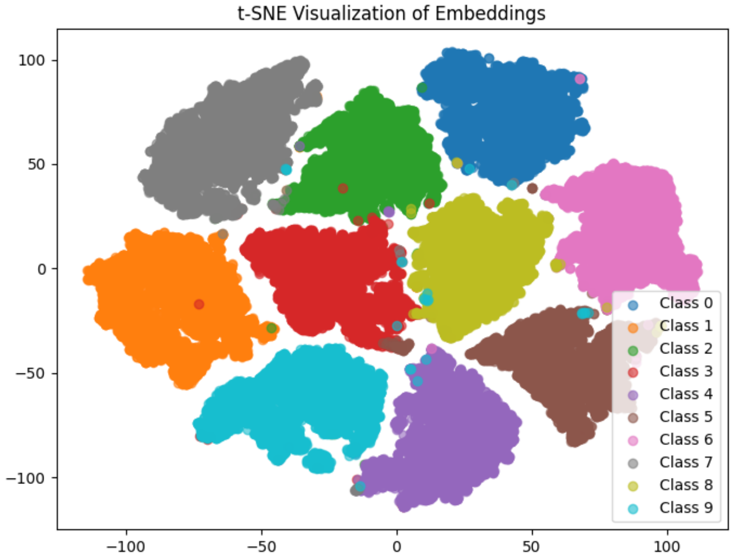
Training results (3/3)

Combination

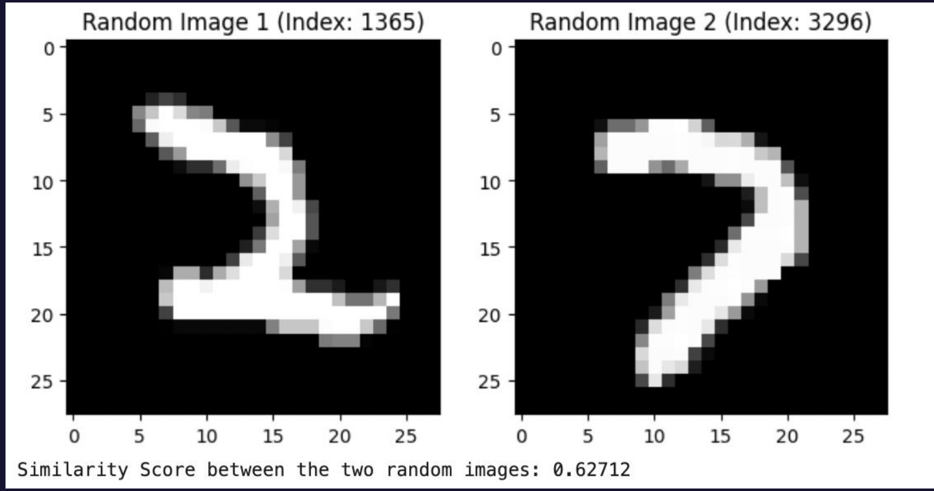
Test Accuracy: 0.9930

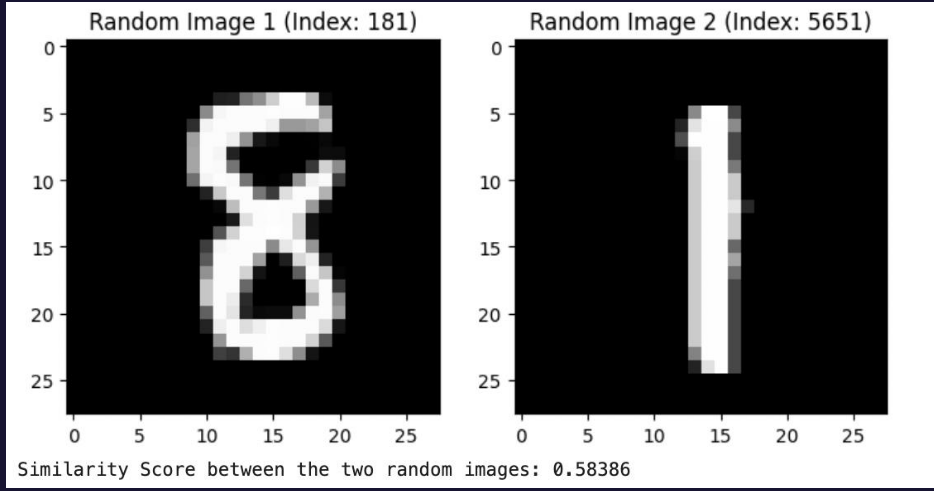


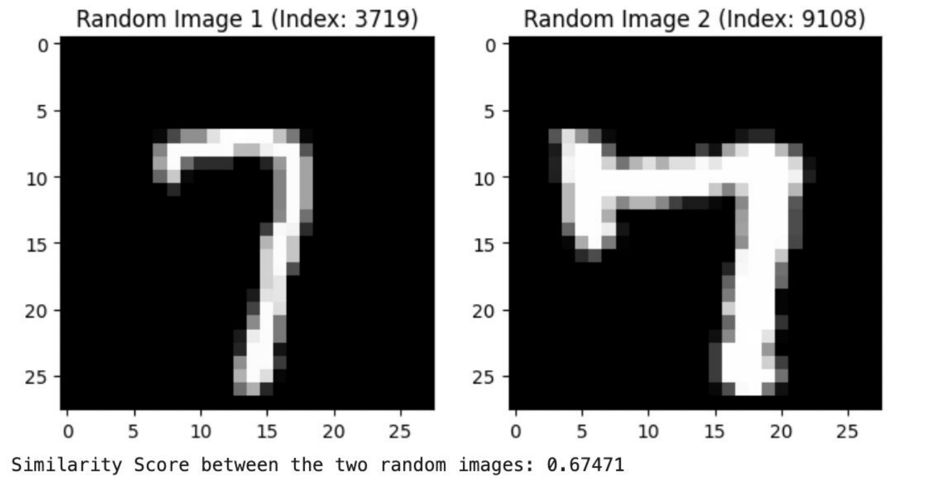




Results


Random Image 1 (Index: 1365)
Random Image 2 (Index: 3296)
Similarity Score between the two random images: 0.62712


Random Image 1 (Index: 181)
Random Image 2 (Index: 5651)
Similarity Score between the two random images: 0.58386


Random Image 1 (Index: 3719)
Random Image 2 (Index: 9108)
Similarity Score between the two random images: 0.67471

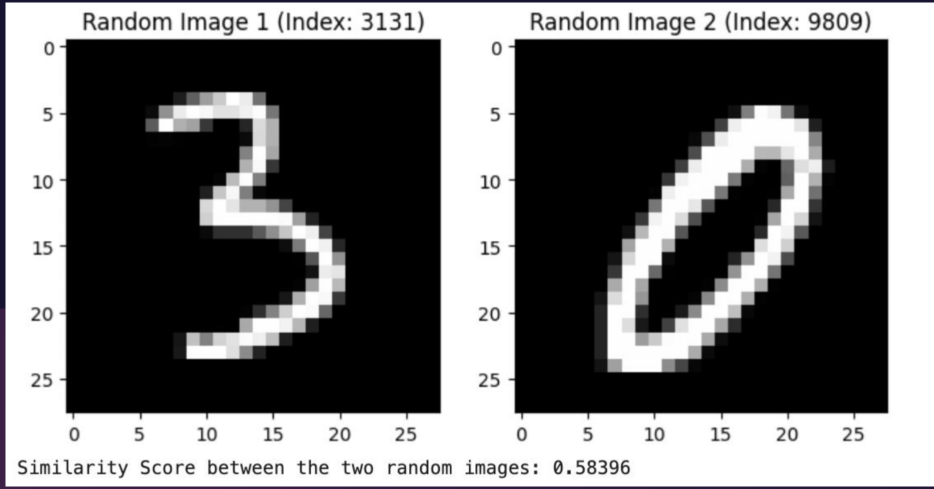
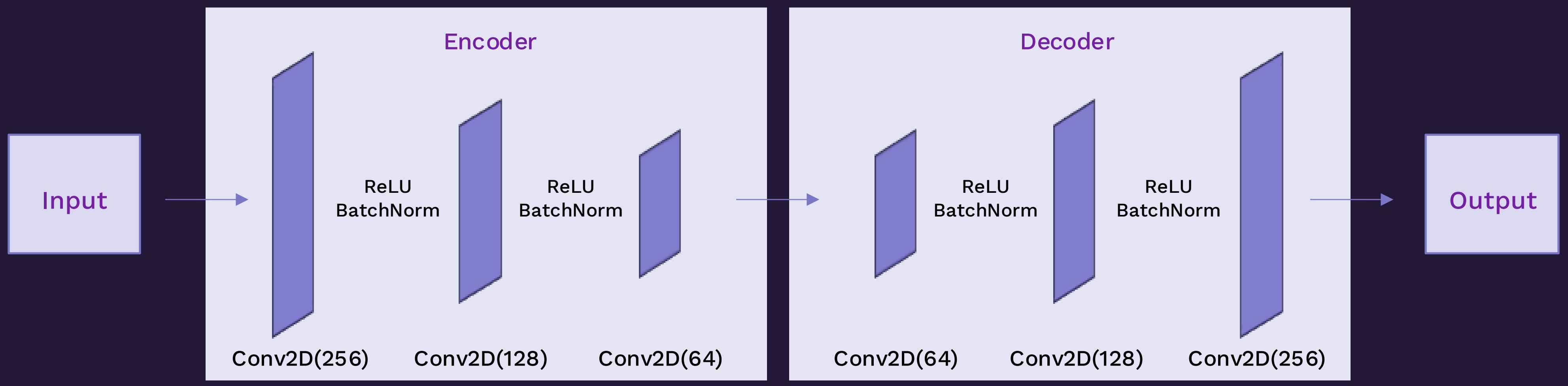

Random Image 1 (Index: 3131)
Random Image 2 (Index: 9809)
Similarity Score between the two random images: 0.58396

Table of contents

Table of contents

Architecture of our Autoencoder



Traditionnal vs. Enhanced Autoencoder

Traditional Autoencoder

- ✗ Uses fully connected layers, which do not preserve spatial structure.
- ✗ Learns compact representations but lacks hierarchical feature extraction.
- ✗ Reconstruction relies on dense layers, leading to higher risk of overfitting.
- ✗ Does not explicitly enforce similarity constraints in the embedding space.
- ✗ Limited performance for clustering and other downstream tasks.

Enhanced Autoencoder

- ✓ Uses convolutional layers, which preserve spatial relationships in images.
- ✓ Extracts hierarchical features, capturing both low- and high-level patterns.
- ✓ Batch normalization stabilizes training and prevents gradient issues.
- ✓ Contrastive loss improves embedding quality by pulling similar samples closer.
- ✓ Better suited for clustering and similarity-based tasks.

First test of the autoencoder

Golden Retriever

German Sherperd

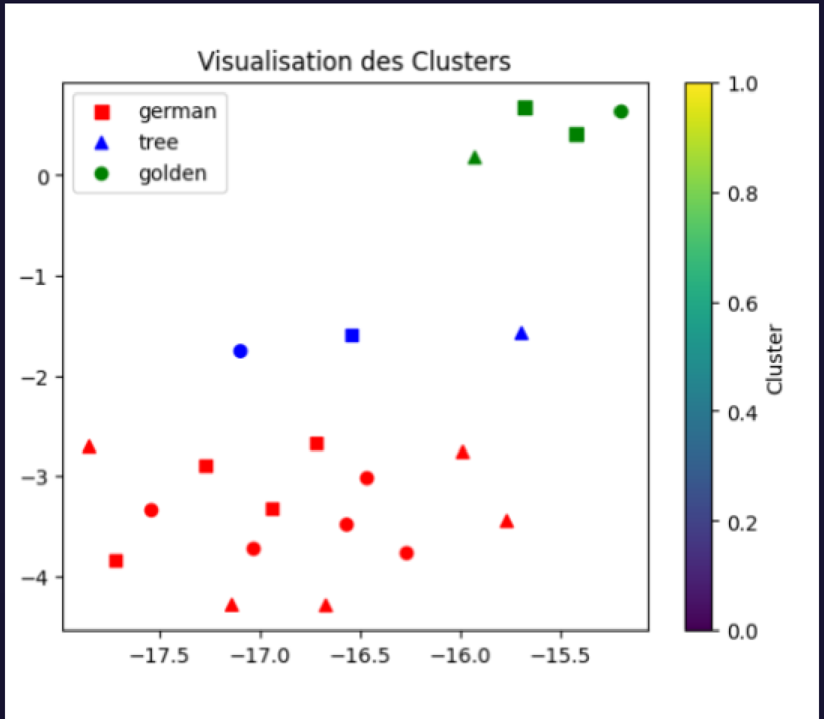
Tree

Plane

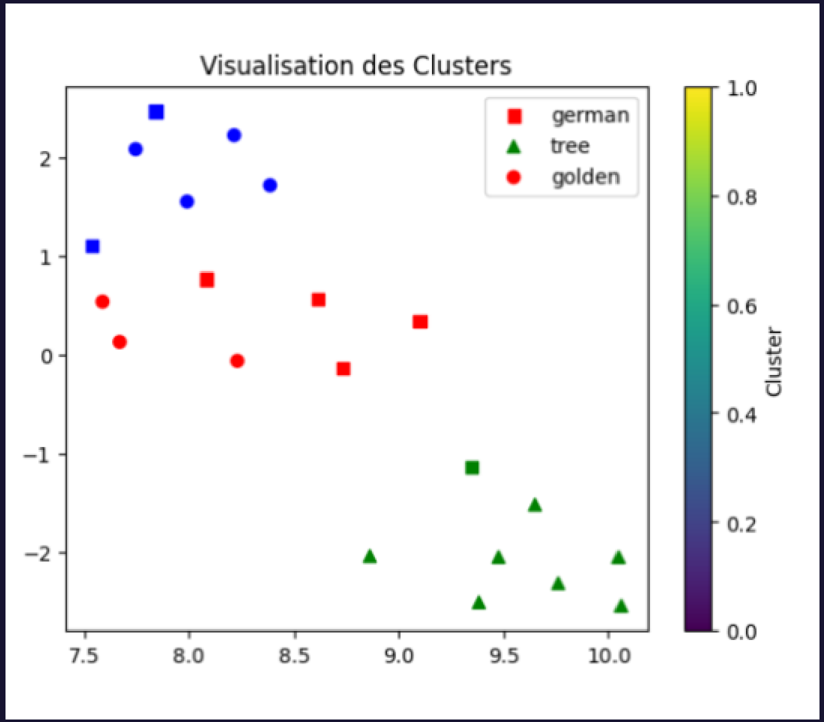
Rose

Traditional vs. Enhanced Autoencoder

Traditional

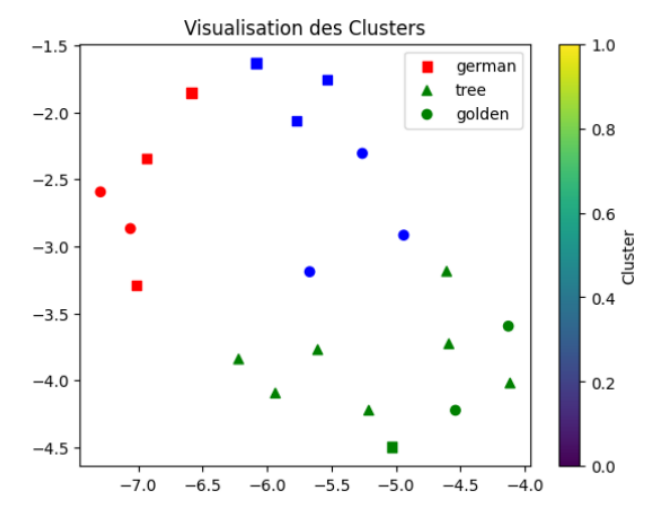


Enhanced

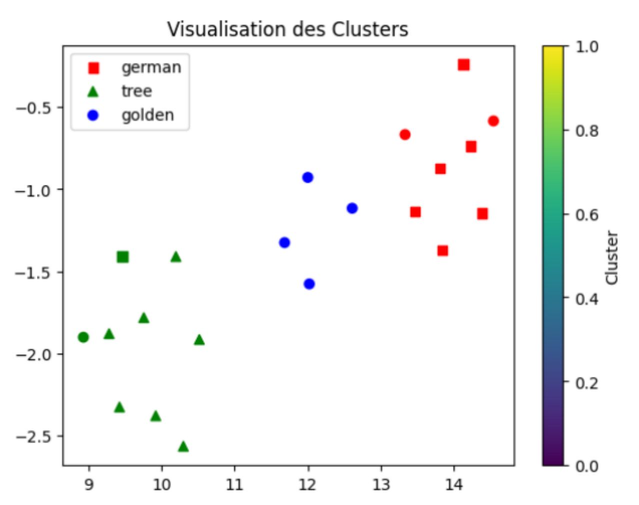


Training and dimensionality reduction

Triplet loss &
MSE Loss



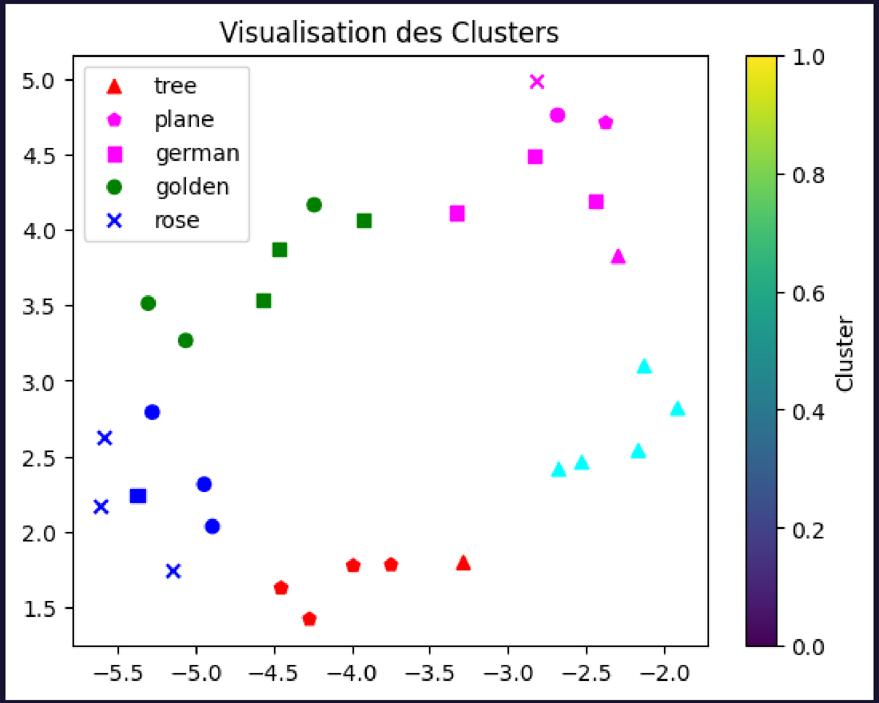
Triplet Loss &
Perceptual Loss



Our final model

- Adam optimizer
- Embeddings normalized with **StandardScaler** and reduced to two dimensions using **UMAP**
- 2 loss functions used during training
  - **Mean Squared Error (MSE) loss :** used for reconstruction
  - **Contrastive loss :** used for better clustering

Clustering and visualization



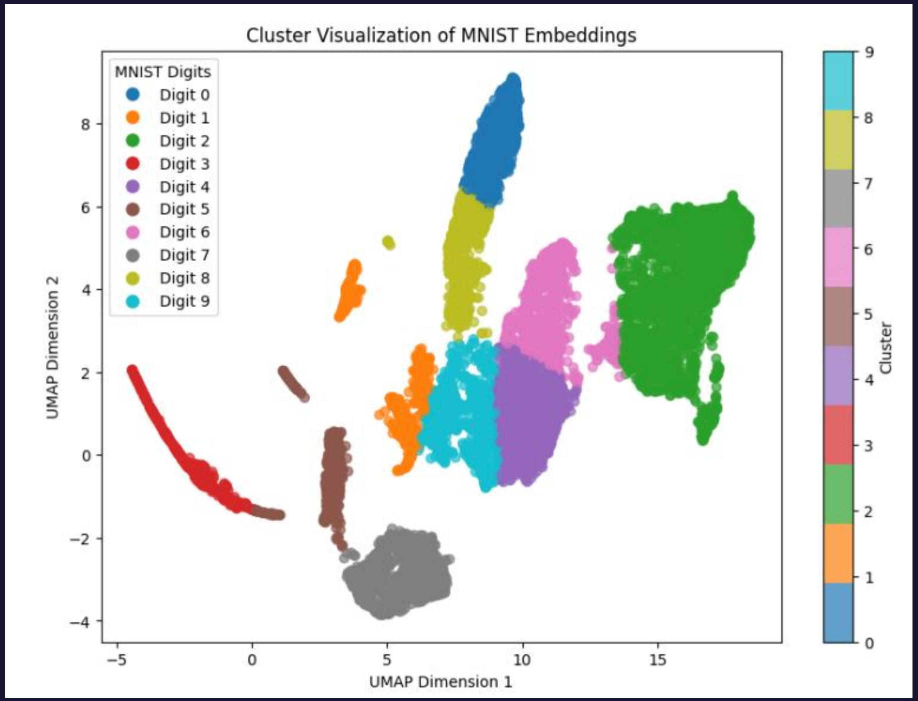


Table of contents

Implementation using a distance matrix

Data preparation

- Textual sequences (DNA sequences)
- Image feature representations, extracted using the Histogram of Oriented Gradients methods

Levenshtein distance

- Dynamic programming
- Handling numeric sequences
- Dynamic cost calculation

Distance matrix

- Distance matrix computed for all combined data points
- Used as an input for the DBSCAN clustering algorithm

Results

- Quadratic time complexity
- Memory constraints

Not working

Implementation using a LSH

What is Locality Sensitive Hashing ?

- Locality-Sensitive Hashing (LSH) enables approximate nearest neighbor searches.
- Hash similar data into "buckets" with high probability, improving scalability.
- Efficiently reduces search space by focusing only on relevant "buckets.“
- Balance between accuracy and computational efficiency via parameter tuning.

How it works

- **Avoid parwise comparisons**
- DNA sequence into **MinHash signature**
- **LSH Binning** for candidate selection
- **Candidate reduction before distance calculation**
- **Parallel processing** of distances
- Clustering using DBSCAN

Results

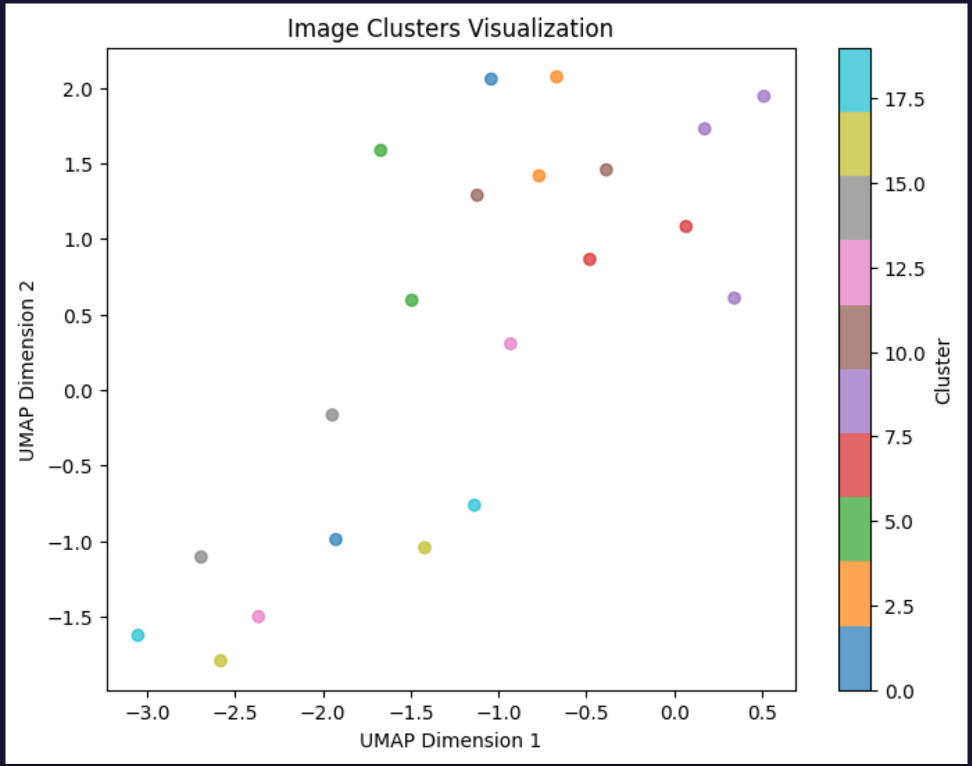


Image Clusters Visualization
UMAP Dimension 1
UMAP Dimension 2
Cluster
−3.0
−2.5
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

Table of contents

Thank you

Questions?