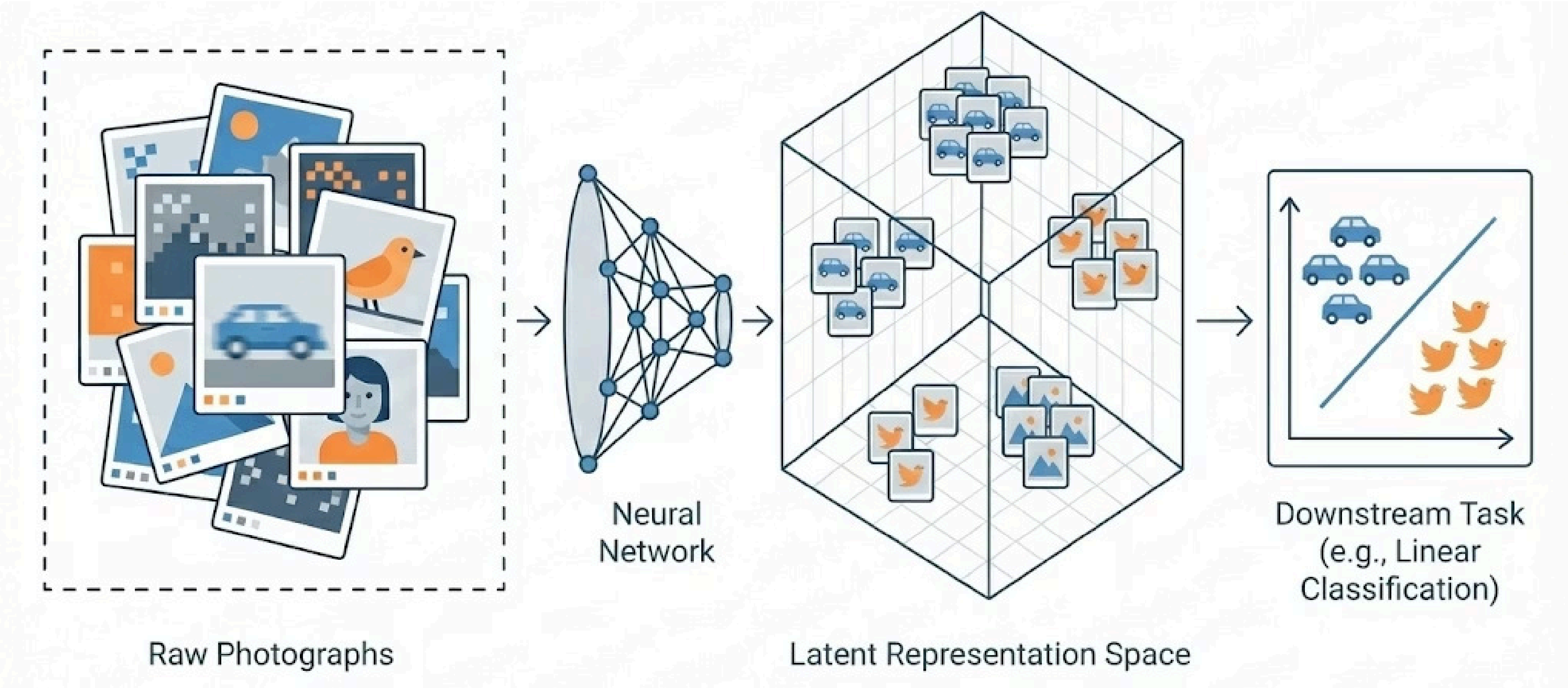
Deep Learning
Advanced Models & Methods

Prof. Antonino Furnari (antonino.furnari@unict.it)
Corso di Laurea Magistrale in Informatica
Dip. di Matematica e Informatica
Università di Catania

Deep Metric Learning

Deep Learning is all About Data Representation

Deep Learning aims to transform high-dimensional raw data, like pixels, into meaningful low-dimensional latent space representations. This process helps capture the essence of the data while **discarding irrelevant details**.



An ideal representation is both **robust** to nuisance factors (e.g., lighting, pose) and **discriminative** of semantic content (e.g., identity, category). Standard classification models (like ResNet with CrossEntropy) implicitly learn such representations as they optimize for classification accuracy.

Distance is All You Need

Measuring distances between data points is fundamental to many AI applications. Deep Metric Learning (DML) addresses the challenge of learning meaningful distance functions that capture semantic relationships rather than just raw data differences.

Retrieval Systems

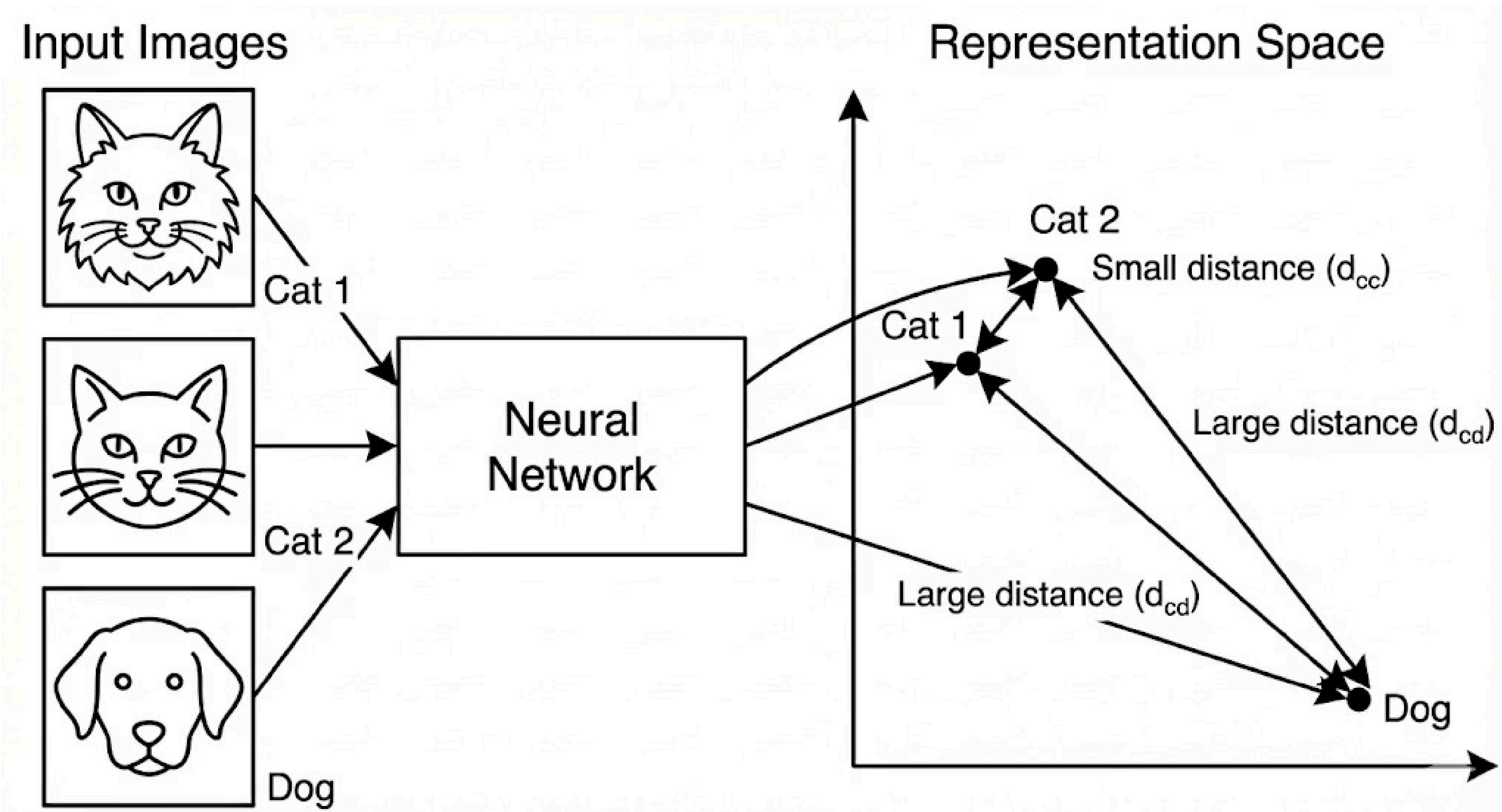
Finding similar items based on queries. Examples include image search engines, document retrieval in RAG systems, and visual product search.

Recommendation Systems

Discovering items similar to user preferences. Examples include finding similar products to recommend, suggesting related content, and collaborative filtering.

Classification

Assigning labels based on proximity to known examples. k-NN classifiers rely entirely on distance metrics to determine class membership.



The Limits of Classification Alone

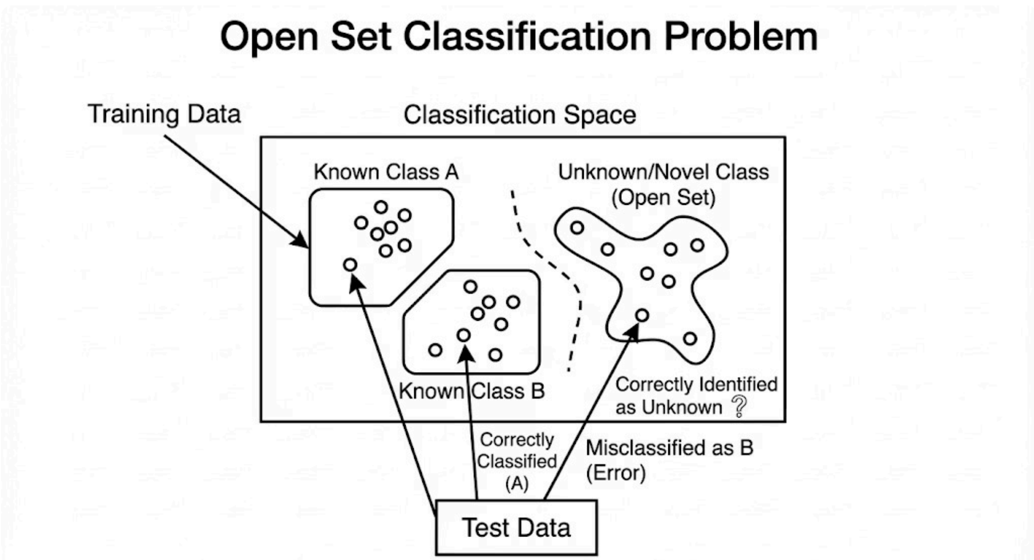
While **classification models learn effective feature representations**, they are optimized for assigning labels within a predefined set. This approach encounters significant limitations in many real-world scenarios where flexibility and nuanced similarity are paramount.

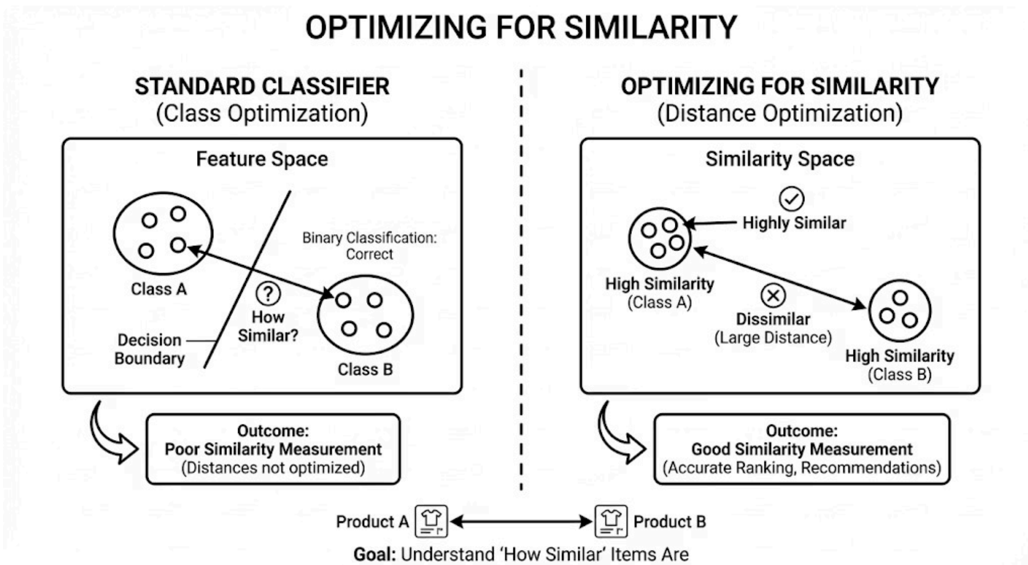
Open-Set & Long-Tail Challenges

Classifiers are inherently "closed-set" systems, struggling with unseen classes or rare examples (long-tail distributions). Updating for new identities is costly, whereas metric learning provides reusable embeddings where new classes can be added dynamically.

Optimizing for Similarity

Many tasks, such as product ranking, personalized recommendations, or anomaly detection, require understanding "how similar" items are, not just "which class" they belong to. Standard classifiers don't directly optimize this crucial similarity objective.





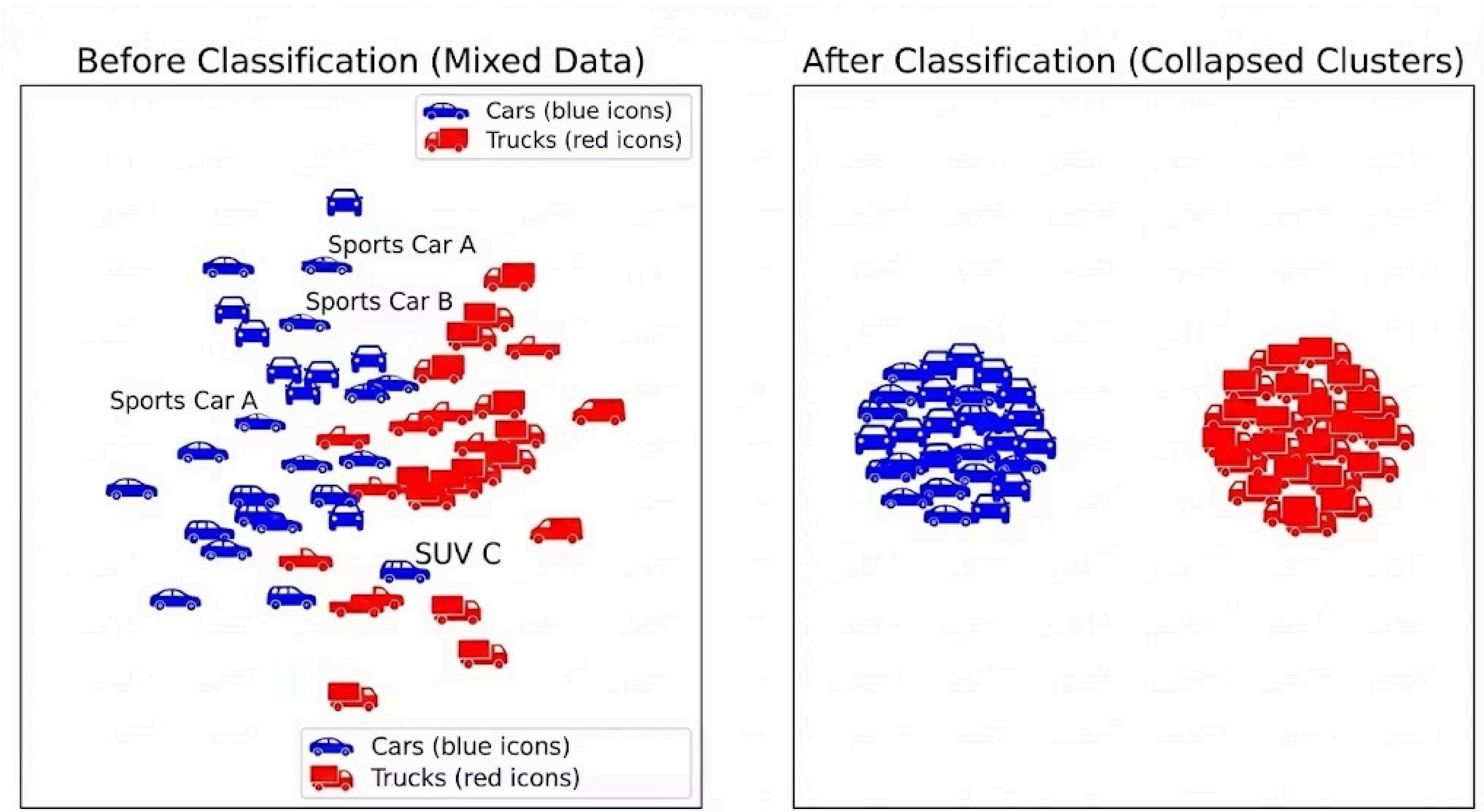
The Problem of Fine-Grained Details

In traditional classification, the primary objective is to maximize separability between distinct classes. The model learns to draw clear boundaries, ensuring that instances from different categories are easily distinguished.

However, this focus on inter-class separation comes with a limitation: **classification does not guarantee that fine-grained details within the same class are preserved or modeled in the latent space.**

The model is optimized to distinguish between classes, but the intricate relationships and subtle differences between elements within the same class may not be captured effectively.

For example, a 'Car' classifier successfully groups all cars together. Yet, it might not preserve the information that 'Sports Car A' is far more similar to 'Sports Car B' than it is to 'SUV C', as these intra-class relationships are not part of the classification objective.



The Data Scarcity Challenge

When Fine-Tuning Fails

One way to recover this specificity is to perform fine-tuning. However, let's consider the following example:

> Imagine needing to retrieve an image that matches a specific query, such as finding a particular dress among many different clothes. This common real-world problem poses a significant challenge for traditional deep learning approaches.

Limited Data

We often have **very few examples** (1 or 2) of the specific instance we want to find. It's very hard to create a "class" for each possible query.

Impractical Fine-Tuning

Not enough data for effective fine-tuning, leading to severe **overfitting**. The computational cost of retraining a classifier for every new query is prohibitive.

This necessitates models that learn **reusable representations**, capable of performing recognition and retrieval without constant retraining for every new item.

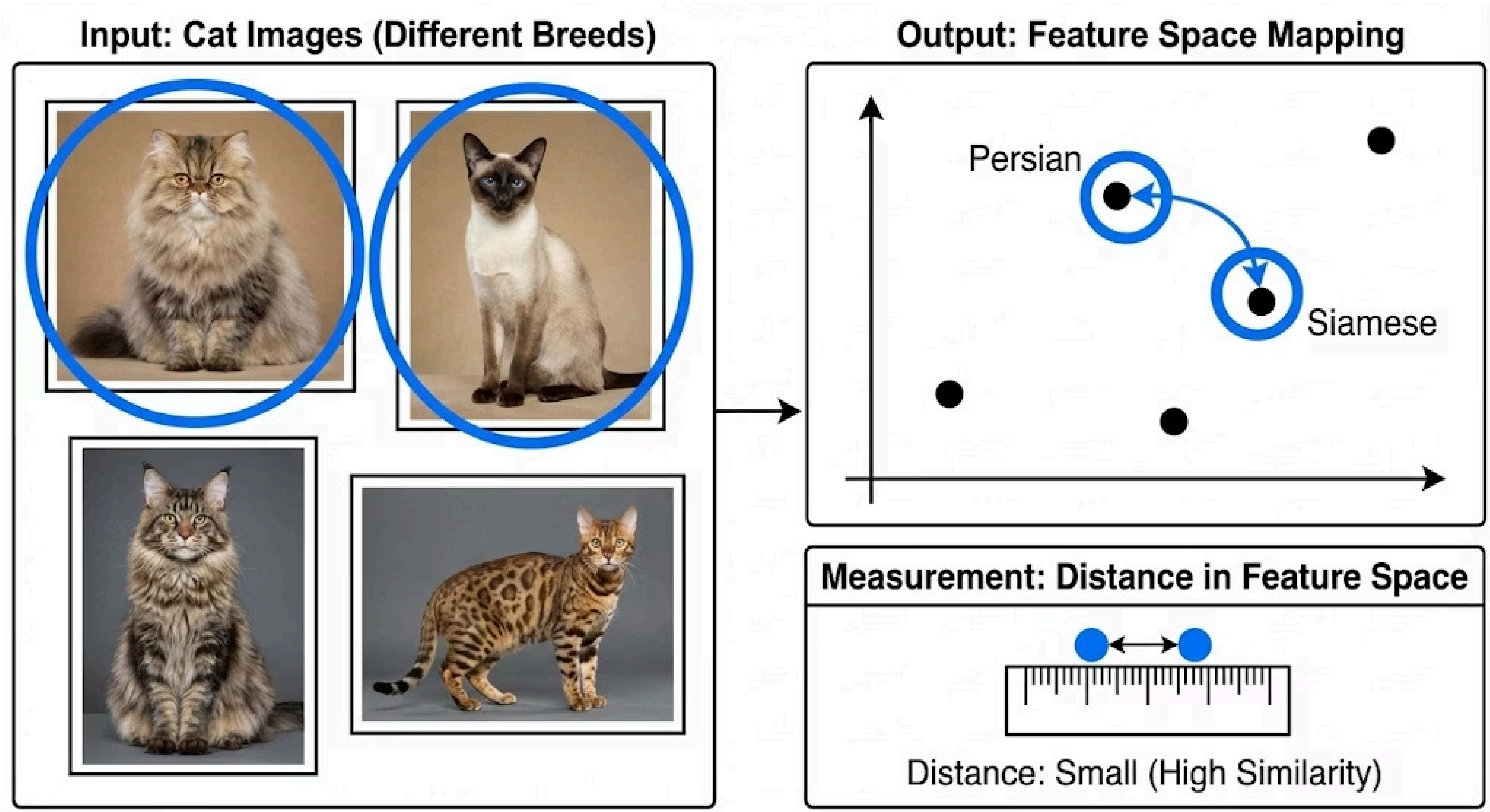
Learning to Measure Distances

The core shift in Deep Metric Learning is from traditional classification ($P(y|x)$) to directly learning to measure the similarity $d(x_1, x_2)$ between data points.

Core Idea: Semantic Similarity in Embedding Space

We train a function $f(x)$ such that distances (e.g., Euclidean) in the resulting embedding space correspond to semantic similarity. **Close embeddings mean similar concepts.**

This fundamental approach allows models to generalize to **Open Sets**—recognizing and relating to classes never encountered during training—by learning the underlying principles of "what makes things similar".



Definition: Deep Metric Learning is the process of optimizing the embedding space directly by using distance-based objectives.

Typical Applications of Metric Learning

Face Recognition

Millions of identities; cannot retrain Softmax for every new user (e.g., FaceID). The system computes embeddings and compares distances rather than retraining weights.

Person Re-Identification

Tracking a subject across non-overlapping camera views. Match embeddings from different perspectives and lighting conditions.

Visual Search

"Find me a shirt that looks like this one." Retrieve visually similar items from massive databases without predefined categories.

The Metric Learning Dictionary

Embedding f(x)

A low-dimensional vector representation of high-dimensional data (e.g., $\mathbb{R}^{3\times224\times224} \to \mathbb{R}^{512}$)

Metric d(x, y)

A function quantifying dissimilarity (usually Euclidean or Cosine distance)

Anchor (x_a)

The reference datum in comparison operations

Positive (x_p)

A datum of the same class as the anchor

Negative (x_n)

A datum of a different class from the anchor

Intra-class Variance

Variability within the same class (minimize this)

Inter-class Variance

Variability between different classes (maximize this)

Historical Evolution: From Classical to Deep

Pre-2012: Classical Methods

Mahalanobis Distance Learning: Learn matrix M such that $d_M(x, y) = \sqrt{(x-y)^T M (x-y)}$

- LMNN (Large Margin Nearest Neighbor)
- Linear transformations of hand-crafted features (SIFT, HOG)
- Limited by linear assumptions

1

2

2014-Present: Deep Metric Learning

Deep CNNs learn the non-linear mapping $f_\theta(x)$

- Distance is usually simple Euclidean/Cosine on top of complex features
- End-to-end optimization
- Breakthrough performance on face recognition and retrieval tasks

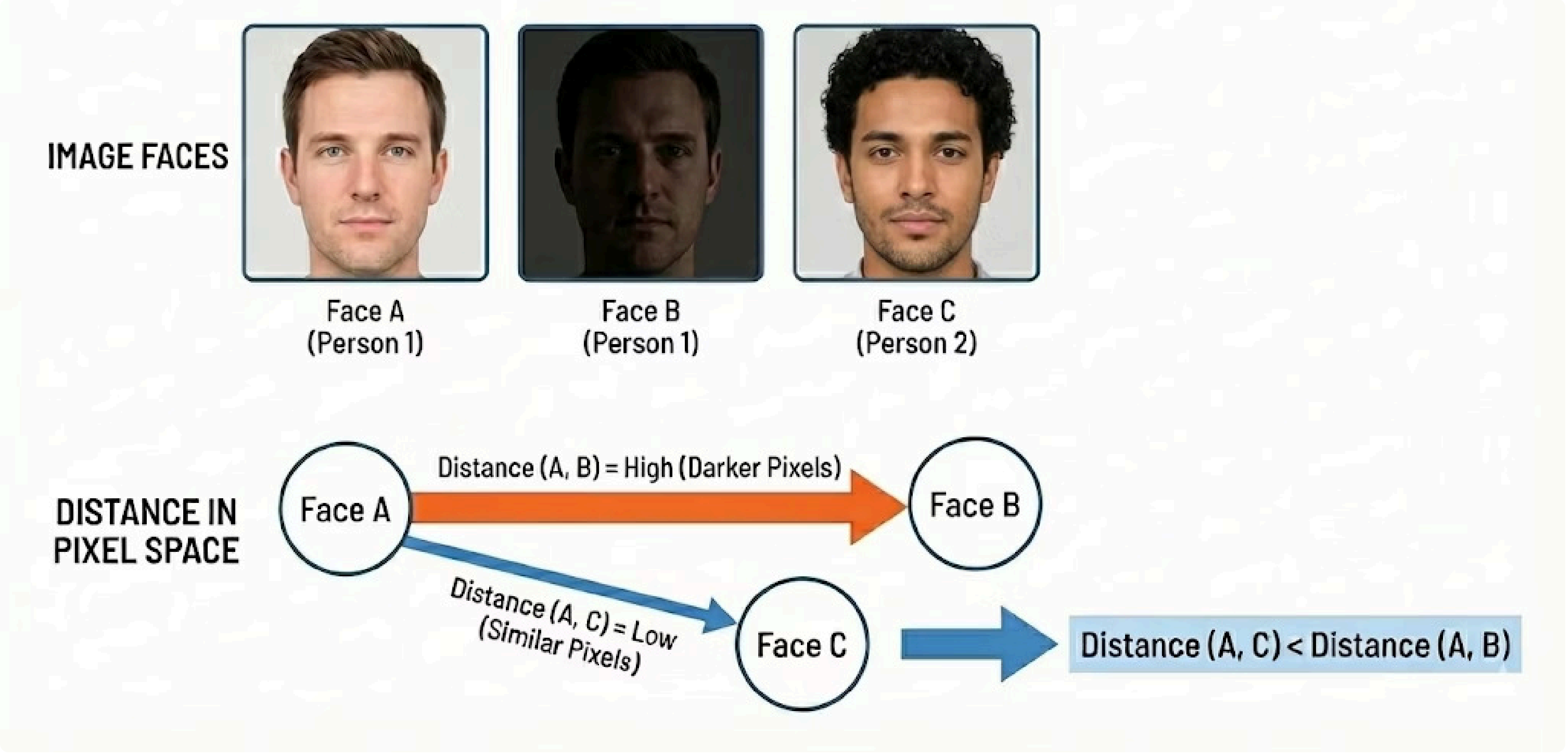
Historically, we hand-engineered features and tried to learn **a linear matrix** M to stretch or shrink the space.

> Today, we assume the distance metric is fixed (usually just Euclidean), and we use a Deep Network to warp the data into a shape that fits that metric. This fundamental shift has enabled unprecedented performance on real-world recognition tasks.

The limits of Pixel Distance

Euclidean distance in pixel space is semantically meaningless in high-dimensional data like images.

Example: L_2 pixel distance between two poses of the same person can exceed the distance between two different people with the same background.

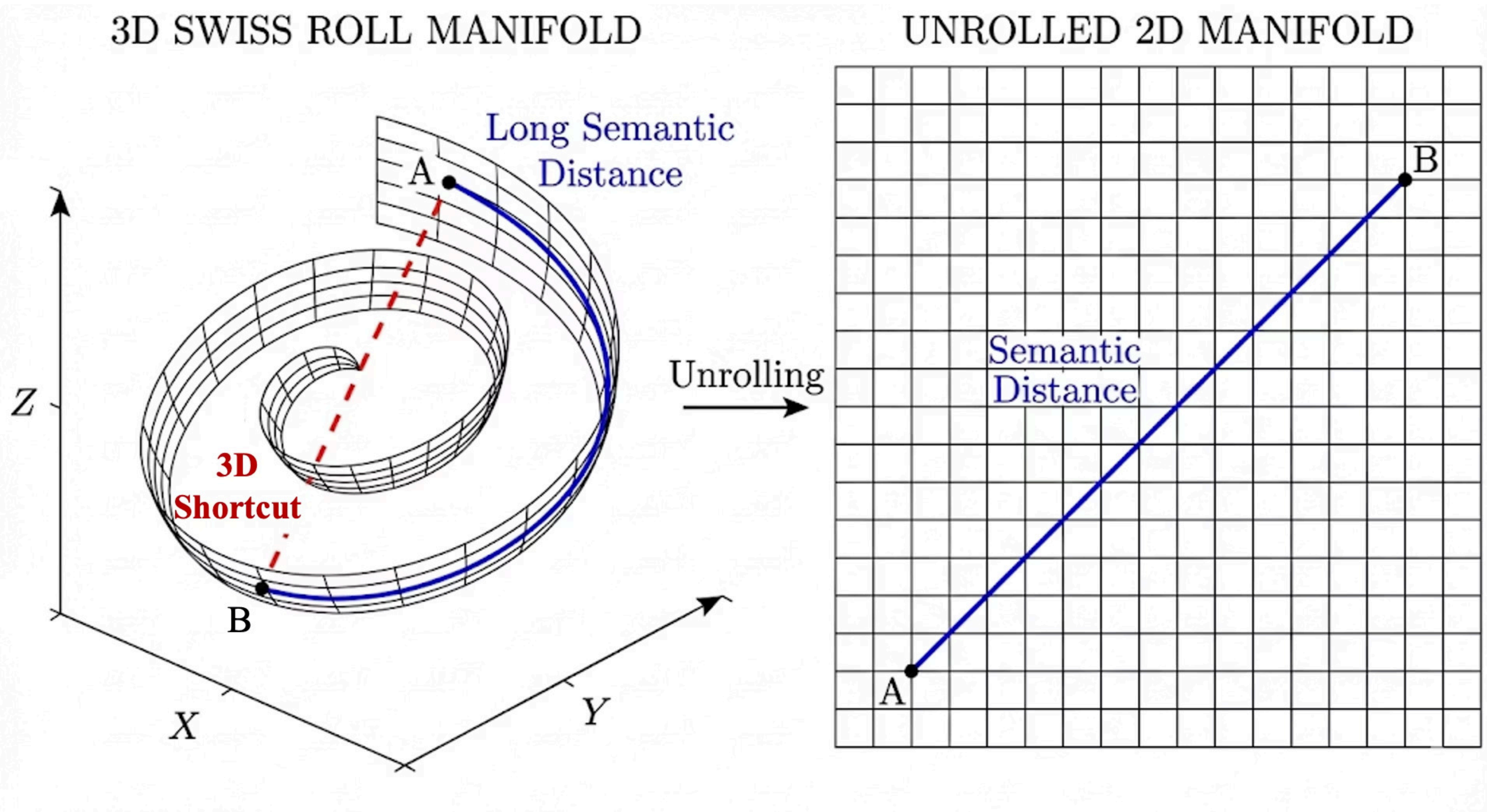


The Manifold Hypothesis

The hypothesis is that high-dimensional data like images lie on low-dimensional manifolds.

Unfold the manifold so that geodesic distance on the manifold becomes Euclidean distance in the embedding space. We need a function that is invariant to lighting and pose but sensitive to identity.

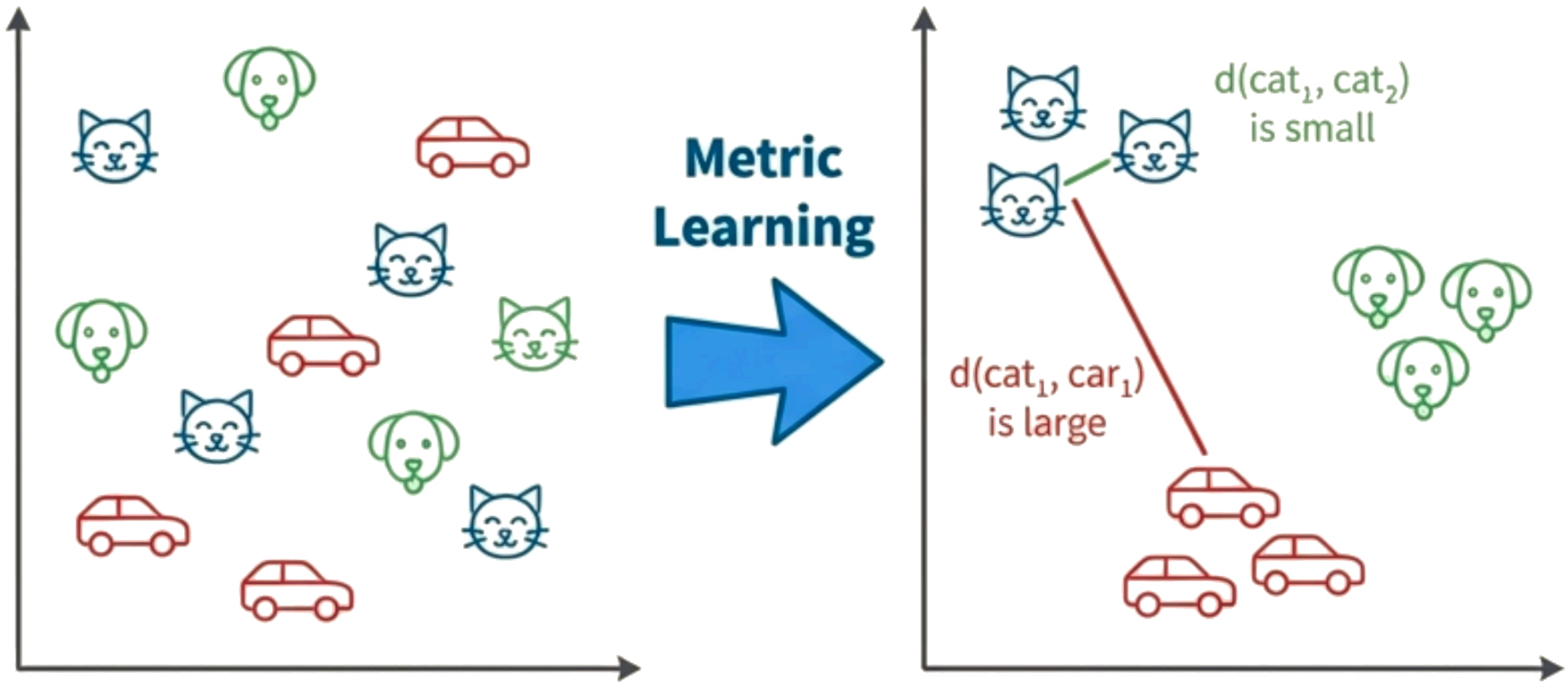
Manifold: a topological space that locally resembles Euclidean space near each point.



This is crucial: Why not just compare pixels? Because pixel distance is sensitive to lighting and pose, not identity. The manifold hypothesis posits that natural images lie on a low-dimensional manifold embedded in high-dimensional pixel space. Deep metric learning learns to unfold this manifold.

Re-arranging the Embedding Space

More in general, we can see metric learning as the problem of transforming raw input data into a meaningful embedding space.



In this learned space, semantically similar items, such as different images of the same object or category, are clustered closely together. Conversely, dissimilar items are pushed farther apart. The distance between points in this embedding space directly corresponds to their semantic similarity, allowing for effective classification and recognition of novel data points even if they haven't been seen during training.

Mathematical Setup

Formal Problem Definition

- **Input Space:** $\mathcal{X} \subset \mathbb{R}^D$
- **Pairwise Labels:**

$$l_{i,j} = \begin{cases} 1 & \text{if } x_i \text{ is similar to } x_j \\ 0 & \text{otherwise} \end{cases}$$

- **Embedding Function:** $f_\theta : \mathcal{X} \to \mathbb{R}^d$ where $d \ll D$
- **Distance Metric:** $D(x_i, x_j) = d(f_\theta(x_i) - f_\theta(x_j))$

Where d is a distance measure (more on this in the next slide).

Objective

if $l_{ij} = 1 : D(x_i, x_j) \to 0$

if $l_{ij} = 0 : D(x_i, x_j) > m$

Where m is a margin providing a safety zone.

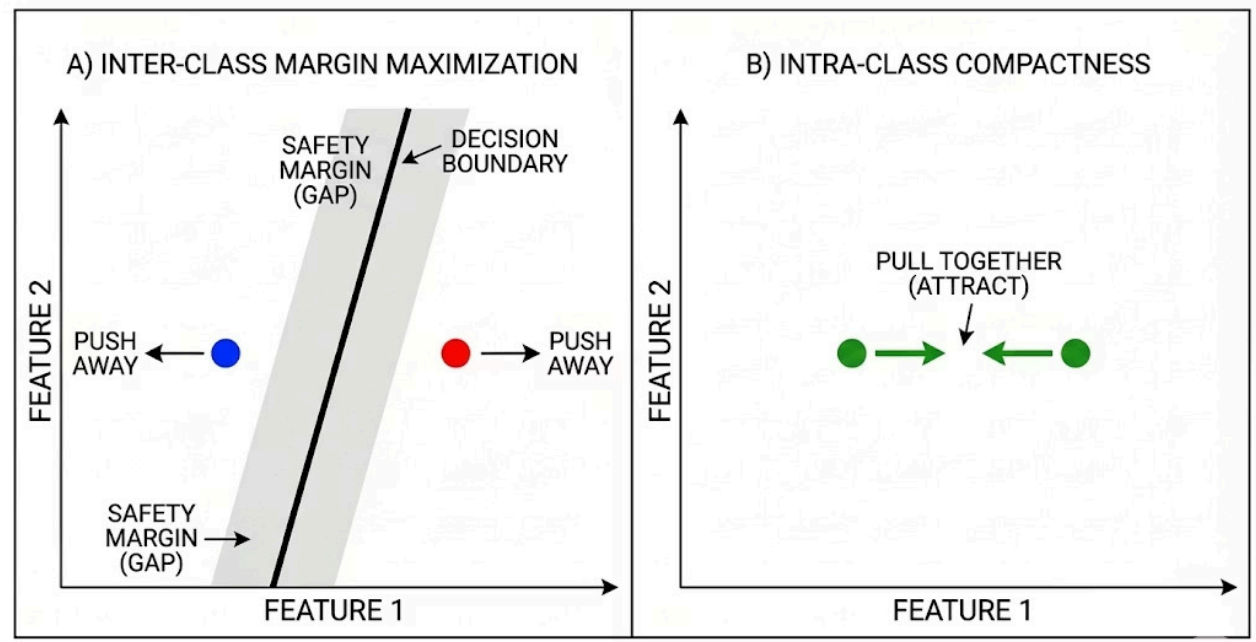


Figure X. Two Key Concepts in Metric Learning: Margin Maximization and Intra-class Compactness

Classes to Pairwise Labels

Often, we will consider elements x_i to be associated to a class label $y_i \in \mathcal{Y} = \{0, \ldots, M\}$. In that case, we can still define $l_{ij} = \mathbf{1}(y_i = y_j)$, where $\mathbf{1}(\cdot)$ is the indicator function.

> Note the dimension d. Usually, this is 128, 512, or 2048. It needs to be compact for efficient storage and retrieval. The objective defines a "safety zone" (margin) between the distributions of positive and negative pairs, ensuring robust classification boundaries.

Choosing the Ruler: Distance Metrics

In metric learning, the choice of distance function is paramount. While many types of distance metrics can be utilized, two stand out for their widespread use and distinct properties in various applications.

Euclidean Distance (L_2)

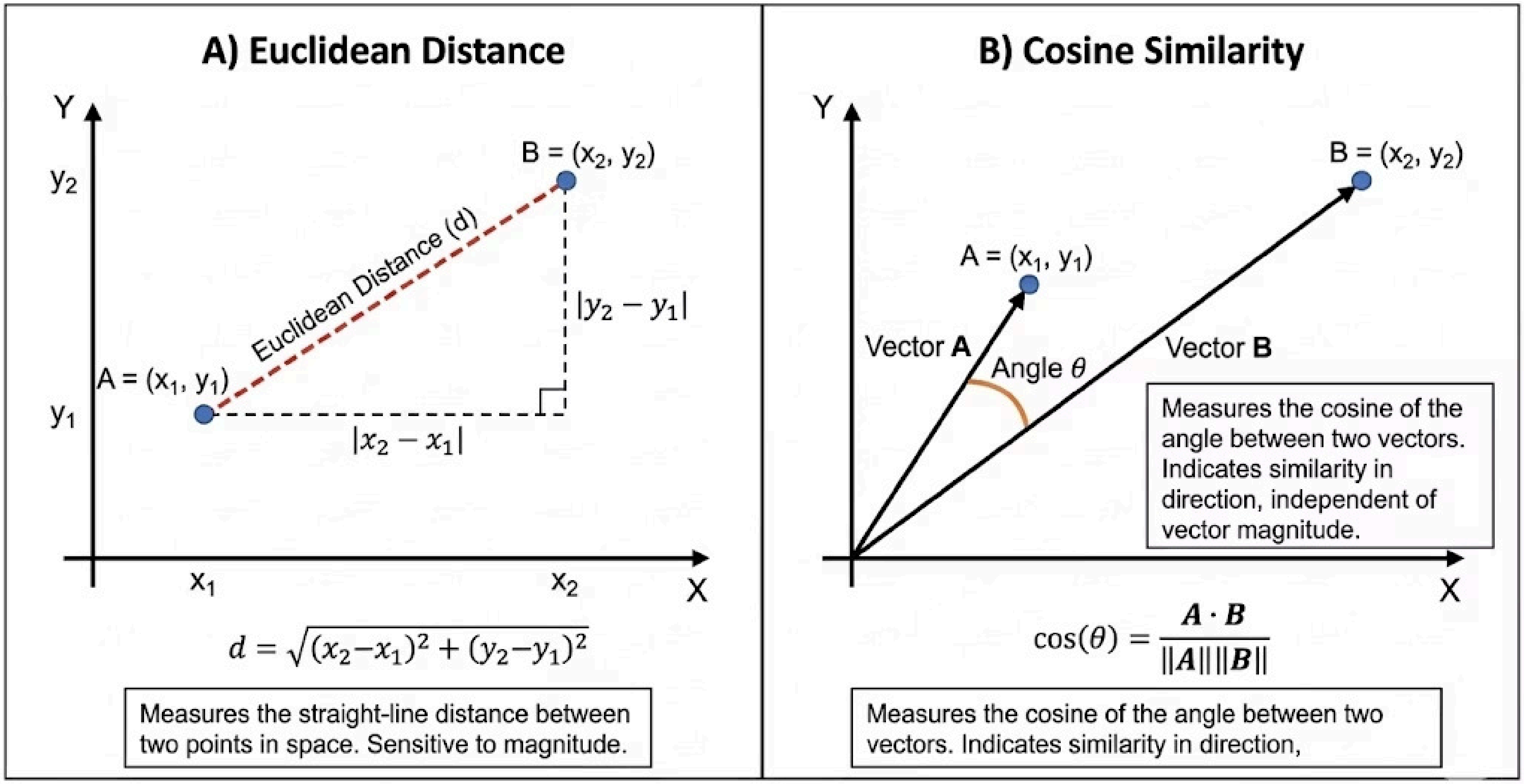
$$d(u,v) = \|u - v\|_2 = \sqrt{\sum_{i=1}^{n}(u_i - v_i)^2}$$

This is the standard, intuitive "straight-line" distance between two points in an N-dimensional space. It's sensitive to both the direction and magnitude of the embedding vectors. Ideal when the absolute difference in feature values is meaningful.

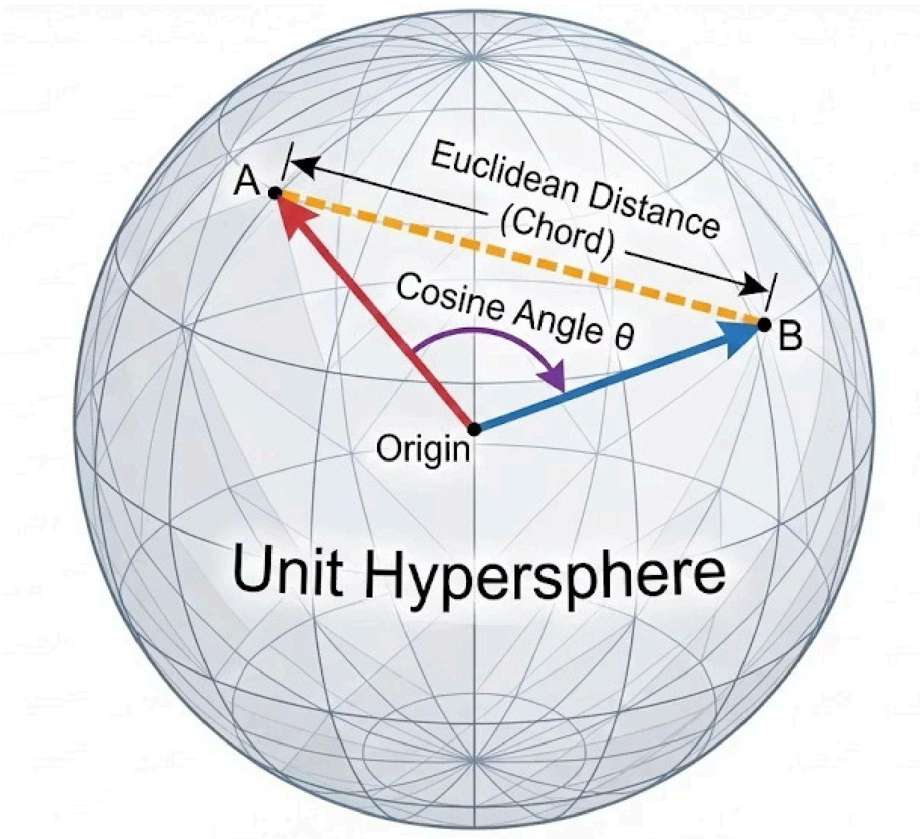
Cosine Similarity

$$S(u,v) = \frac{u \cdot v}{\|u\|\|v\|}$$

Measures the cosine of the angle between two vectors, ranging from -1 (opposite) to 1 (identical direction). The actual distance is often represented as $d_{cos} = 1 - S(u,v)$. Crucially, it ignores vector magnitude and focuses solely on their orientation. This makes it particularly effective for **Face Recognition** and other tasks where feature magnitude might vary due to lighting or expression, but the underlying identity's "direction" in the embedding space remains consistent.



The Geometry of High Dimensions



Hypersphere Embedding

Constraining $\|f(x)\|_2 = 1$ (unit radius) is a powerful technique primarily used to:

- **Regularize the learning process** and prevent exploding values within the network.
- **Avoid a "race of magnitude"** during training, ensuring the network focuses on learning meaningful angular relationships instead of indefinitely increasing embedding magnitudes.
- **Focus optimization on angular distances**, enabling more robust comparisons between embeddings.

This approach transforms the embedding space, offering further benefits:

- The effective "space" for comparison becomes the surface of the manifold.
- It enables the use of interpretable angular distances for similarity.

When we use "**Cosine Similarity**", we are effectively projecting everything onto a hypersphere.

Why Do We Need a Margin?

Separability Is Not Enough

The bare minimum requirement is $d(A, P) < d(A, N)$, but this is insufficient for robust performance.

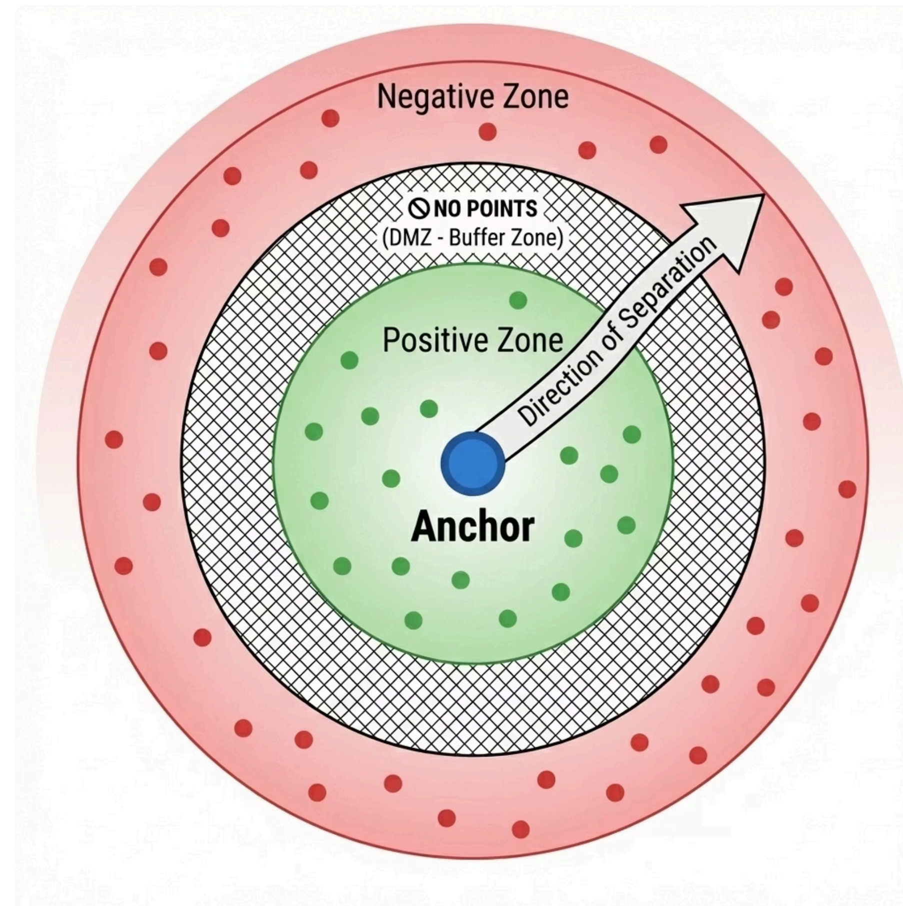
Problem: If $d(A, P) = 0.5$ and $d(A, N) = 0.50001$, the loss is zero, but the decision is highly unstable and sensitive to noise.

The Margin (m)

We enforce $d(A, P) + m < d(A, N)$

This forces the model to push negatives *further* away than strictly necessary, improving generalization to unseen data.

> Think of the margin as an insurance policy against noise in the test set. By requiring a buffer zone between positive and negative pairs, we ensure that small perturbations or variations in test conditions don't cause misclassifications. The margin is a hyperparameter that trades off between tighter clusters and more stable boundaries.



Siamese Networks

Architecture

Two identical subnetworks (clones) with **Shared Weights**: W is the same for both branches.

- **Input:** Pair (x_1, x_2)
- **Output:** Distance $d(x_1, x_2) = D(f(x_1) - f(x_2))$

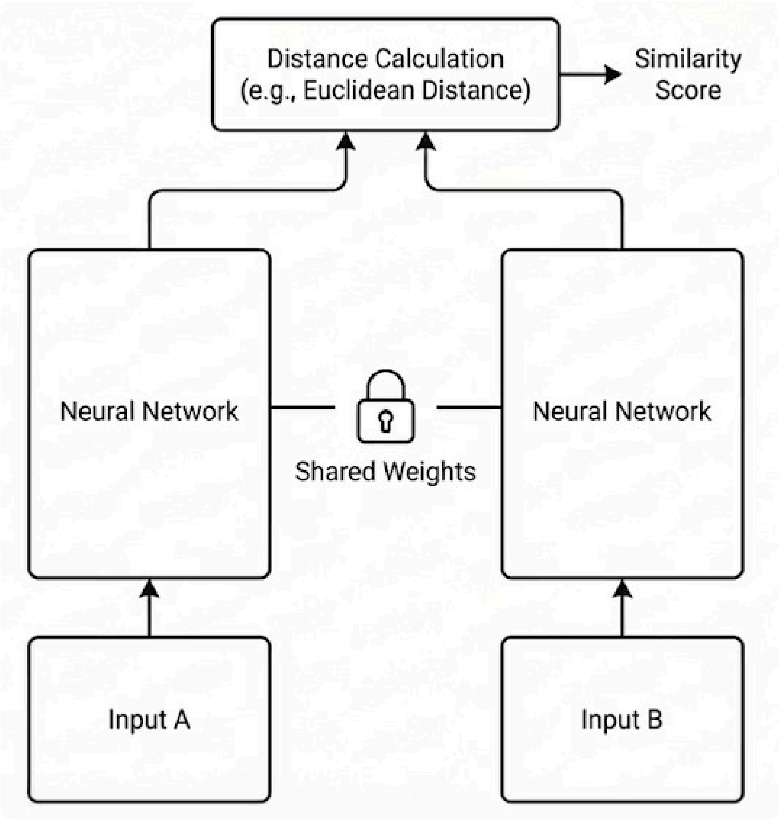
Where D is usually the Euclidean or cosine distance.

Expected Behavior

The siamese network takes as input two elements and returns a real number. We expect this number to be:

- `small` if the two elements are similar
- `large` if the two elements are dissimilar

If this happens, we can use the siamese network as as **distance function** and, importantly, the neural network f as a **feature extractor**.



Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., & Shah, R. (1993). Signature verification using a" siamese" time delay neural network. *Advances in neural information processing systems, 6.*

Training a Siamese Network

Training Strategy

- Minimize distance if Label=1 (Same class)
- Maximize distance if Label=0 (Different class)

This is also called contrastive learning.

Loss Function

In practice, we use the following contrastive Loss formula (Hadsell et al., 2006):

$$L(x_i, x_j, l_{ij}) = \frac{1}{2}(1 - l_{ij})d(x_i, x_j)^2 + \frac{1}{2}l_{ij} \max[0, m - d(x_i, x_j)]^2$$

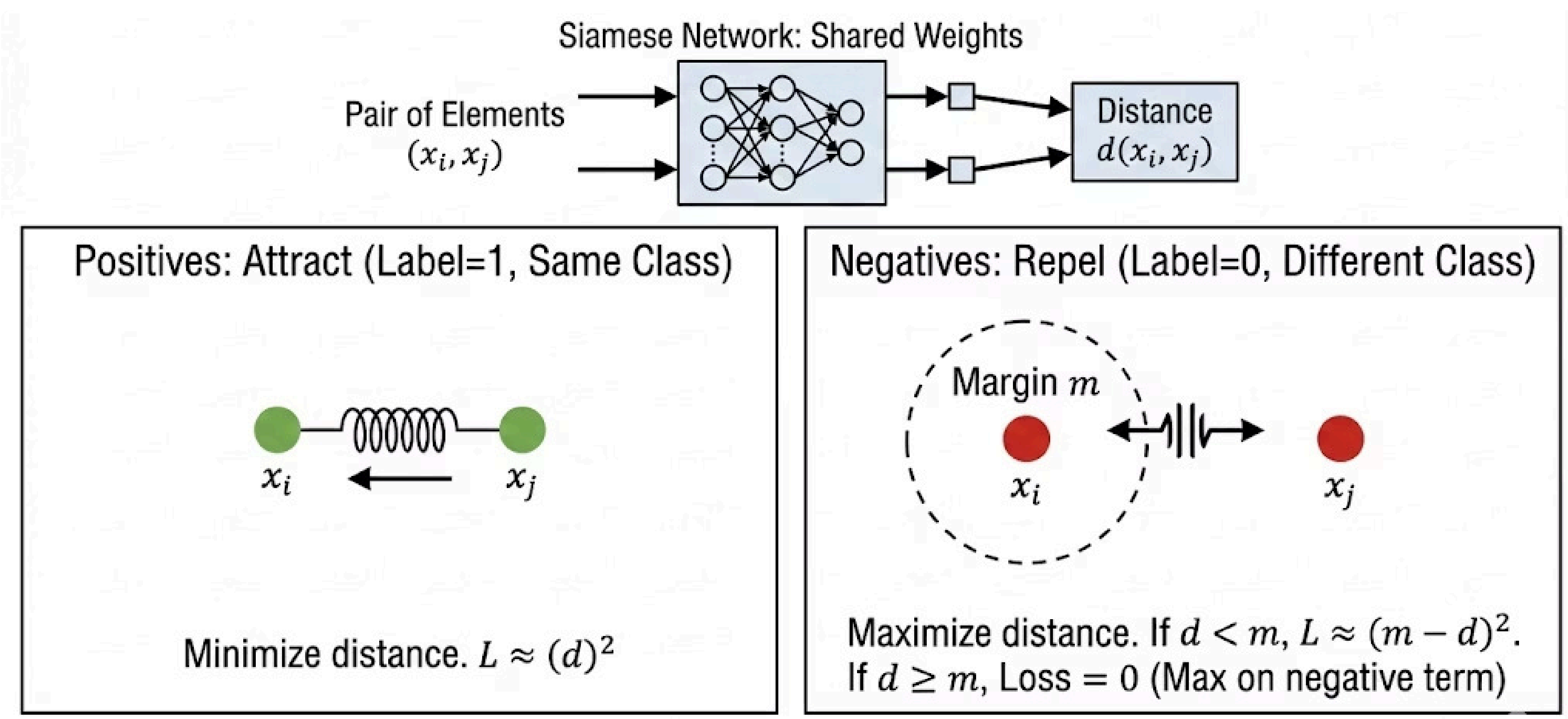
Where:

- l_{ij} = 0 for similar pairs (same class)
- l_{ij} = 1 for dissimilar pairs (different classes)
- m is the margin parameter

Dynamics

- **Positives** ($y = 1$): Attract like a spring. Quadratic penalty for any distance.
- **Negatives** ($y = 0$): Repel, but only if they are within margin m. If they are far enough, loss is 0.

> Note the max on the negative term. Why? If a negative is already far away, we don't care. We don't want to spend computational energy pushing "sky" away from "grass" if they are already clearly distinct. We focus gradient updates on the confusing pairs near the margin boundary.



Loss Function Dynamics: Attract positives, Repel negatives beyond margin.

Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., & Shah, R. (1993). Signature verification using a" siamese" time delay neural network. *Advances in neural information processing systems*, *6*.

Hadsell, Raia, Sumit Chopra, and Yann LeCun. "Dimensionality reduction by learning an invariant mapping." In *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*, vol. 2, pp. 1735-1742. IEEE, 2006.

The Feature Extractor: Architecture Backbones

ResNet-50 / ResNet-101

Standard choice for most DML applications. Reliable performance with reasonable computational cost.

Inception-BN

Used in original FaceNet. Multi-scale feature extraction through parallel convolutions.

MobileNet

For wearable and mobile applications. Efficient depthwise separable convolutions.

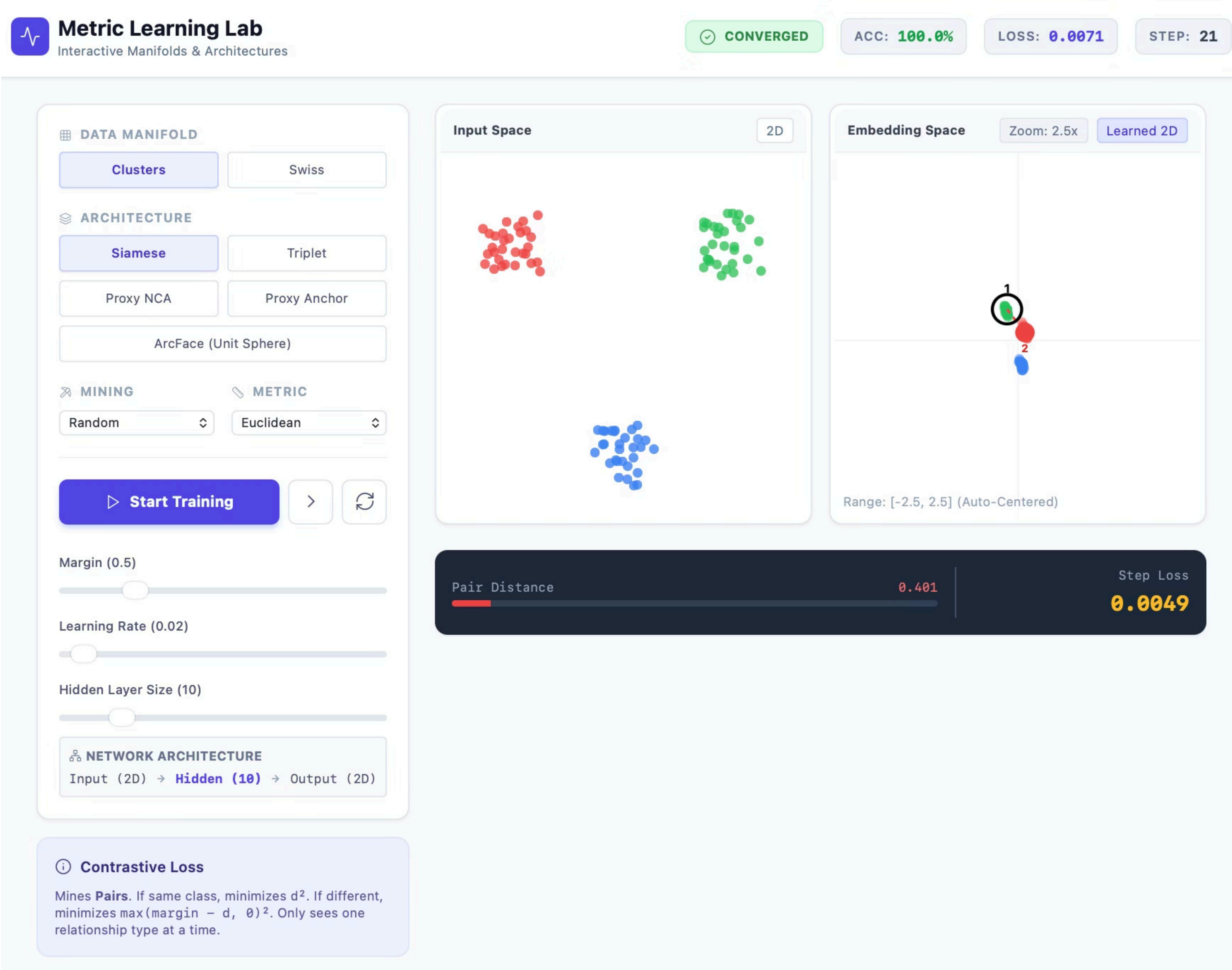
Vision Transformers (ViT)

Current trend. Captures global context better than CNNs through self-attention mechanisms.

> The DML "Head" sits on top of a standard backbone. For aggregation, we typically use Global Average Pooling (GAP) or Global Max Pooling (GMP). Backbones are often **pretrained on supervised datasets** (e.g., Imagenet).

Siamese Network Demo

Metric Learning Demo



Shortcomings of Siamese Networks

While Siamese Networks laid crucial groundwork for metric learning, their pairwise training approach presents several challenges that highlighted the need for more sophisticated architectures and loss functions.

Pairwise Training Bottleneck

Siamese networks consider only pairs of examples. This **N-squared complexity** leads to slow convergence and doesn't efficiently capture the global structure of the embedding space.

Absolute Margin, Not Relative

The contrastive loss enforces an **absolute margin** between dissimilar pairs. It doesn't explicitly ensure that an anchor is closer to its positive than to a specific negative, making decision boundaries less robust.

Inefficient Negative Sampling

Many randomly chosen negative pairs are "easy" and quickly satisfy the margin constraint, providing **little to no learning signal**. Training can become inefficient, focusing on already-separated samples.

Triplet Networks

Architecture

Three identical branches with Shared Weights.

Input: Triplet (x_a, x_p, x_n)

Output: Pairwise distances $d(x_a, x_p)$ and $d(x_a, x_n)$

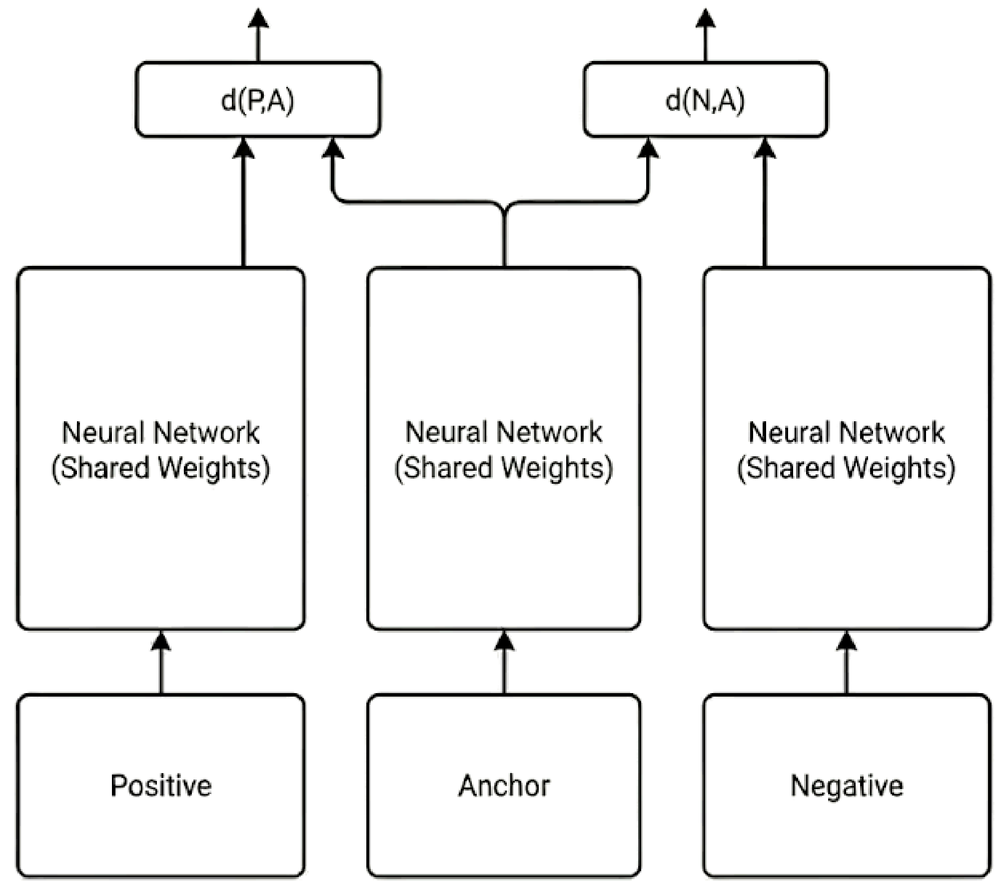
Expected behavior

For any triplet, we expect the distance between the **anchor (x_a)** and the **positive (x_p)** to be smaller than the distance between the **anchor** and the **negative (x_n)** by at least m :

$$d(x_a, x_p) + m < d(x_a, x_n)$$

Core Logic

Contextual comparison: the anchor here serves as a sort of context. We don't care about absolute distances, only relative distances $d(x_a, x_p) < d(x_a, x_n)$.



Schroff, F., Kalenichenko, D., & Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 815-823).

Triplet Loss

The Triplet network is trained using a Triplet loss function, defined as follows:

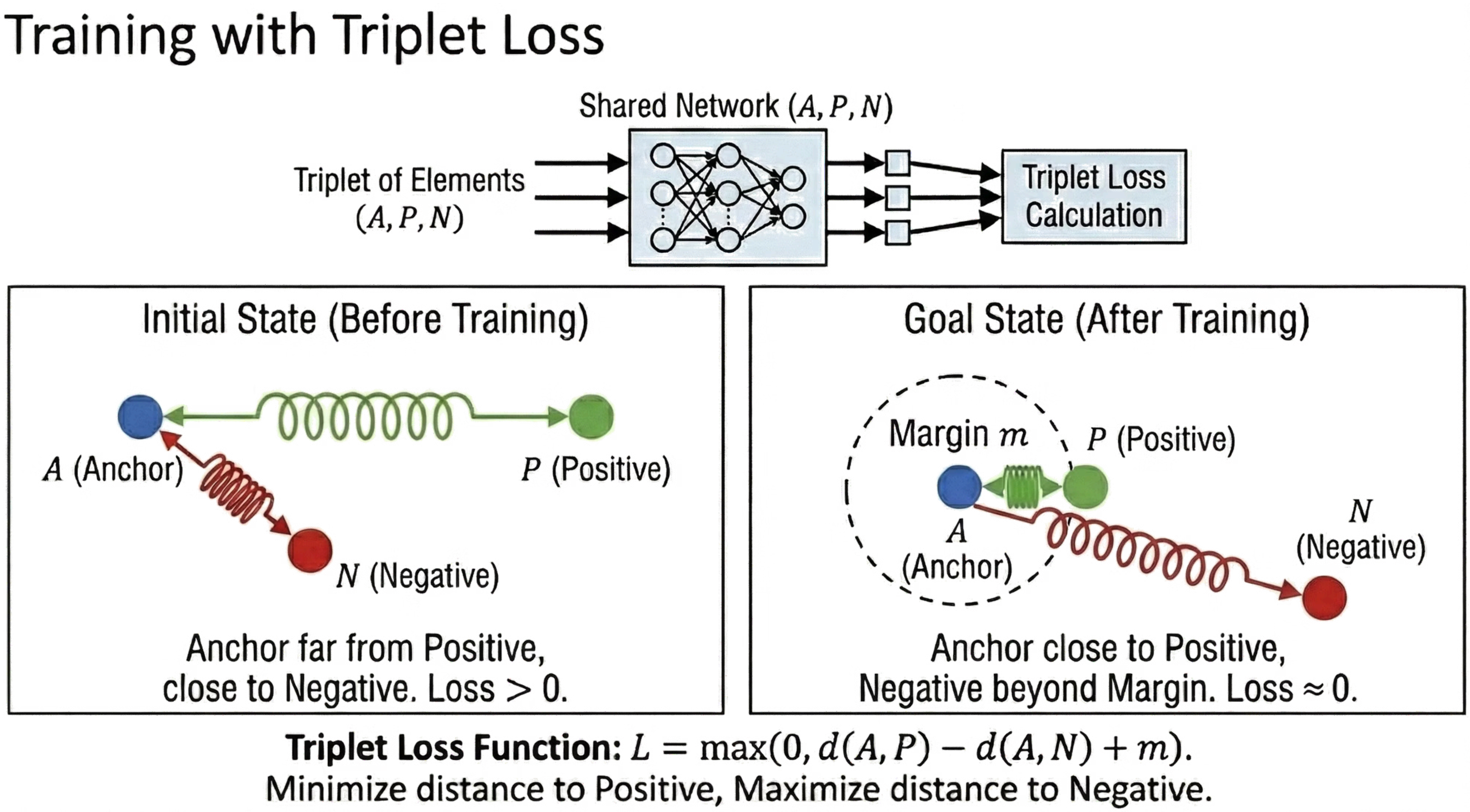
Input: Triplet (x_a, x_p, x_n)

$$L = \max(0, d(x_a, x_p) - d(x_a, x_n) + m)$$

Objective: Ensure the anchor is closer to the positive than to the negative by at least a margin m.

$$d(x_a, x_p) + m < d(x_a, x_n)$$

Often, we choose d as the Euclidean distance, but other distance measures are possible.



> **Issue:** The number of possible triplets is cubic $O(N^3)$. Most triplets satisfy the constraint easily (loss=0) and contribute no gradient, leading to extremely slow convergence without proper mining strategies.

Schroff, F., Kalenichenko, D., & Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 815-823).

The Importance of Mining

Random Sampling

Approximately 99% of triplets produce 0 loss (gradients vanish). Convergence is extremely slow and inefficient. The network learns nothing from easy triplets.

Hard Negative Mining

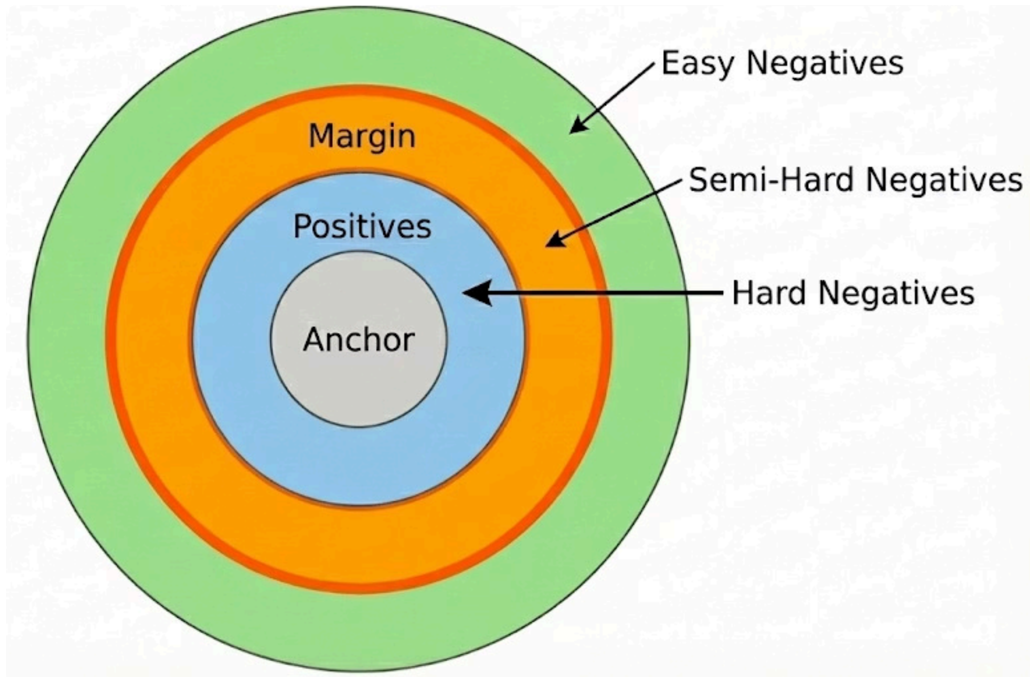
Find x_n such that $d(a,n)$ is smallest.

Risk: Model collapse into bad local minima, noise amplification from outliers and mislabeled data.

Semi-Hard Mining (FaceNet)

Find x_n where $d(a,p) < d(a,n) < d(a,p) + m$

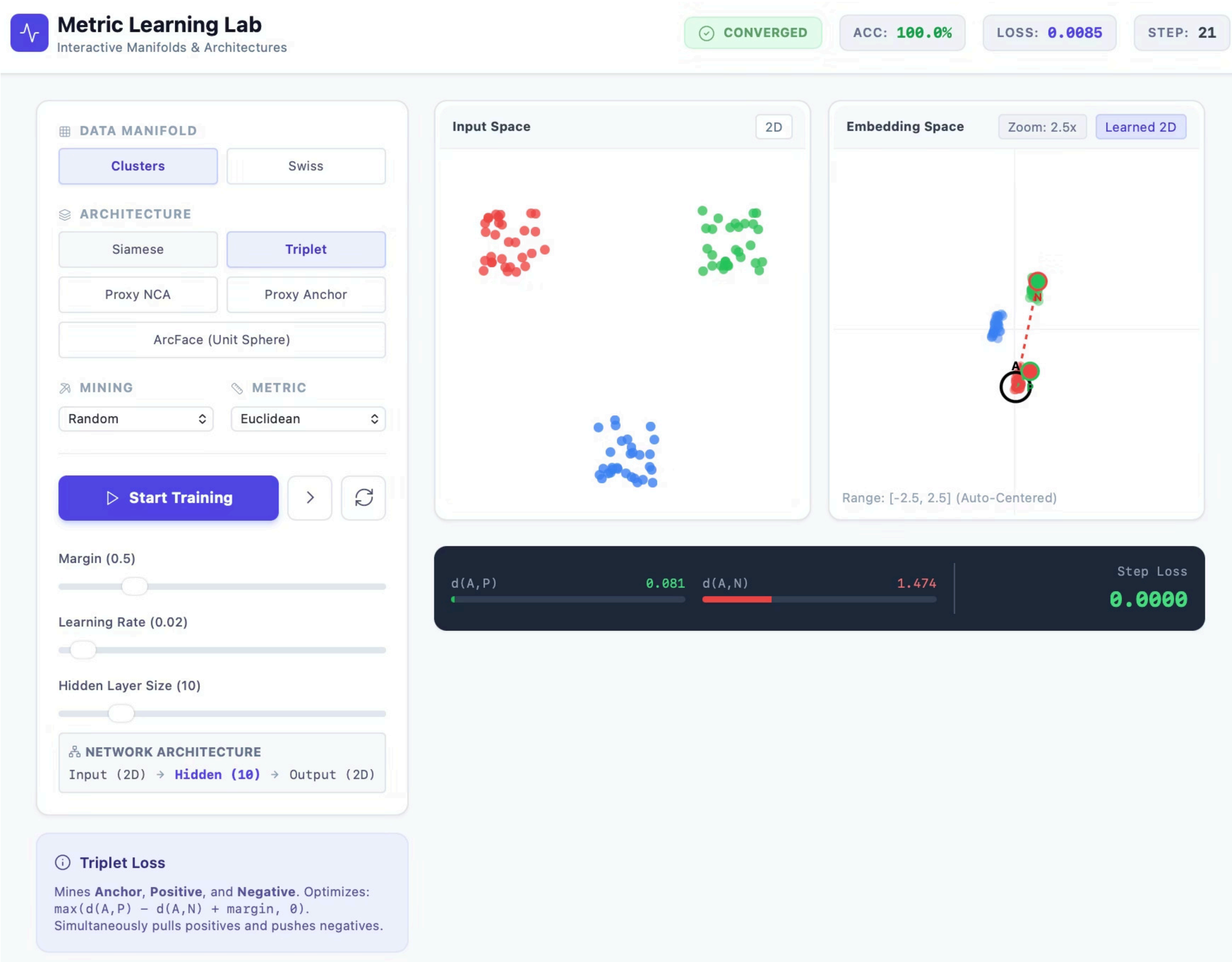
These are "confusing" negatives but not "impossible" ones. They violate the margin but not the ordering, providing informative gradients without the instability of hard mining.



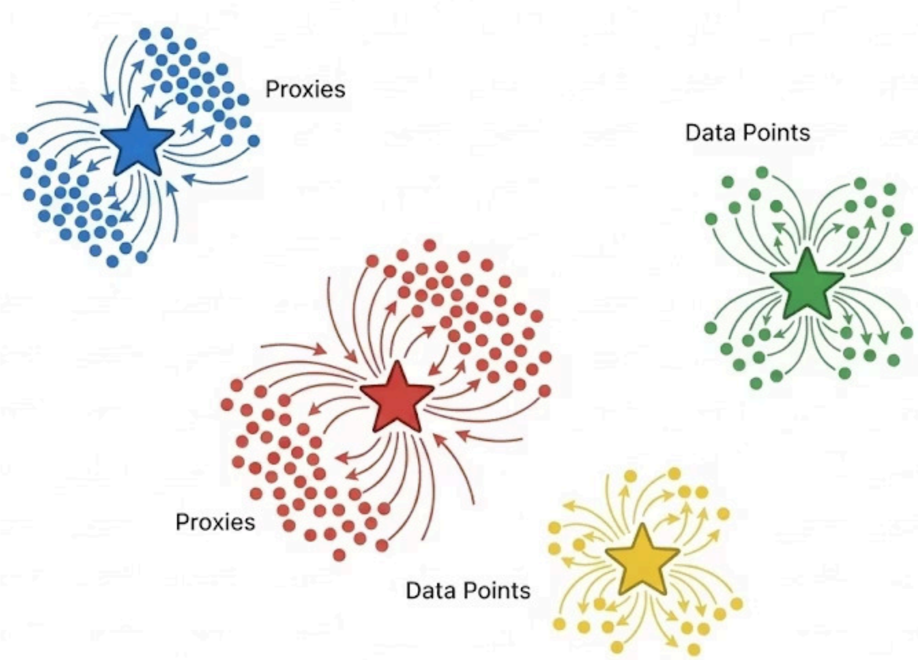
Key Insight: Sampling matters more than architecture in DML. If you feed your network easy triplets, it learns nothing. Proper mining strategies are the difference between convergence in days versus weeks.

Triplet Network Demo

Metric Learning Demo



Proxy-Based Methods
Using Classes Explicitly



> **Note**: proxies are learnable parameters. In coding terms, this means, they are initialized as `nn.Parameter` and added to the optimizer.

Motivation

Triplet mining is expensive and noisy. Computing all pairwise or triplet distances is $O(N^2)$ or $O(N^3)$.

Core Idea

If examples x_i are explicitly associated to classes y_i, we can define "proxy" vectors p_y for each class y.

Instead of comparing Data ↔️ Data, compare Data ↔️ Proxy.

Proxy:

- A learnable vector p_c representing class c.
- Stored in a memory bank, updated via backprop.

Loss:

- Pull $f(x)$ towards its class proxy p_y, push away from other proxies.

Benefits:

- Reduces complexity from $O(N^2)$ or $O(N^3)$ to $O(N \cdot C)$.

Kim, Sungyeon, Dongwon Kim, Minsu Cho, and Suha Kwak. "Proxy anchor loss for deep metric learning." In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3238-3247. 2020.

Learning Mechanism: Proxy-Based Methods

Forward Pass

Input data point x is embedded to $f(x)$. Distances are computed between $f(x)$ and all class proxies p_c.

Backward Pass (Dual Updates)

Gradients update the embedding network f_θ (pulling $f(x)$ to its proxy, pushing from others) and the proxies p_c (moving them towards class centers).

> **Note**: Proxies are **not** computed as the average of points of the same class. They are randomly initialized parameters and are free to move during training. The Proxy-NCA loss pulls each class proxy toward its class embeddings (and pushes it away from others), so proxies naturally drift toward class centers as an *emergent* effect of optimization, without any explicit 'center' term.

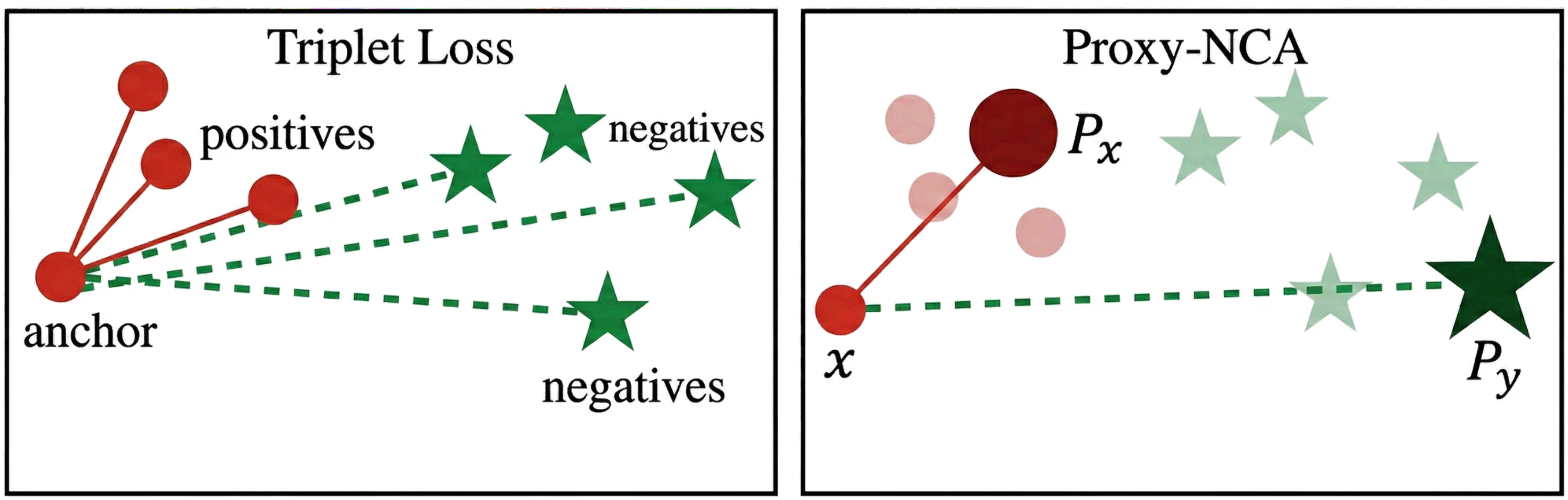
Loss Function (Proxy-NCA)

$$L(x) = -\log \frac{\exp(-d(x, p_y))}{\sum_{c=1}^{C} \exp(-d(x, p_c))}$$

Where, $d(x, p_y) = D(f(x), p_y)$ and D is chosen as the Euclidean distance.

This loss pulls the embedding $f(x)$ towards its true class proxy p_y while simultaneously pushing it away from all other negative proxies. Proxies p_c are learnable parameters updated via backpropagation to optimally represent their respective classes.

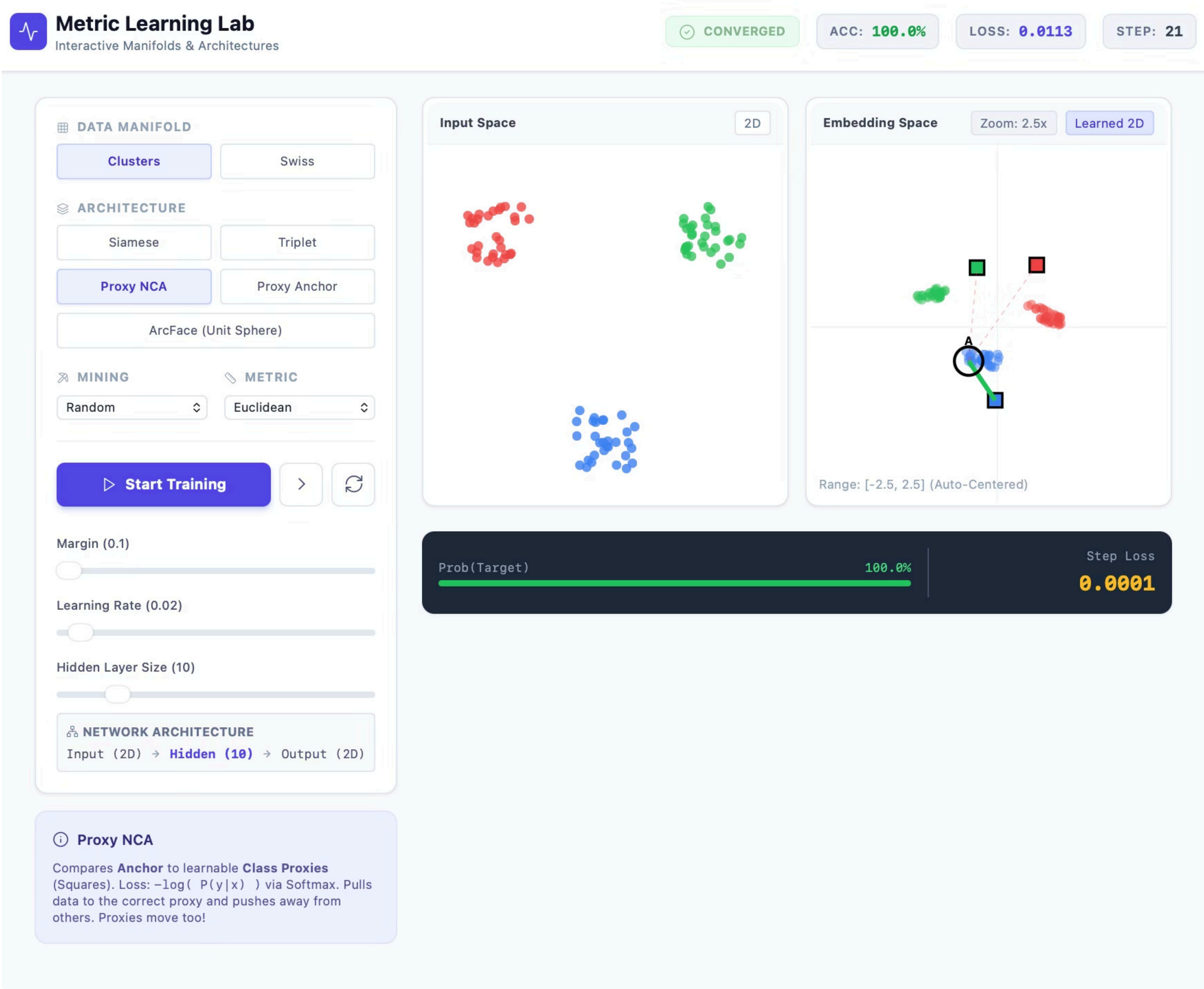
> **Note**: the loss function is a proxy-based variant of the NCA (Neighbourhood Components Analysis) objective, and can be seen as a softmax cross-entropy that tries to correctly 'classify' the current sample xx among proxies, where proxies play the role of class weight vectors.



Kim, Sungyeon, Dongwon Kim, Minsu Cho, and Suha Kwak. "Proxy anchor loss for deep metric learning." In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3238-3247. 2020.

Proxy-NCA Demo

Metric Learning Demo



Motivation: Moving Beyond Proxy-NCA

Problems with Proxy-NCA

- **Fixed Positive Gradient:** All positive examples pull with the same force, regardless of their distance to the proxy.
- **Ignores Data-to-Data Hardness:** No mechanism to prioritize harder examples within a batch, leading to inefficient learning.
- **Limited Fine-Grained Control:** Cannot exploit which positives/negatives are more challenging to optimize.

Idea - Use proxies as anchors and compare to all data in the batch

Instead of each data point comparing itself to all proxies, Proxy-Anchor flips this: each proxy now acts as an anchor, evaluating its relationship with all positive and negative samples in the batch. This fundamental shift enables more nuanced comparisons.

Key Benefits of Proxy-Anchor

Efficiency

O(N·C) complexity, matching Proxy-NCA. Much faster convergence than Triplet or N-Pair, eliminating costly tuple mining.

Hardness Weighting

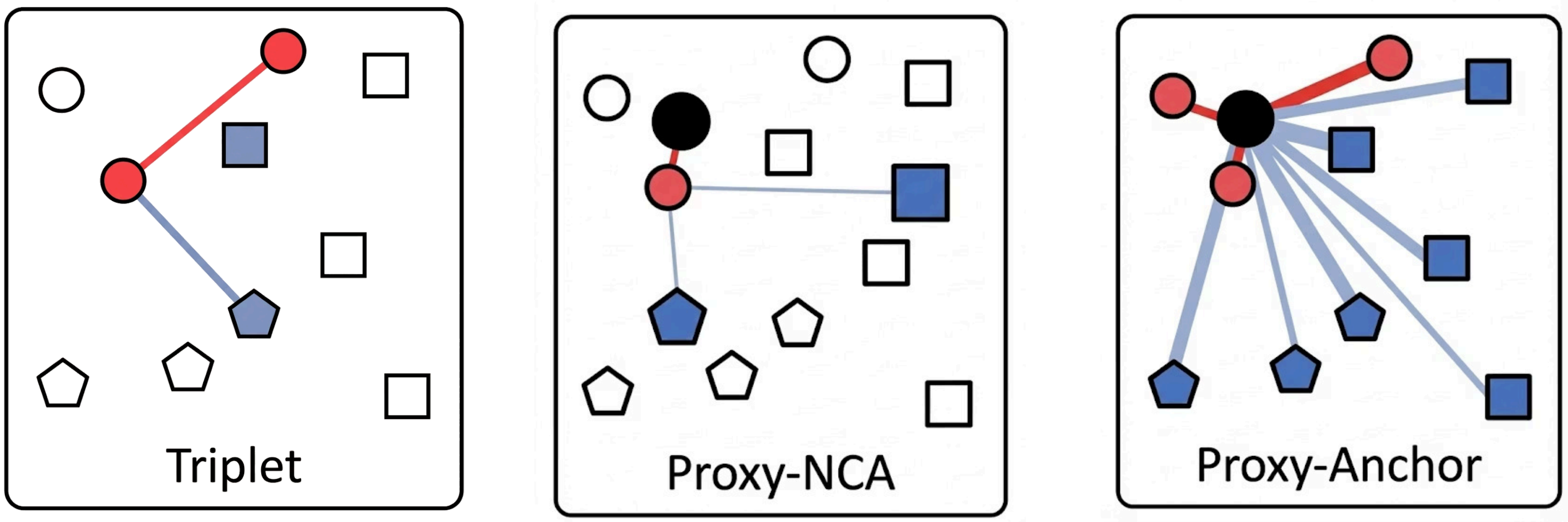
Enables **relative hardness weighting**: stronger pulls for far positives, stronger pushes for near negatives, optimizing informative gradients.

Robustness

More **robust to noisy labels** and outliers, preserving the integrity of the proxy representation.

Hybrid Strength

Combines proxy-based speed with the **fine-grained structure** and rich supervisory signals of pair-based losses.



Kim, Sungyeon, Dongwon Kim, Minsu Cho, and Suha Kwak. "Proxy anchor loss for deep metric learning." In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3238-3247. 2020.

Learning Mechanism: Proxy-Anchor Loss

We use the following loss function:

$$\ell(X) = \frac{1}{|P^+|} \sum_{p \in P^+} \log \left(1 + \sum_{x \in X_p^+} e^{-\alpha(s(x,p)-\delta)} \right) + \frac{1}{|P|} \sum_{p \in P} \log \left(1 + \sum_{x \in X_p^-} e^{\alpha(s(x,p)+\delta)} \right)$$

Where:

- X is the batch
- P^+ is the set of proxies such that at least an element (a positive) appears in X
- X_p^+ is the set of elements in X which are of the same class as proxy p (positives for p)
- α is a scaling factor
- s is the similarity matrix
- δ is the margin
- P is the full set of proxies
- X_p^- is the set of elements in X which are of a different class from p

We can "read" the loss as follows:

- (First term) Take all proxies of classes appearing in the batch. For each proxy, maximise the similarities between the proxy and all positives, beyond margin δ. In practice, if $s(x,p) > \delta$ for all positives, the exponential $e^{\cdots}$ is smaller than zero and $log(1+\ldots) \approx 0$.
- (Second term) Take all proxies. For each of them, sample negatives and minimise the similarities between the proxy and the negatives below margin δ. In practice, if $s(x,p)) < \delta$, then $e^{\cdots}$ is smaller than zero and the term $log(1+\ldots) \approx 0$.

Kim, Sungyeon, Dongwon Kim, Minsu Cho, and Suha Kwak. "Proxy anchor loss for deep metric learning." In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3238-3247. 2020.

Weighing on hardness

Unlike Proxy-NCA (which uses constant gradients for positives), each sample's gradient in Proxy-Anchor is normalized by its hardness relative to others in the batch, enabling data-to-data interactions for more nuanced optimization.

We define positive and negative hardness as follows:

Positive Hardness

$$h_p^+(x) = \exp\big(-\alpha\,(s(x,p)-\delta)\big)$$

Negative Hardness

$$h_p^-(x) = \exp\big(\alpha\,(s(x,p)+\delta)\big)$$

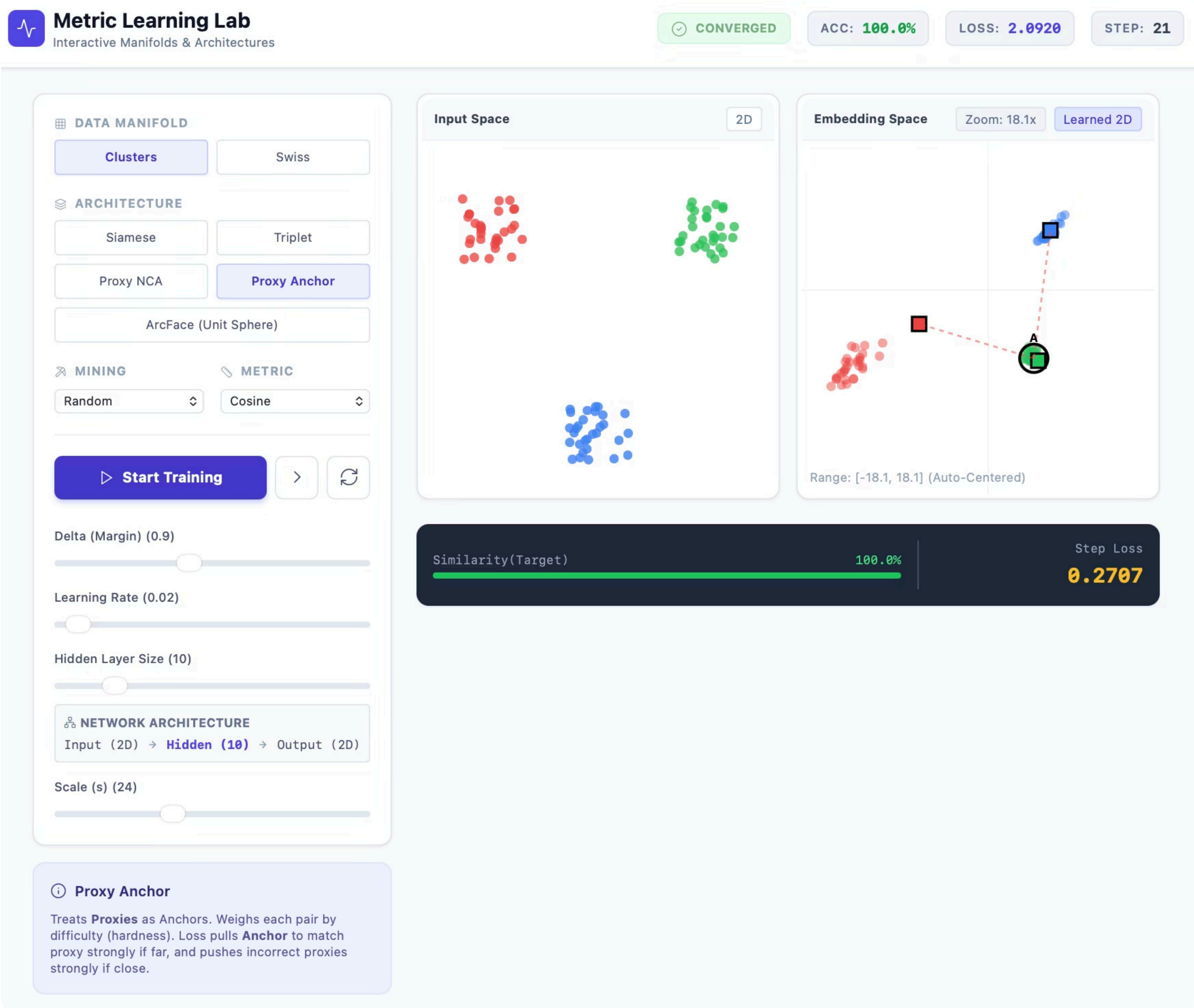
Here, positive hardness $h_p^+(x)$ is large when a positive sample is **far** from its proxy, while negative hardness $h_p^-(x)$ is large when a negative sample is **close** to a proxy. This ensures informative gradients.

Indeed, it can be shown that the gradient weight for each sample is proportional to an exponential **hardness** term: hard positives (far from the proxy) and hard negatives (close to the proxy) receive larger updates, while easy samples contribute almost nothing.

Kim, Sungyeon, Dongwon Kim, Minsu Cho, and Suha Kwak. "Proxy anchor loss for deep metric learning." In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3238-3247. 2020.

Proxy-Anchor Demo

Metric Learning Demo



Classification Weights ARE Proxies

Revisiting Softmax

$$P(y|x) = \frac{e^{W_y^T x + b_y}}{\sum e^{W_i^T x + b_i}}$$

Let's assume no bias terms $b_y = b_i = 0$.

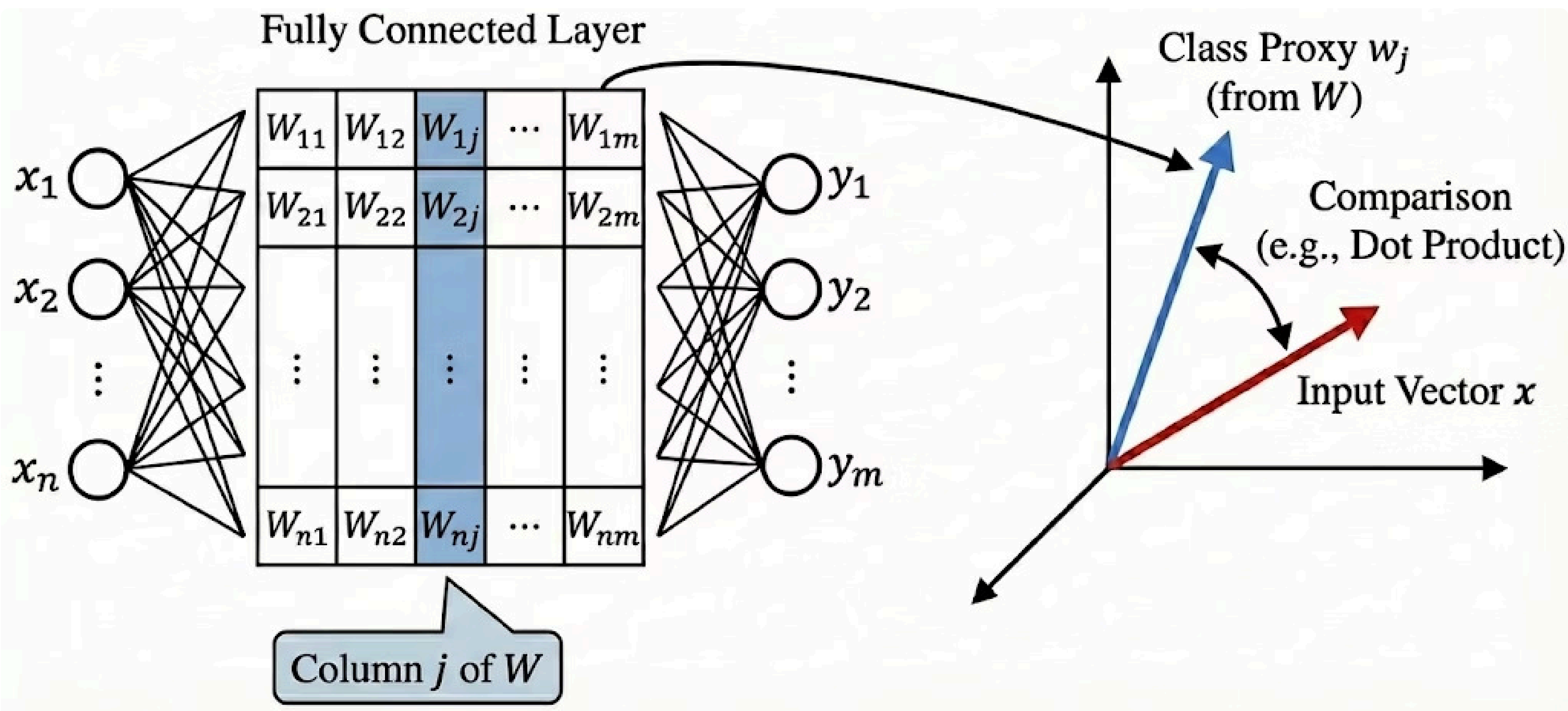
- The last layer of a classifier computes $z_j = W_j \cdot x$
- W_j is a weight vector for class j
- The dot product measures similarity between input and class weights

The Insight

If we constrain W_j and x to be unit vectors (this can be done by dividing the output of the network by its module):

- W_j acts exactly like a Proxy for class j
- The dot product $W_j \cdot x$ measures cosine similarity between the input and the class proxy

> **Classification is Metric Learning if we constrain the geometry**



Deng, Jiankang, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. "Arcface: Additive angular margin loss for deep face recognition." In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4690-4699. 2019.

Learning Mechanism: ArcFace Loss

The above intuition is formalised in the the ArcFace, formulated as follows:

$$L(x_i, y_i) = -\log \frac{e^{s\cdot\cos(\theta_{y_i}+m)}}{e^{s\cdot\cos(\theta_{y_i}+m)} + \sum_{j\neq y_i} e^{s\cdot\cos(\theta_j)}}$$

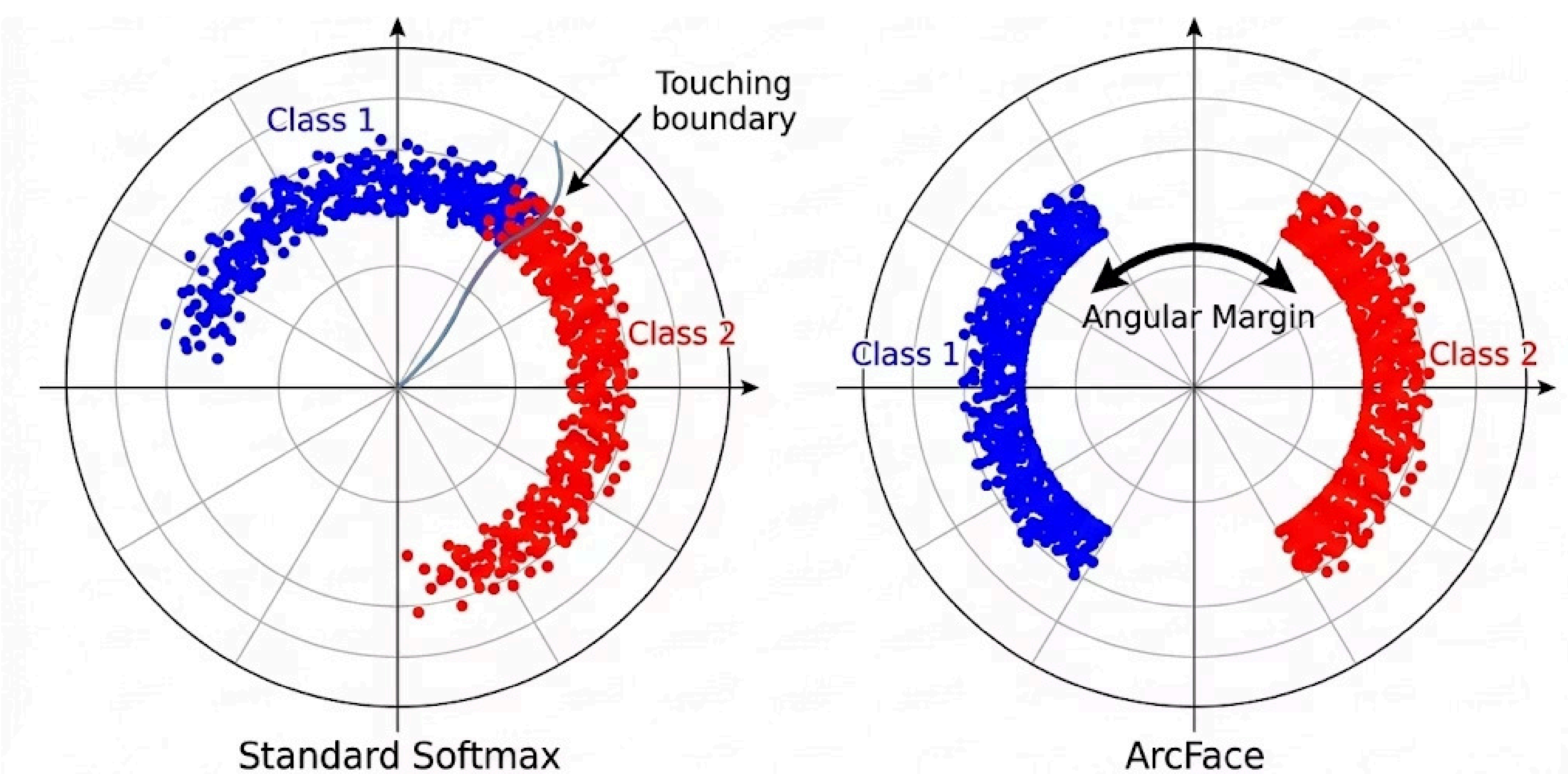
Where:

- x_i is the input data point
- y_i is its class
- s is a feature scale or temperature (a scaling factor)
- θ_y is the angle between proxy W_y and data point x_i computed as $\theta_y = arccos(W_y^T x_i)$
- m is a margin

ArcFace introduces the additive angular margin m to the ground-truth class proxy only, directly optimizing for greater discriminative power in embedding spaces. This method ensures that feature embeddings for different classes are separated by a clear angular boundary, leading to highly distinct representations.

The loss function components serve distinct purposes:

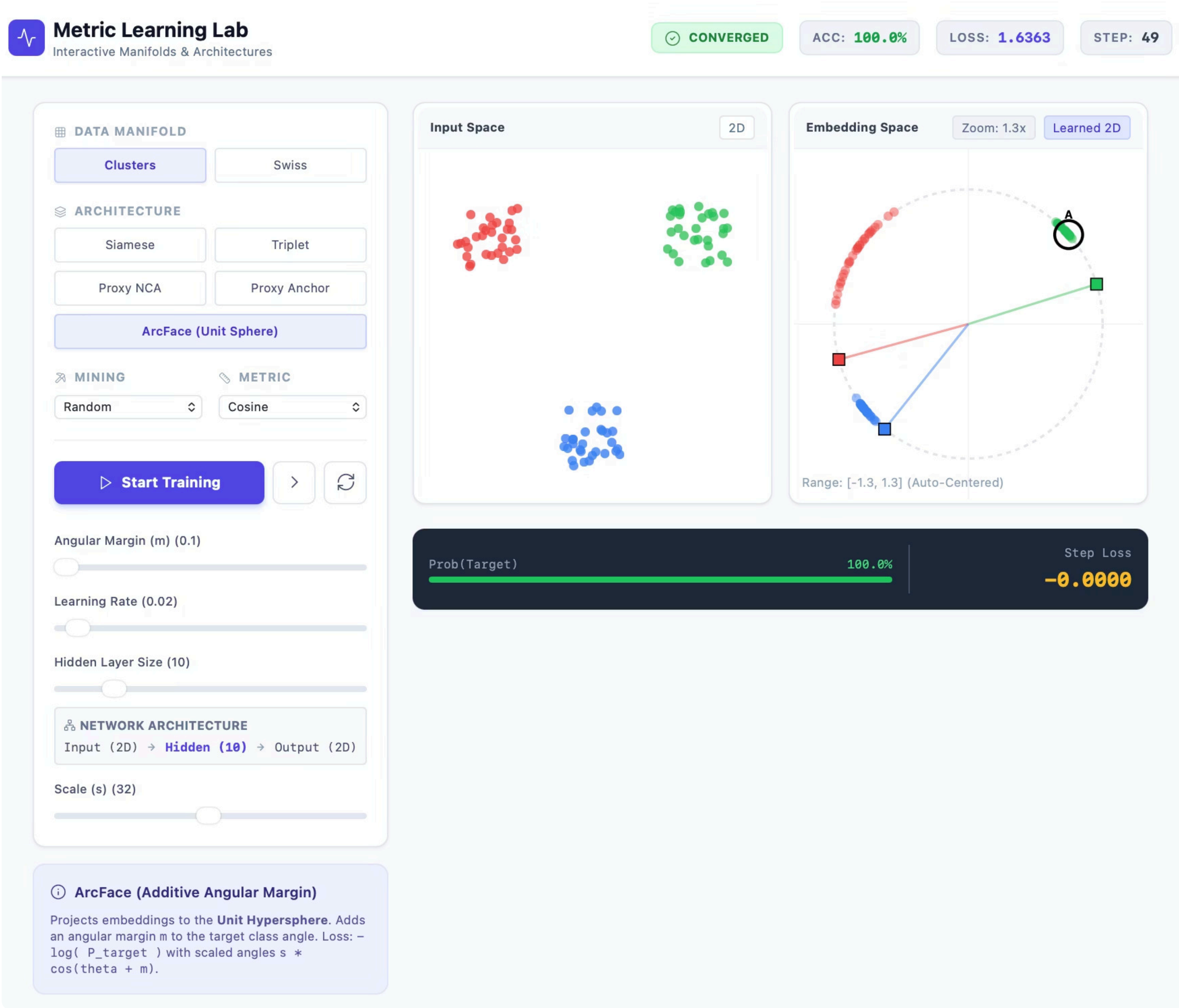
- **Positive terms:** This part of the loss aims to maximize the similarity (represented by $\cos(\theta)$) between a sample's embedding and its ground-truth class proxy. The additive margin $+m$ is applied to the angle θ_y, which effectively makes the decision boundary for the positive class more stringent. If $\theta_y + m$ is sufficiently large, meaning the sample is very close to its proxy with the required margin, this term's contribution to the total loss approaches zero.
- **Negative term:** The summation over all other classes $j \neq y$ in the denominator ensures that the sample's embedding is pushed away from incorrect class proxies. This term works to minimize the angular similarity to negative classes, thereby enforcing a clear separation. The softmax mechanism intrinsically drives the decision boundary to create an angular gap where $\cos(\theta_y + m) > \cos(\theta_j)$ for all other classes.



Deng, Jiankang, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. "Arcface: Additive angular margin loss for deep face recognition." In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4690-4699. 2019.

ArcFace Demo

Metric Learning Demo



Metric Learning in the Foundation Model Era

Do we still train from scratch? **Often, No.**

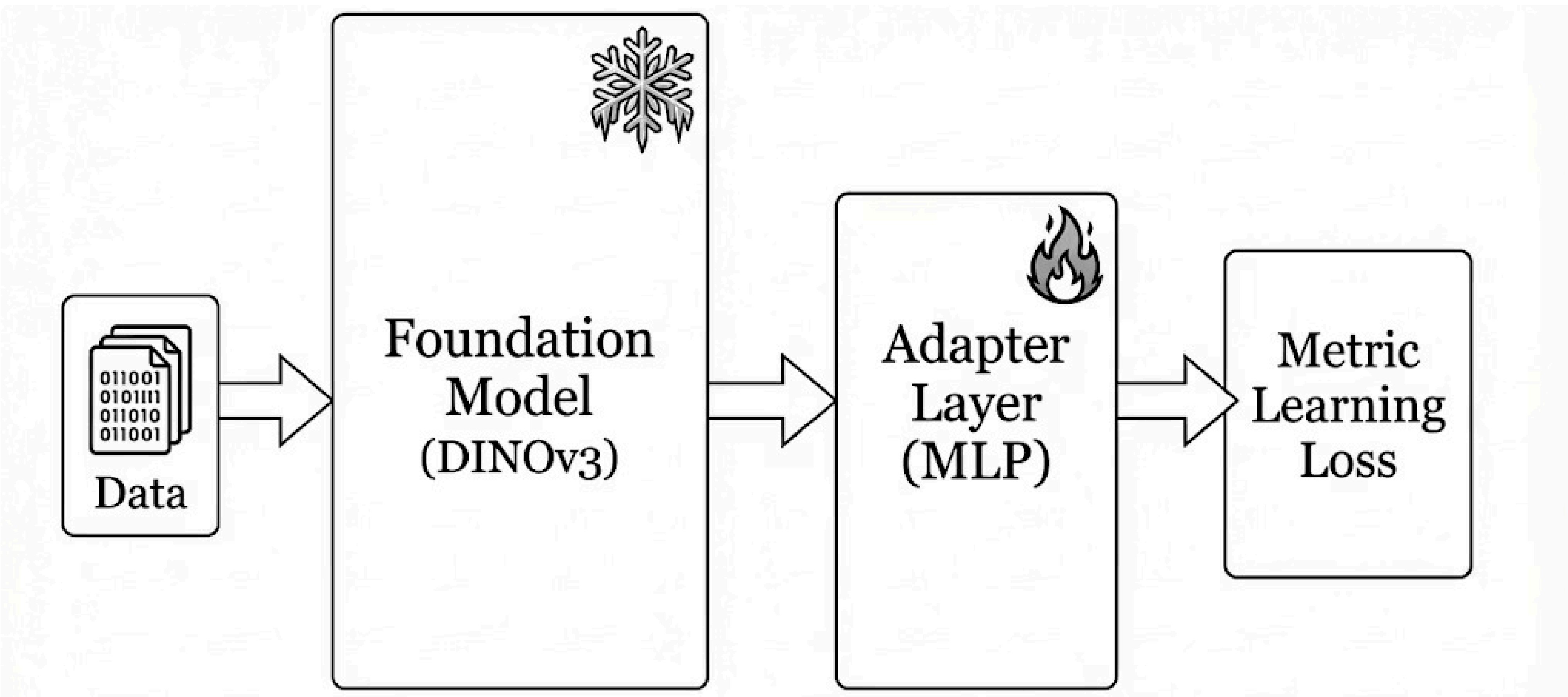
Foundation Models

CLIP / DINOv2: Trained on billions of images using Contrastive Learning, these models already possess a powerful and generalized metric space, learned from vast datasets.

The Modern Workflow

1. Extract features with DINOv2 (Frozen).
2. Train a lightweight "Adapter" or "Head" using ArcFace/Proxy-Anchor.

Result: State-of-the-art performance with a fraction of the cost and data, leveraging pre-trained knowledge.



How do we know it works?

Visual Assessment (Dimensionality Reduction)

Tools like t-SNE and UMAP project high-dimensional embeddings (e.g., 512D) into 2D or 3D for human inspection, allowing us to see how well classes are separated.

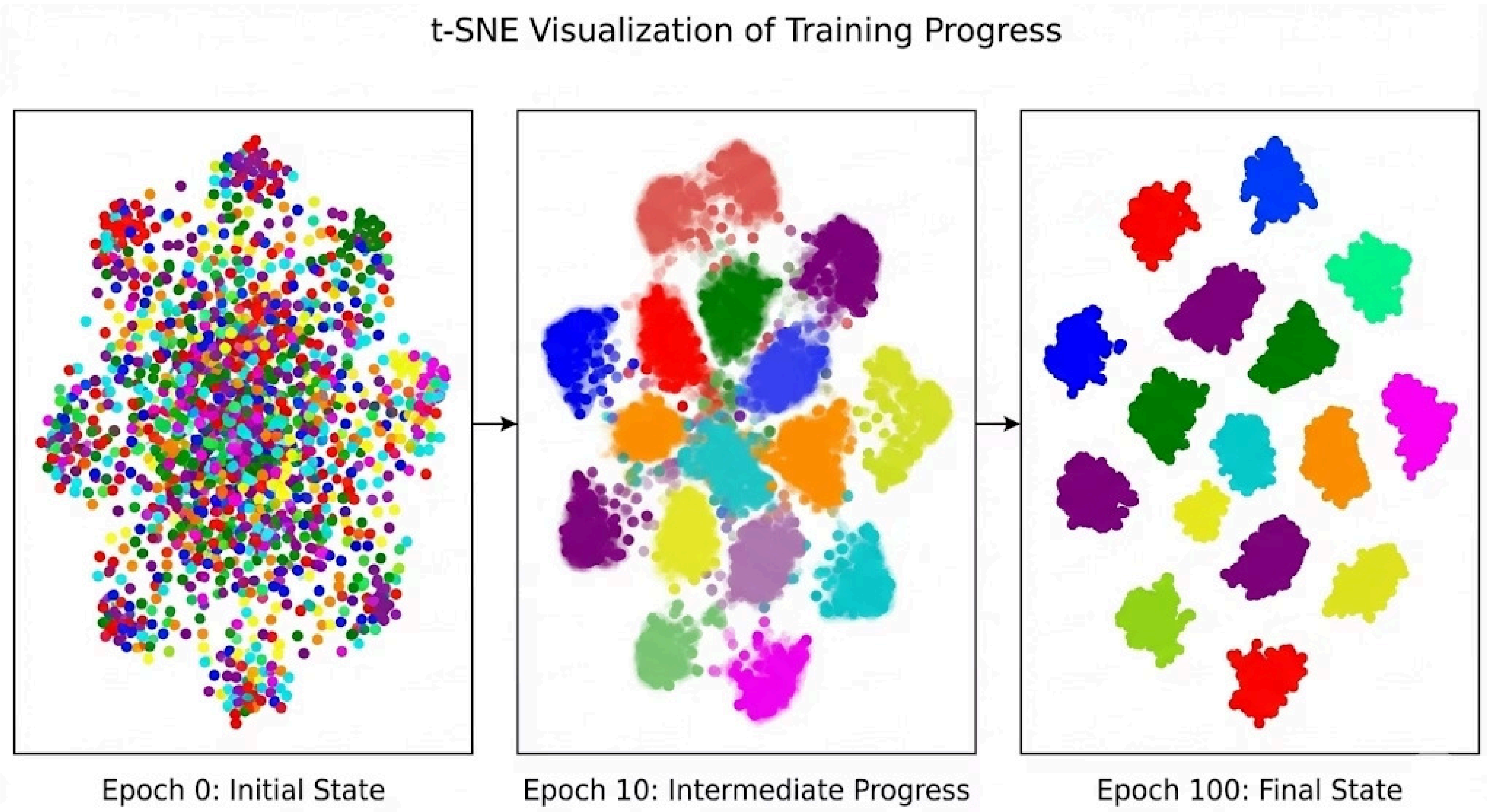
The primary goal is to visually confirm that different classes form distinct, well-separated clusters in the projected space.

> Visualizations can be misleading! Parameters like 'perplexity' or 'number of neighbors' in t-SNE/UMAP significantly alter the perceived shape and separation of clusters.

Quantitative Metrics (The Truth)

Relying solely on visual cues is insufficient. Robust quantitative metrics are essential for accurate evaluation:

- **Recall@K:** Measures if the correct match for a given query is present within the top K retrieved results. Higher is better.
- **NMI** (Normalized Mutual Information): Assesses the quality of clustering by measuring how well discovered clusters align with ground-truth class labels. Higher NMI indicates better clustering.
- **R-Precision:** Calculates precision at a cutoff equal to the total number of relevant items (R) for each query. This gives a balanced view of retrieval performance.



Where is Deep Metric Learning used?

Deep Metric Learning powers a variety of real-world applications where precise similarity comparison is crucial for effective results.

Face Recognition

This domain is heavily dominated by sophisticated Angular Losses like ArcFace, which create highly discriminative feature embeddings.

The ability to handle millions of identities efficiently makes proxy-based methods absolutely essential for large-scale deployments.

Person Re-Identification (Re-ID)

Re-ID often employs a hybrid approach, combining Triplet Loss for fine-grained local detail matching with Softmax/Proxy-based losses for robust global cluster separation across different camera views.

Visual Product Search

Applications like "Find similar shoes" leverage Deep Metric Learning. Proxy-Anchor, for instance, is highly effective for training on massive datasets with 100k+ classes, such as those found in Google Landmarks or large e-commerce catalogs.

Real-World Deployment: Visual Search

The core challenge is how to efficiently search through 100 million or more images and return relevant results within milliseconds.

1. Offline Indexing

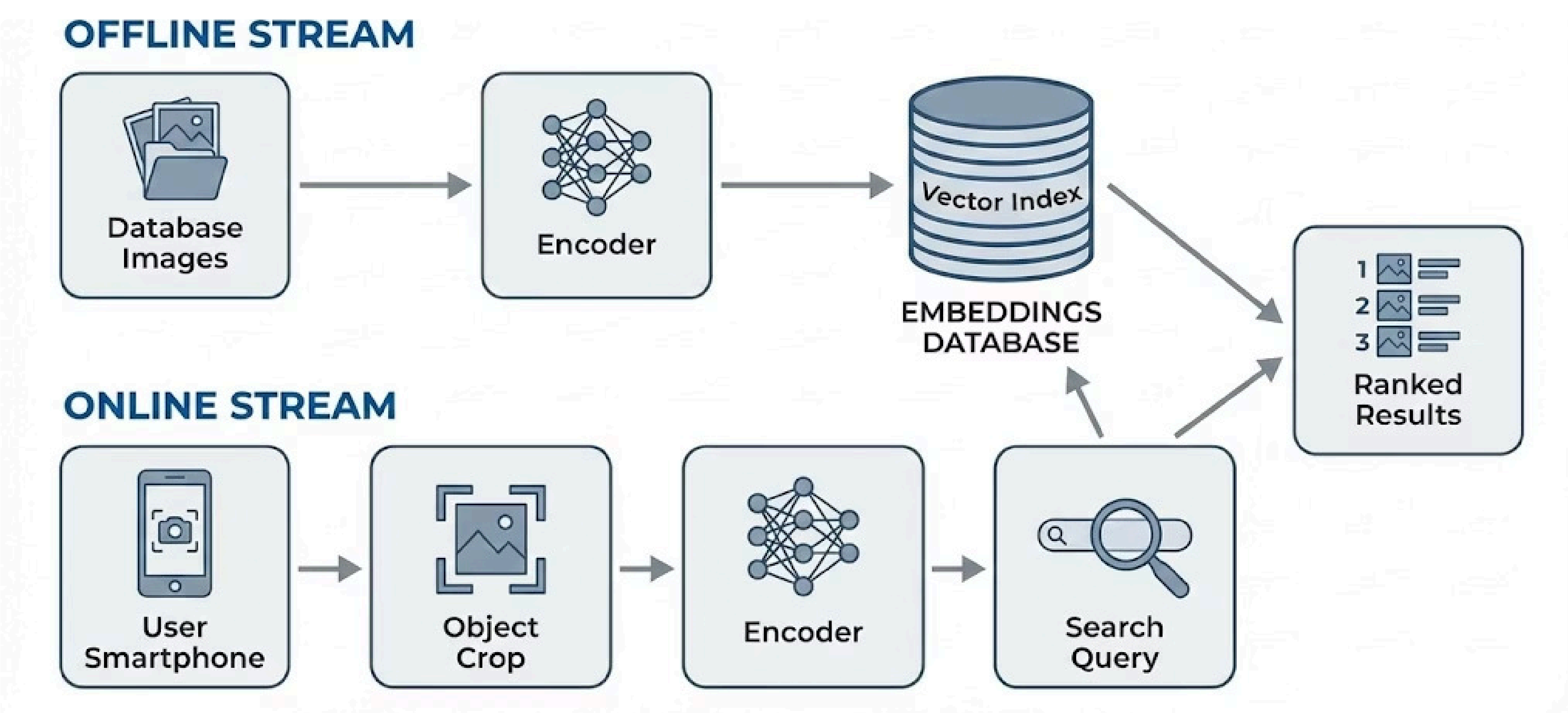
All catalog images are passed through a pre-trained Deep Metric Learning backbone to generate their embeddings. These high-dimensional vectors are then stored in an Approximate Nearest Neighbor (ANN) index like Faiss or HNSW for rapid similarity search. To optimize memory usage, techniques such as Product Quantization (PQ) are often applied to compress these embeddings.

2. Online Query & Embedding

When a user provides an input photo (e.g., of a shoe), an object detection model first crops the region of interest. This cropped image is then fed into the same DML backbone used during indexing to generate its corresponding embedding vector, placing the query into the same metric space as the catalog.

3. Retrieval & Ranking

The generated query embedding is used to perform a "coarse search" on the ANN index, quickly retrieving the top 1,000 most similar candidate images. These candidates are then subjected to a "re-ranking" step, where their exact cosine distance to the query is calculated, providing a precise ordering of results for the user.



Collaborative Metric Learning (CML)

In recommender systems, predicting user preferences is key. Collaborative Metric Learning offers a powerful approach to model these interactions by leveraging distance-based relationships.

The Challenge: Matrix Factorization's Limitations

- Traditional **Matrix Factorization (MF)** models predict preference via a simple dot product: $\hat{y}_{ui} = \mathbf{u}_i \cdot \mathbf{v}_j$.
- This approach fundamentally struggles with the Triangle Inequality. Just because User U likes Item A, and Item A is similar to Item B, traditional MF doesn't guarantee U is positioned close to B in its latent space.
- This lack of transitivity can lead to suboptimal recommendations.

The Metric Learning Solution

Collaborative Metric Learning (CML), introduced by Hsieh et al. (WWW 2017), transforms this problem by mapping users and items into a unified metric space. The objective is to minimize the distance between User U and their positively interacted items V:

$$d(u, v) = \|\mathbf{u} - \mathbf{v}\|_2^2$$

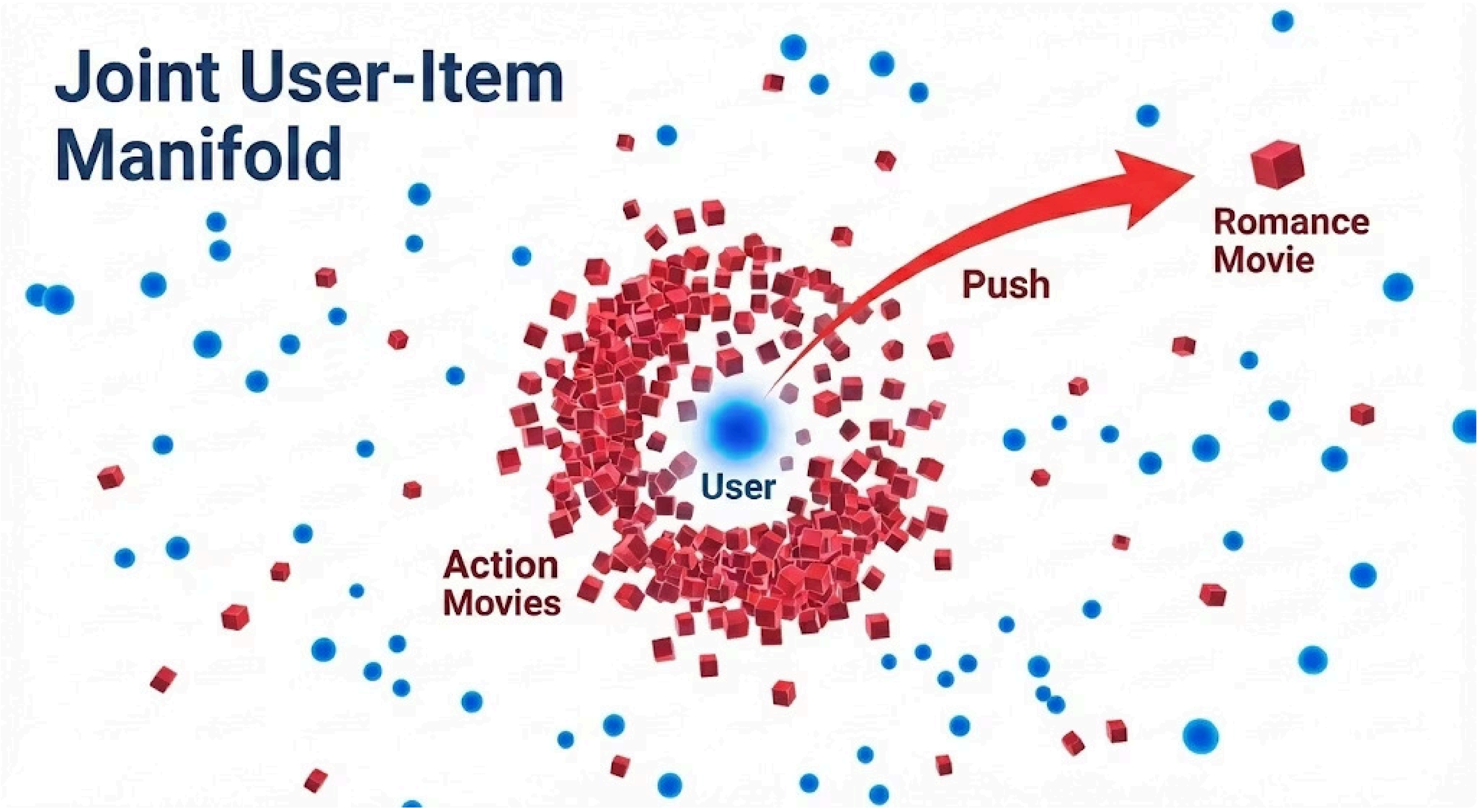
Crucially, CML uses a margin loss to explicitly push non-interacted items away from the user, ensuring clear separation.

Enforces Transitivity

By operating in a metric space, CML inherently respects the Triangle Inequality, allowing for more consistent and logical recommendations based on inferred similarity.

Better Generalization

CML excels with implicit feedback datasets and addresses the "cold start" problem more effectively for new users or items due to its robust distance-based embedding.



Metric Learning for RAG Pipelines

Deep Metric Learning is pivotal in Retrieval-Augmented Generation (RAG), enabling large language models to find and use relevant information by training sophisticated text embedding models.

Query Example

"How do I reset my router?"

Positive Document

"To reset your router, hold the button for 10s..."

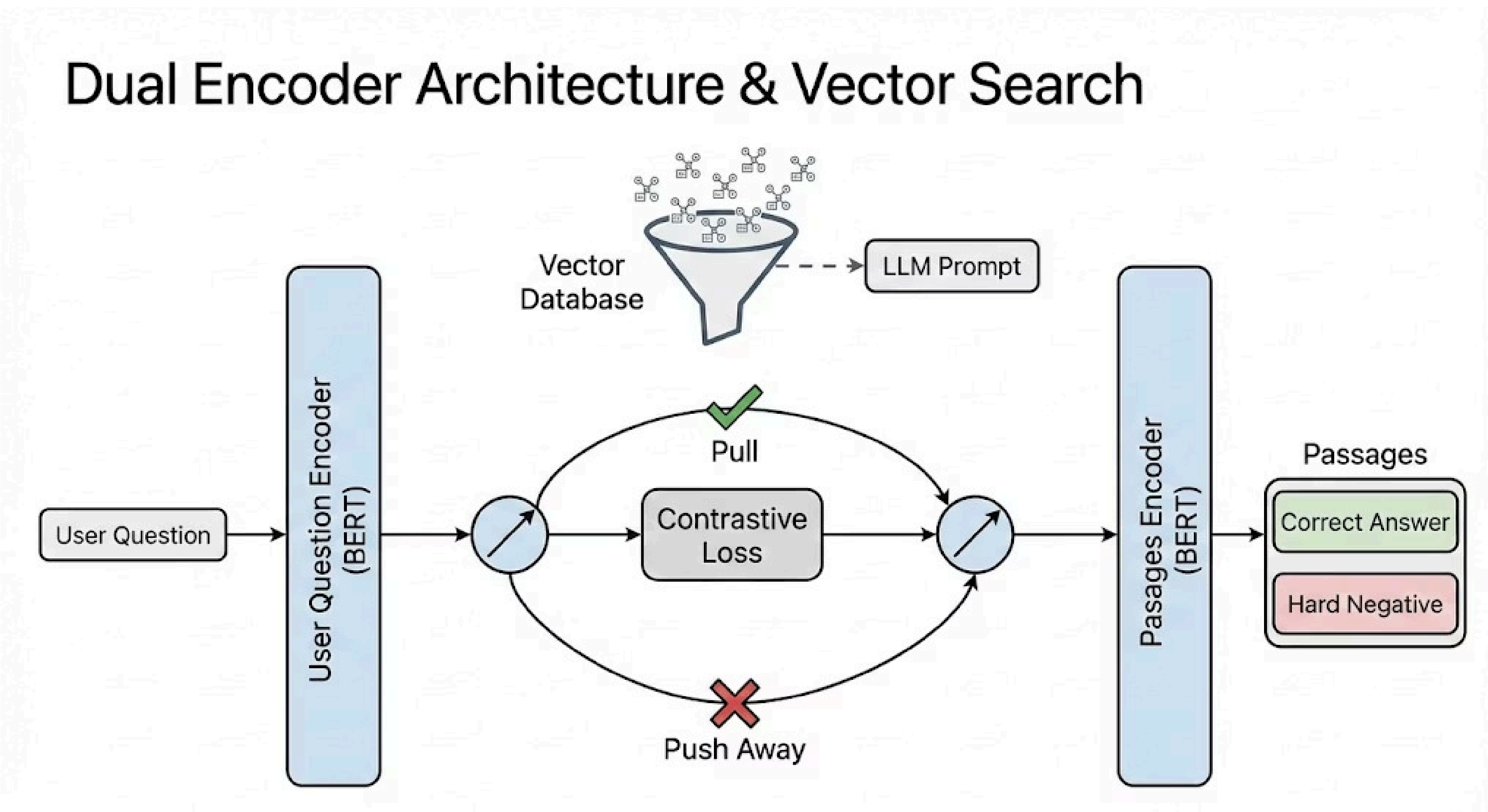
Negative Document

"The history of the router dates back to..."

Training Methodology: Contrastive Learning

- **In-Batch Negatives:** Leverage other samples within the batch as negatives (e.g., a batch size of N provides N-1 negatives for each query).
- **Hard Negative Mining:** Crucial for performance, focusing on negatives that are lexically similar but semantically irrelevant.
- **Loss Function:** Typically InfoNCE, a softmax-based version of Contrastive Loss.

This process creates a semantic vector space where questions and correct answers are embedded closely, facilitating highly effective vector search for LLMs.

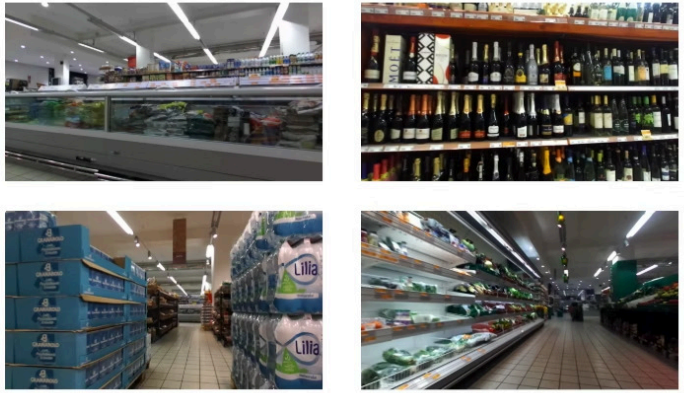


> **Note**: We will see InfoNCE in details when talking about self-supervised learning.

Metric Learning for Camera Localization

Problem Setup

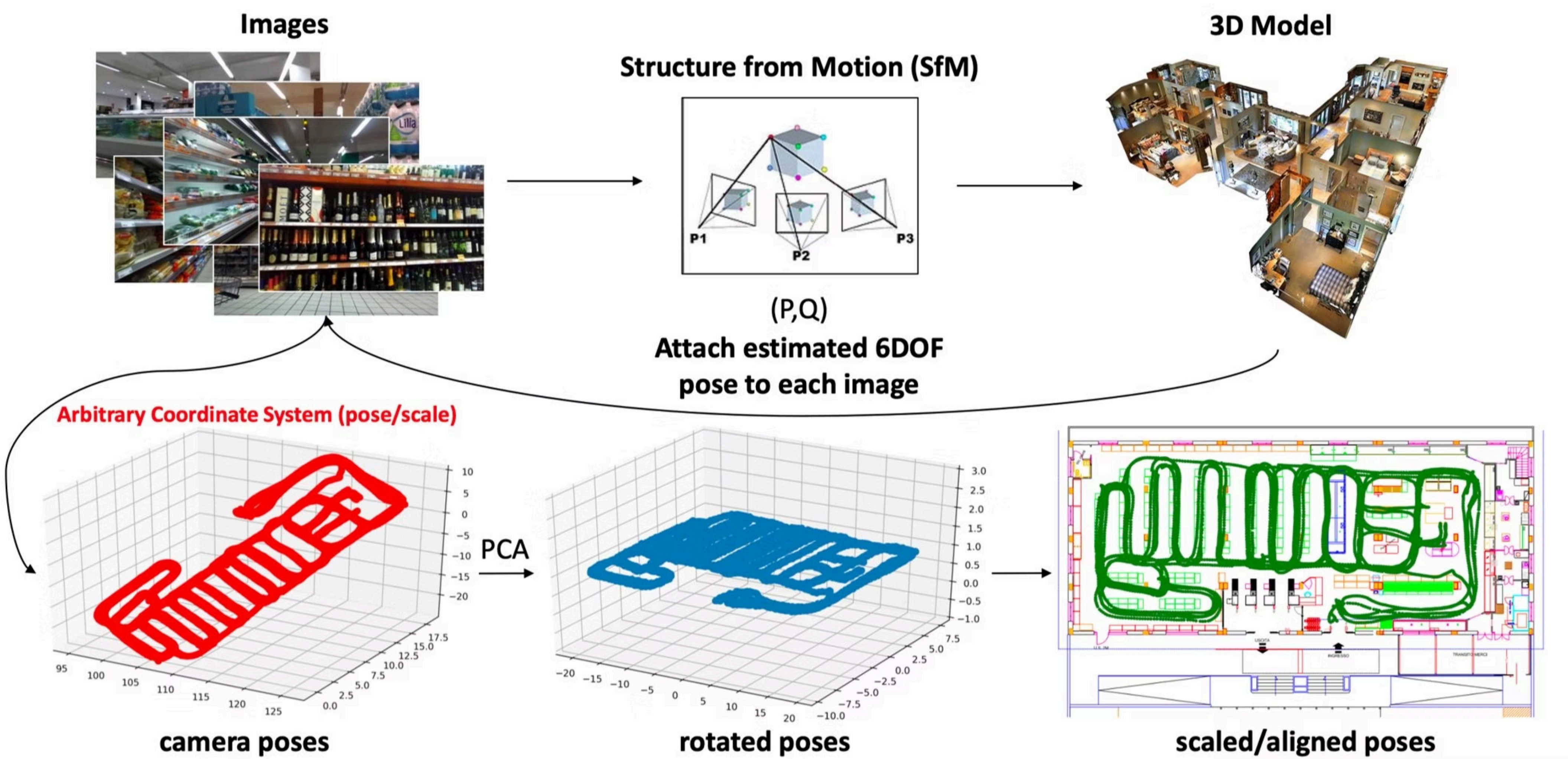
Given an image, localize the shopping cart in the supermarket



Emiliano Spera, Antonino Furnari, Sebastiano Battiato, Giovanni M. Farinella (2021). EgoCart: a Benchmark Dataset for Large-Scale Indoor Image-Based Localization in Retail Stores. IEEE Transactions on Circuits and Systems for Video Technology, 31, pp. 1253-1267.

Metric Learning for Camera Localization

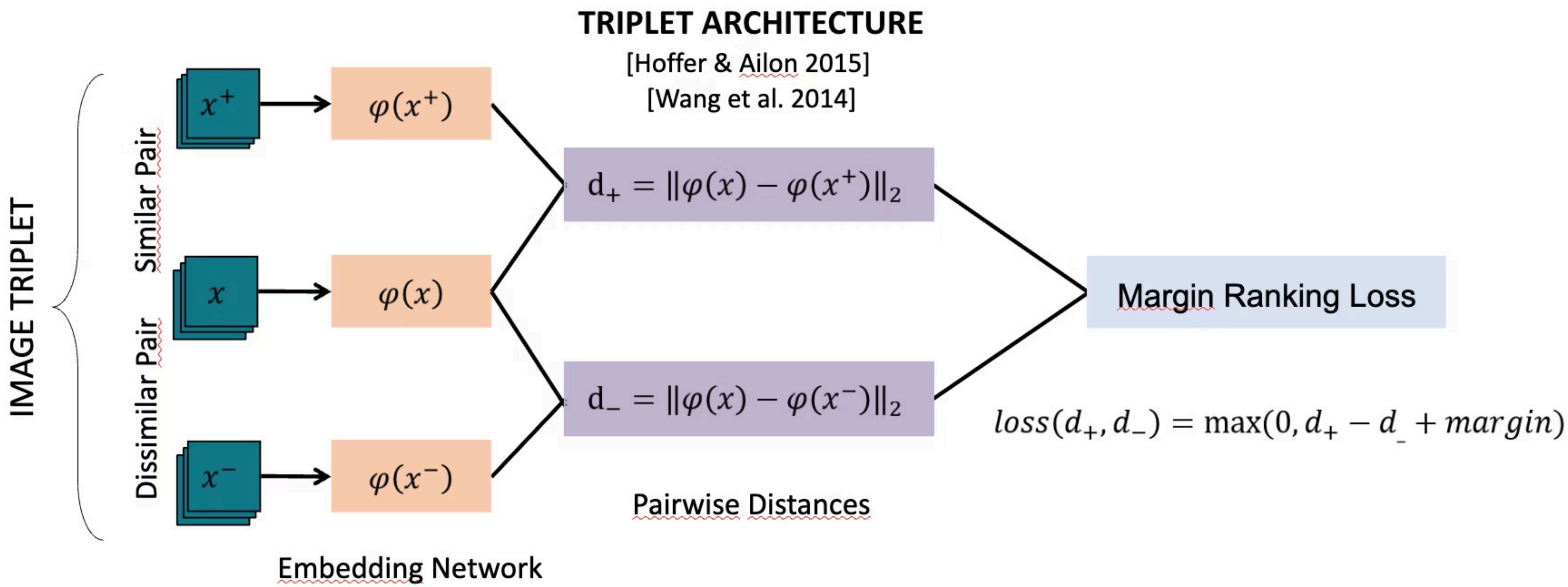
Dataset Creation



Emiliano Spera, Antonino Furnari, Sebastiano Battiato, Giovanni M. Farinella (2021). EgoCart: a Benchmark Dataset for Large-Scale Indoor Image-Based Localization in Retail Stores. IEEE Transactions on Circuits and Systems for Video Technology, 31, pp. 1253-1267.

Metric Learning for Camera Localization

Learning the Distance Function



Triplets formed based on location similarity/dissimilarity:

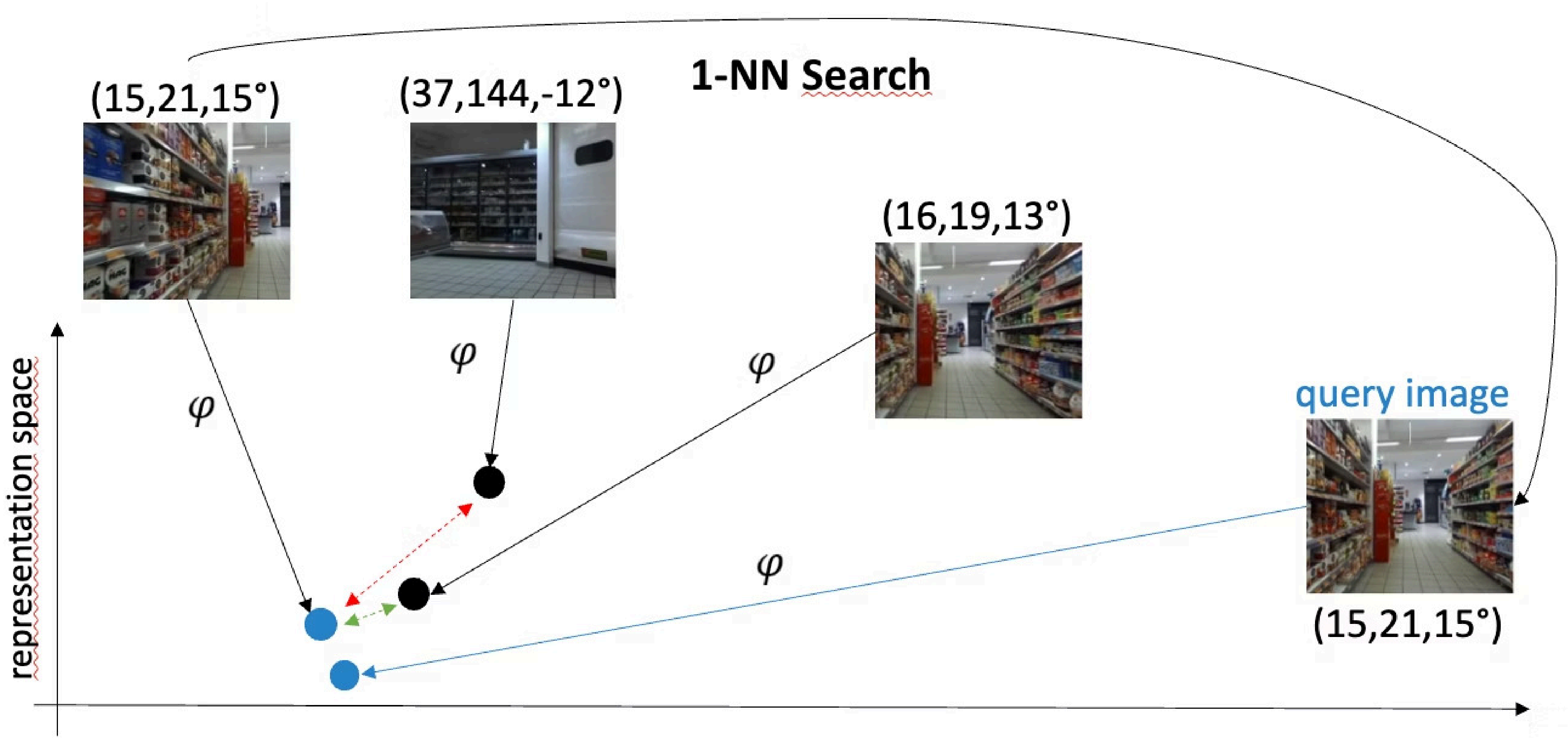
- Select a random image as anchor
- Select a distant image (distance in camera position over a threshold) as a negative
- Select a close image (distance in camera position below a threshold) as a positive

> **Note** that this approach does not require to define "classes" at all.

Emiliano Spera, Antonino Furnari, Sebastiano Battiato, Giovanni M. Farinella (2021). EgoCart: a Benchmark Dataset for Large-Scale Indoor Image-Based Localization in Retail Stores. IEEE Transactions on Circuits and Systems for Video Technology, 31, pp. 1253-1267.

Metric Learning for Camera Localization

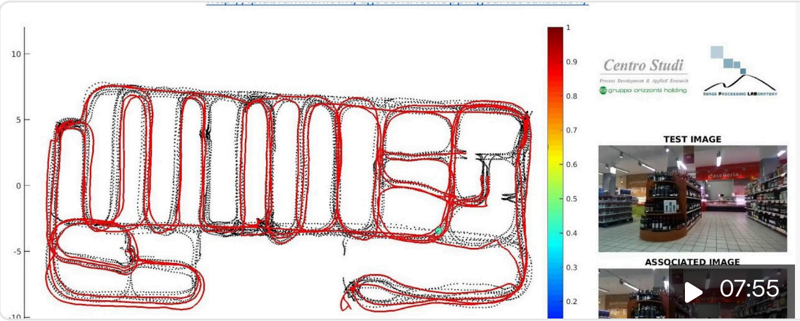
Localization Through a 1-NN Search



Emiliano Spera, Antonino Furnari, Sebastiano Battiato, Giovanni M. Farinella (2021). EgoCart: a Benchmark Dataset for Large-Scale Indoor Image-Based Localization in Retail Stores. IEEE Transactions on Circuits and Systems for Video Technology, 31, pp. 1253-1267.

Metric Learning for Camera Localization

Demo



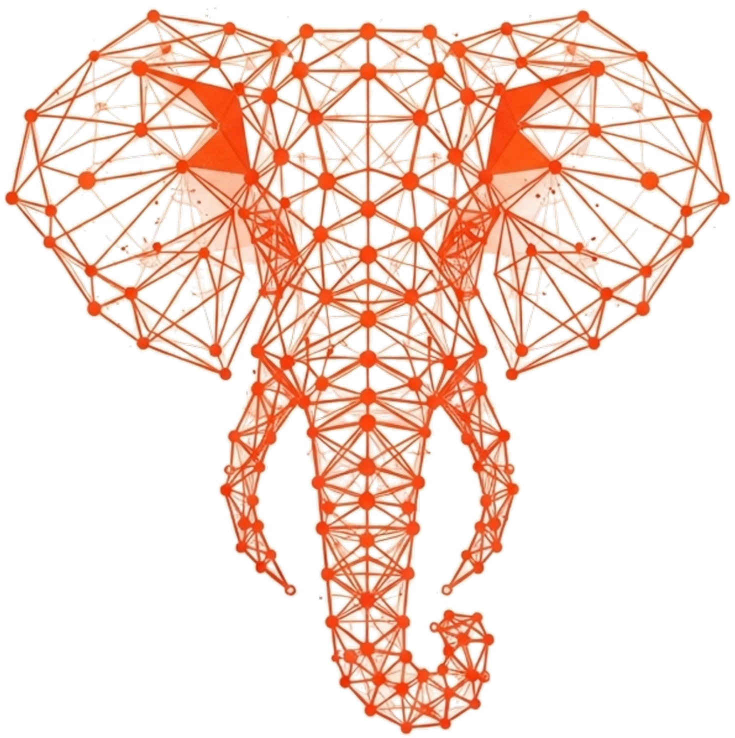
YouTube

Egocentric Shopping Cart Localization

The video shows the performance of the 1–NN approach based on image representations obtained using deep metric learning (Triplet Networks). On the right, we report the query image...

Emiliano Spera, Antonino Furnari, Sebastiano Battiato, Giovanni M. Farinella (2021). EgoCart: a Benchmark Dataset for Large-Scale Indoor Image-Based Localization in Retail Stores. IEEE Transactions on Circuits and Systems for Video Technology, 31, pp. 1253-1267.

Conclusions and Next Steps



DEEP LEARNING
ADVANCED MODELS AND METHODS

We Have Explored:

- Metric Learning Foundations
- Siamese Loss
- Triplet Loss
- Proxy-NCA
- Proxy-Anchor
- ArcFace
- Applications

References

1. **Dr. Lim & Dr. Belongie:** "A Metric Learning Reality Check" (ECCV 2020).
2. **Schroff et al.:** "FaceNet: A Unified Embedding for Face Recognition and Clustering" (CVPR 2015).
3. **Deng et al.:** "ArcFace: Additive Angular Margin Loss for Deep Face Recognition" (CVPR 2019).
4. **Sun et al.:** "Circle Loss: A Unified Perspective of Pair Similarity Optimization" (CVPR 2020).
5. **Teh et al.:** "ProxyNCA++: Revisiting and Revitalizing Proxy Neighborhood Component Analysis" (2020).
6. **Kim et al.:** "Proxy Anchor Loss for Deep Metric Learning" (CVPR 2020).
7. **Ren et al.:** "Towards Improved Proxy-Based Deep Metric Learning via Data-Augmented Domain Adaptation" (arXiv 2024).
8. **Hsieh et al.:** "Collaborative Metric Learning" (WWW 2017).