Siamese Networks

Introduction

- Traditionally, neural network learns to predict multiple classes
  - Poses a problem when need to add/remove new classes to data
  - Have to update neural network and retrain it on whole dataset
  - Deep neural networks need a large volume of data to train on
- For certain problems like face recognition and signature verification, cannot always rely on getting more data
- Applications exist with not enough data for each class
  - Number of classes can also increase exponentially for use cases like employee attendance system
  - Cost of data collection and re-training high each time a new class is added or a new employee joins

Similarity Learning

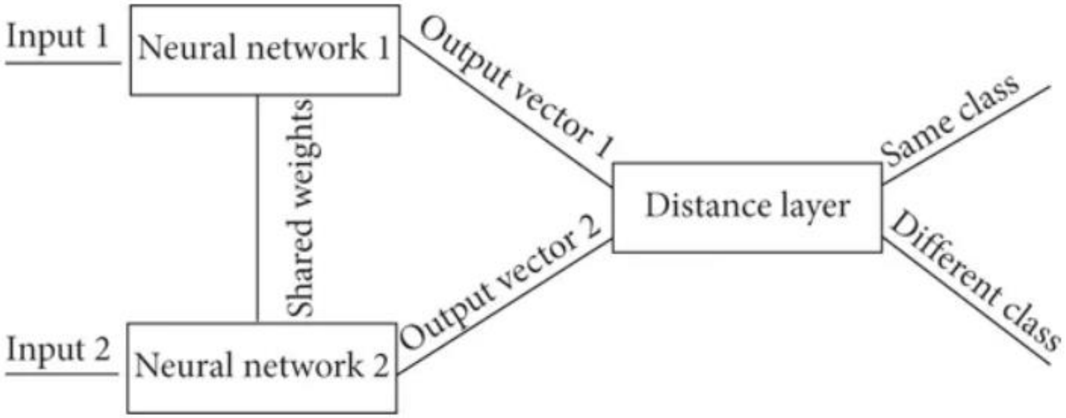
- To solve such tasks use architecture called **Siamese Networks**
  - Uses only a few numbers of images **(one-shot classification)** to get better predictions
  - Uses **similarity learning**
- Similarity learning is a technique of supervised machine learning
  - Goal: make model learn a similarity function that measures how similar two objects are and returns a similarity value
  - A high score returned when objects are similar
  - A low score returned when images or objects are different

Siamese Neural Network

- Siamese Neural Network (SNN) - class of neural network architectures that contain two or more *identical* subnetworks
  - Have same configuration with same parameters and weights
  - Parameter updating is mirrored across both sub-networks
  - Used to find similarity of inputs by comparing its feature vectors
  - Enables to classify new classes of data without retraining network
  - Require only one training example for each class – thus the name **One Shot**

Architecture

- SNN contains two or more identical sub-networks
- Mostly, only train one of *N* (number of subnetworks chosen for solving problem) subnetworks
  - Use same configuration (parameters and weights) for other sub-networks
- SNN used to find similarity of inputs by comparing their feature vectors

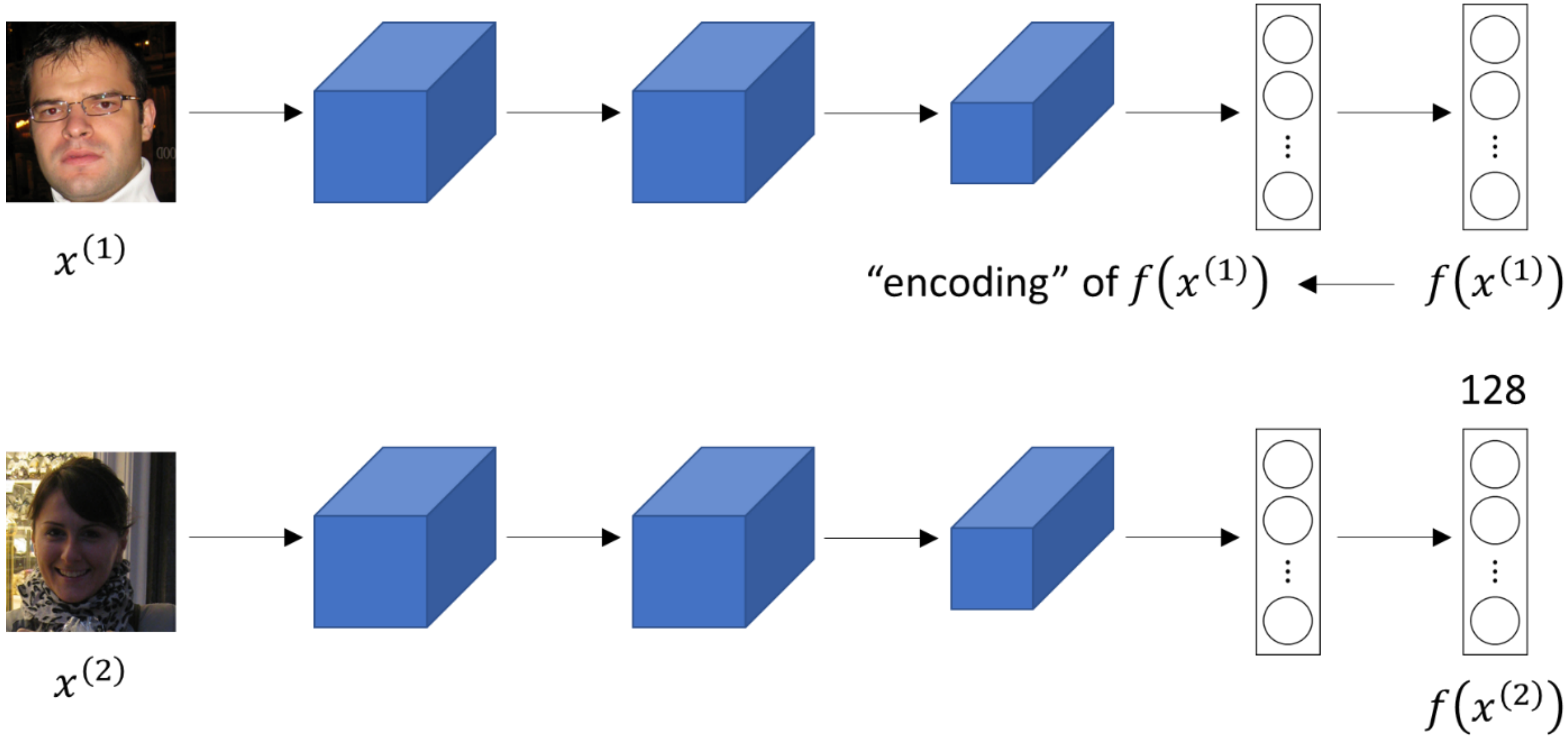


Architecture

- Given two images - want to compare and see if they are similar or dissimilar pairs
  1. First subnetwork takes an image (A) as input
     - Passes through convolutional layers and fully connected layers
     - Gets a vector representation of image
  2. Pass second image (B) through a network
     - Exactly the same with same weights and parameters
  3. Two encodings E(A) and E(B) from respective images
  4. Can compare these two to know how similar the two images are

Architecture

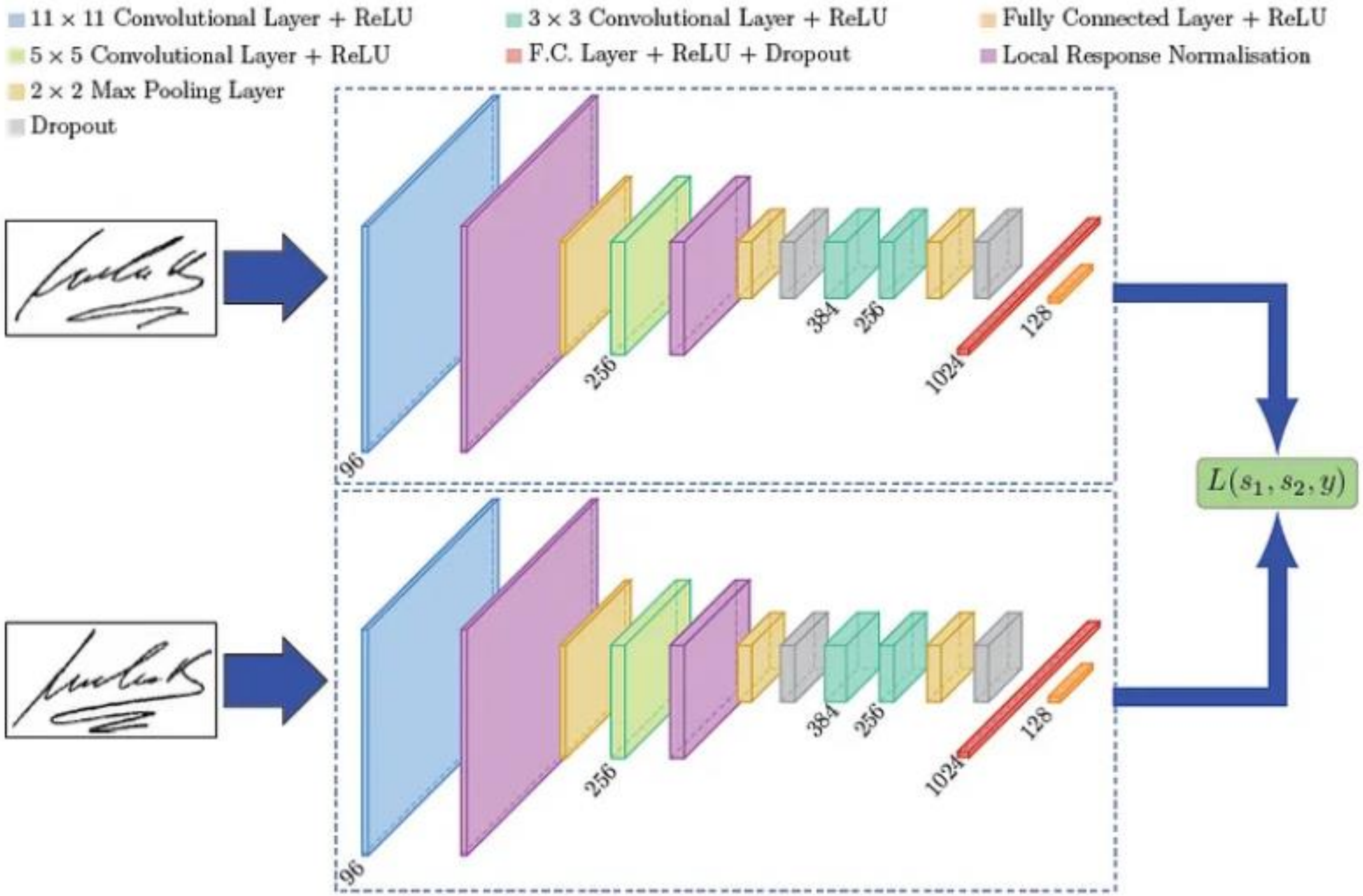
- If images are similar then encodings will also be quite similar
- Measure distance between these two vectors
    - If distance small → vectors are **similar** or of same class
    - If distance large → vectors are **different** from one another



Learn parameters so that:

- If $x^{(i)}$, $x^{(j)}$ are same person, $||f(x^{(i)}) - f(x^{(j)})||^2$ is small
- If $x^{(i)}$, $x^{(j)}$ are different persons, $||f(x^{(i)}) - f(x^{(j)})||^2$ is large

https://datahacker.rs/one-shot-learning-with-siamese-neural-network/



Siamese network used in Signet

Pros and Cons

- Pros
  - **More Robust to Class Imbalance -** Few images per class sufficient for SNN to recognize those images in future with aid of one-shot learning
  - **Learning from Semantic Similarity -** SNN focuses on learning embeddings that place same classes/concepts close together. Hence, can learn semantic similarity
- Cons
  - **Needs More Training Time Than Normal Networks -** Since SNNs involves learning from quadratic pairs they are slower than normal classification type of learning (pointwise learning)
  - **Don’t Output Probabilities -** Since training involves pairwise learning, SNNs do not output probabilities of prediction, only distance from each class

Loss Functions

- Since training SNNs involve pairwise learning, cannot use cross entropy loss
- Two loss functions typically used to train Siamese networks
  - Triplet Loss
  - Contrastive Loss

Triplet Loss

- **Triplet loss -** allows model to map two similar images close and far from dissimilar sample image pairs
- This approach done by using triplet constituting:
  1. ***Anchor Image:*** A sample image
  2. **2.** ***Positive Image:*** Another variation of anchor image
     - Helps SNN learn similarities between the two images
  3. **3.** ***Negative Image:*** Different image from above two similar image pairs
     - Helps model learn dissimilarities with **anchor images**

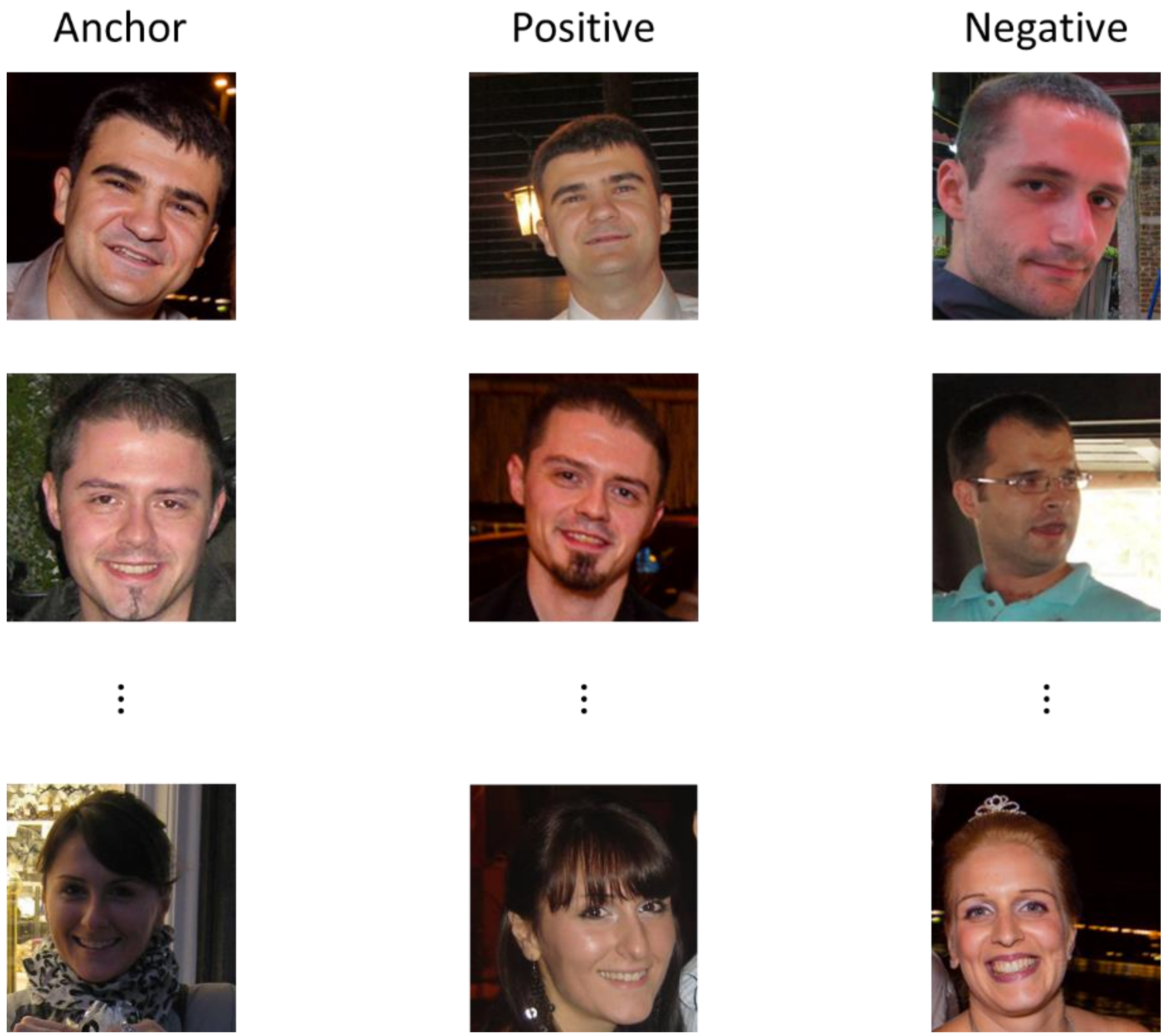
Anchor
A

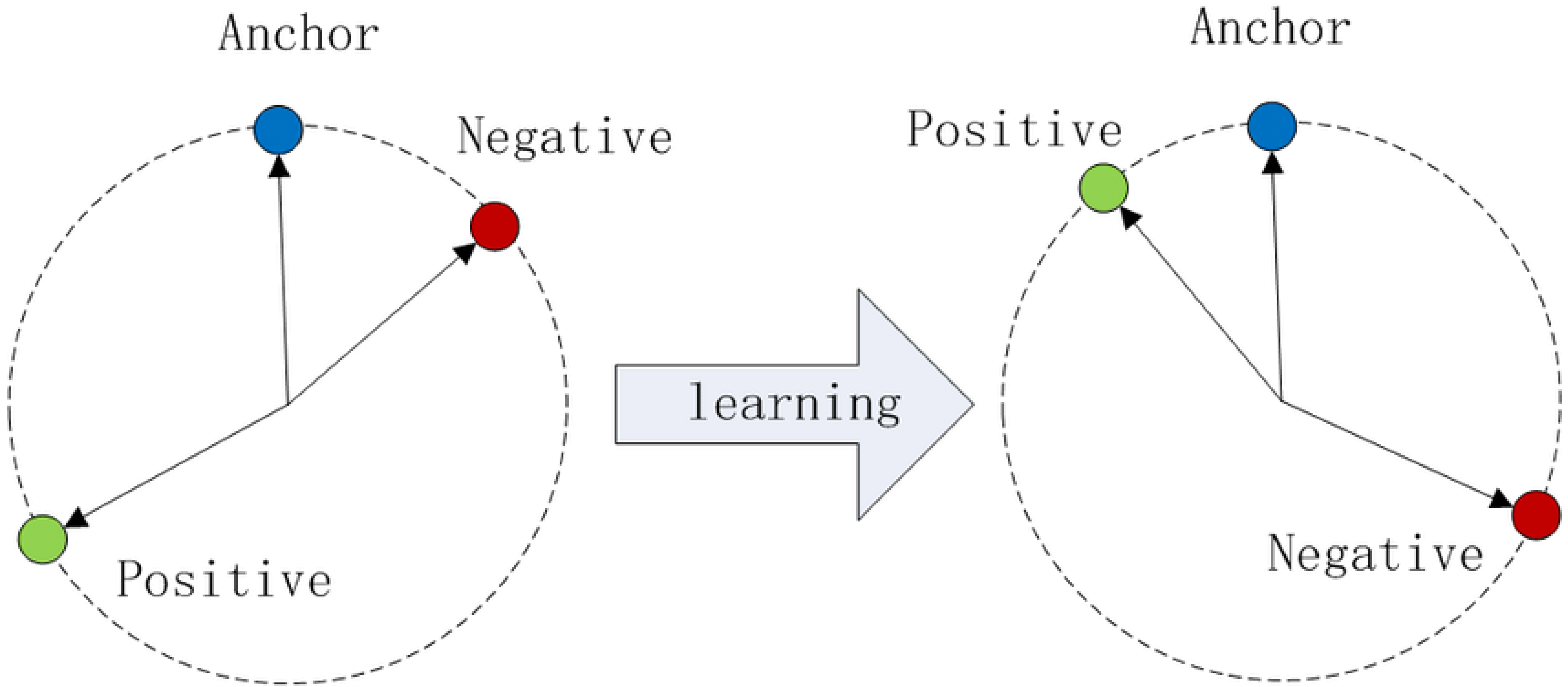
Positive
P

Anchor
A

Negative
N

https://datahacker.rs/siamese-network-triplet-loss/


Anchor
Positive
Negative


Anchor
Negative
Positive
learning
Anchor
Positive
Negative

Triplet Loss

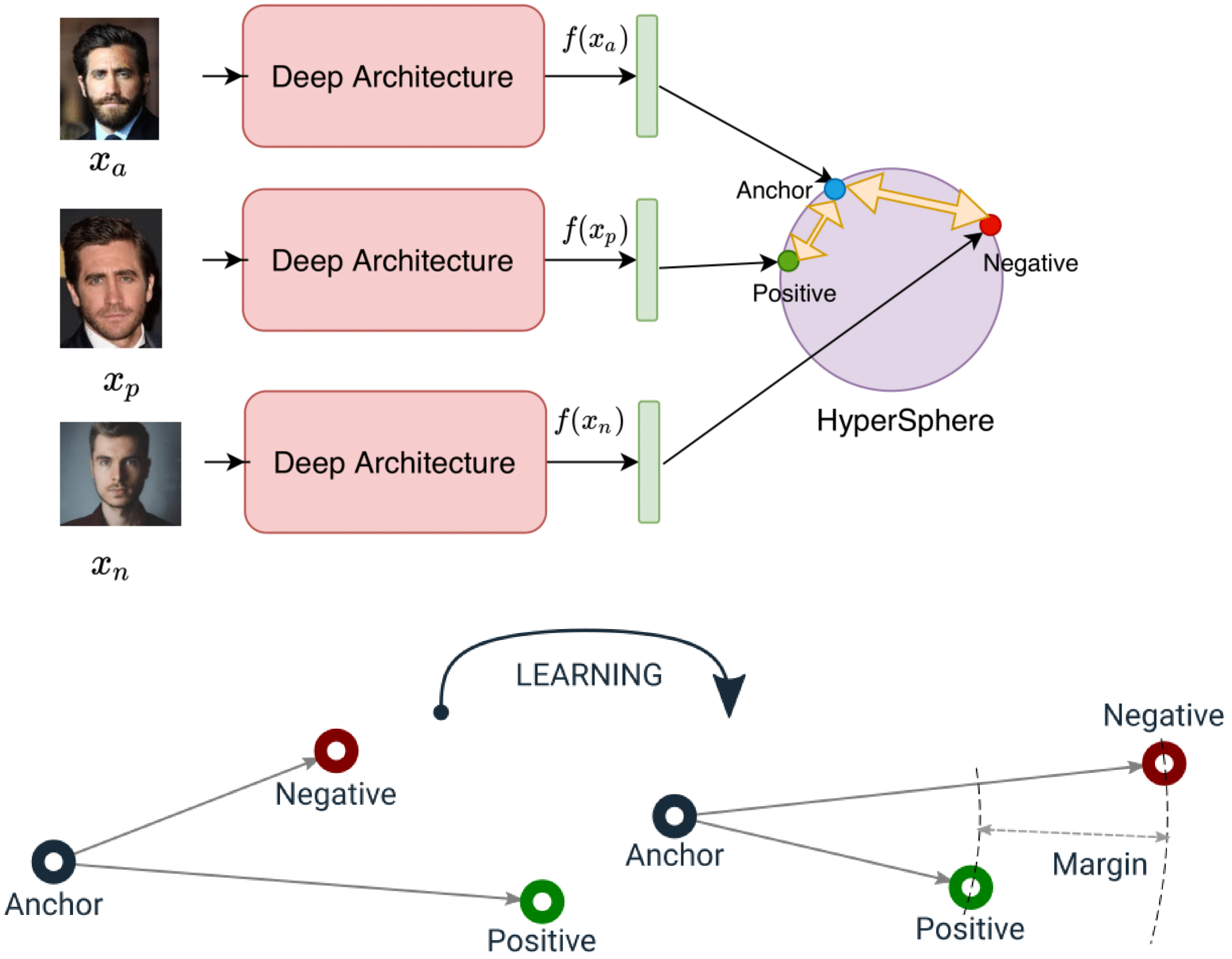
- Distance from *anchor* to *positive* input is minimized, and distance from *anchor* to *negative* input is maximized

$$L(A, P, N) = \max\left(\left||f(A) - f(P)|\right|^2 - \left||f(A) - f(N)|\right|^2 + \alpha, 0\right)$$

  - α - margin term used to stretch distance between similar and dissimilar pairs
  - *f*(*A*), *f*(*P*), *f*(*N*) are feature embeddings for anchor, positive and negative images
- If $\left||f(A) - f(P)|\right|^2 > \left||f(A) - f(N)|\right|^2$ → not desired
  - $L(A, P, N) = \left||f(A) - f(P)|\right|^2 - ||f(A) - f(N)||^2 + \alpha$
- If $\left||f(A) - f(P)|\right|^2 < \left||f(A) - f(N)|\right|^2$ → desired
  - $L(A, P, N) = 0$

Triplet Loss

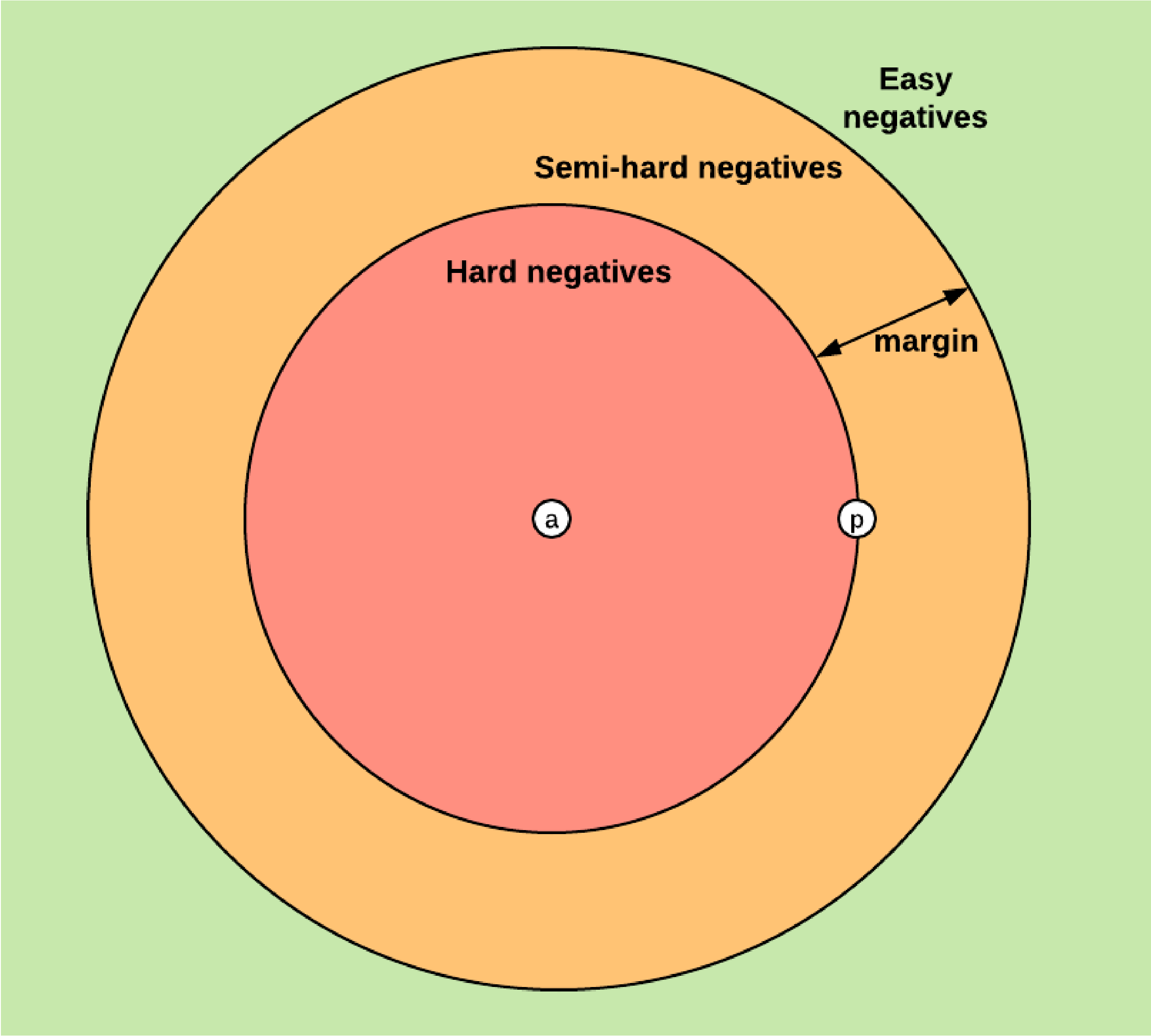
- Similarity or dissimilarity measured by distance between two vectors using **L2 distance and cosine distance**
- During training process, feed an image triplet into model as a single sample
  - Distance between anchor and positive images should be smaller than that between anchor and negative images



Deep Architecture
Deep Architecture
Deep Architecture
$f(x_a)$
$f(x_p)$
$f(x_n)$
x_a
x_p
x_n
Anchor
Positive
Negative
HyperSphere
LEARNING
Anchor
Negative
Positive
Anchor
Negative
Positive
Margin

Triplet Loss

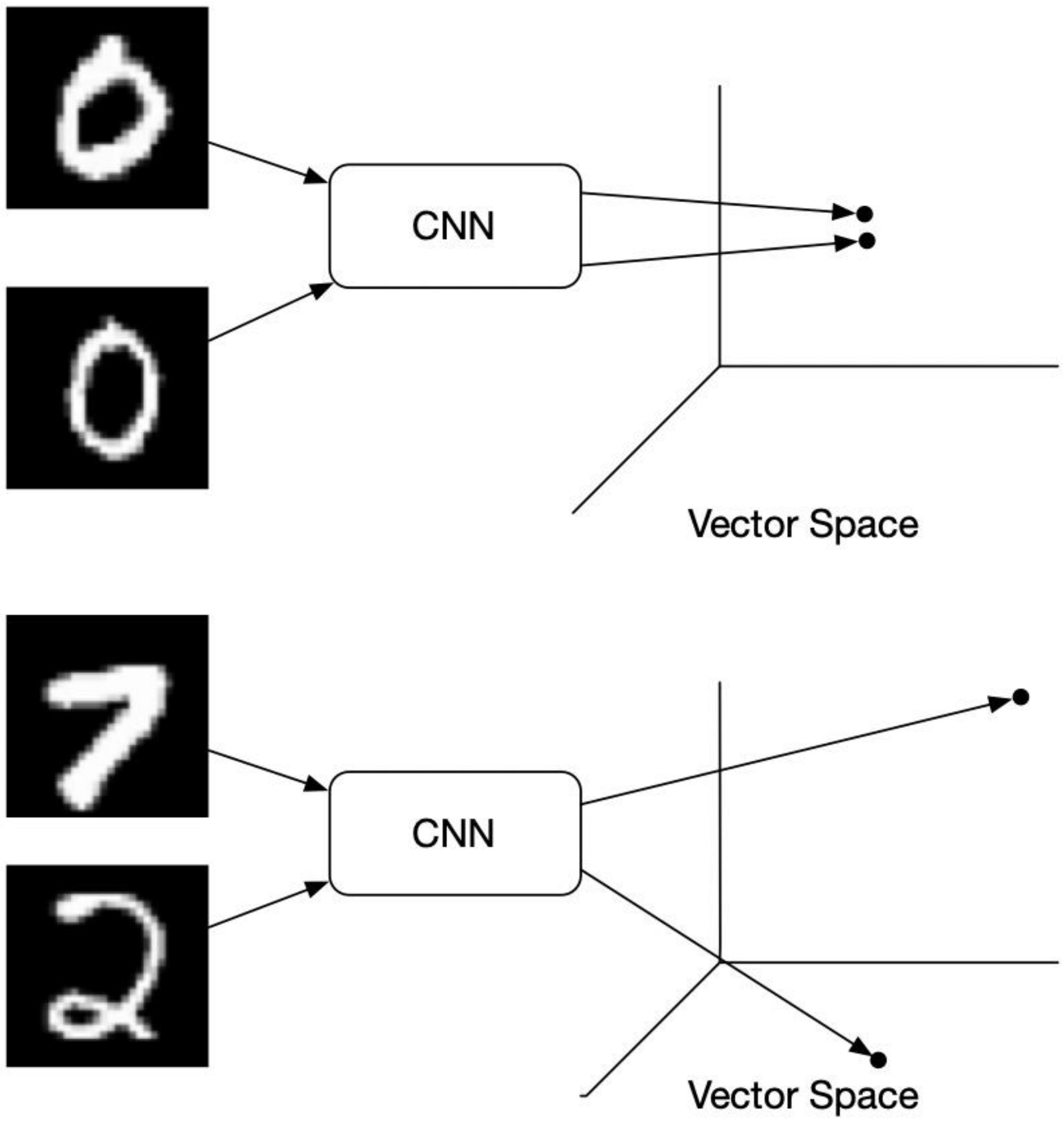
- Based on definition of loss, three categories of triplets:
  - **Easy triplets**: triplets which have a loss of 0, because $f\,(A, P)$ + margin < $f\,(A, N)$
  - **Hard triplets**: triplets where negative is closer to anchor than the positive, i.e. $f\,(A, N) < f\,(A, P)$
  - **Semi-hard triplets**: triplets where negative is not closer to anchor than the positive, but which still have positive loss: $f\,(A, N) < f\,(A, P) + margin$
- Each of these definitions depend on where the negative is, relatively to the anchor and positive
- Choosing what kind of triplets we want to train on will greatly impact metrics



https://omoindrot.github.io/triplet-loss

Contrastive Loss

- Contrastive loss is a distance-based loss
- Loss is low if:
  - Positive samples are encoded to similar (closer) representations
  - Negative examples are encoded to different (farther) representations
- Accomplished by taking distances of vectors and treating resulting distances as prediction probabilities from a typical categorization network
  - Can treat distance of positive example and distances of negative examples as output probabilities and use cross entropy loss


CNN
Vector Space
CNN
Vector Space

Contrastive Loss

- Used to learn embeddings: two similar points have a low Euclidean distance and two dissimilar points have a large Euclidean distance

$$(1-Y)\frac{1}{2}(D_w)^2 + (Y)\frac{1}{2}\{\max(0, m - D_w)\}^2$$

  - D_w is Euclidean distance between outputs of sister networks

$$D_w = \sqrt{\{G_w(X_1) - G_w(X_2)\}^2}$$

  - G_w is output of network for one image
  - Y is either 1 or 0: If first image and second image are from same class, then value of Y is 0, otherwise, Y is 1
  - m is a **margin** value greater than 0 and is the lower bound distance between dissimilar samples
  - Having a margin indicates that dissimilar pairs beyond this margin will not contribute to loss

Contrastive Loss

Loss = $(1 - Y)\frac{1}{2}(D_w)^2 + (Y)\frac{1}{2}\{\max(0, m - D_w)\}^2$

- If images are from same class, Y = 0, Loss = $\frac{1}{2}(D_w)^2$
  - Minimize D_w
  - If D_w is large, loss will be more
  - If D_w is small, loss will be less
- If images are from different class, Y = 1, Loss = $\frac{1}{2}\{\max(0, m - D_w)\}^2$
  - Maximize D_w till some limit m
  - If $D_w < m$, loss will be $(m - D_w)^2$
  - If $D_w > m$, loss will be 0

Triplet vs Contrastive Loss

- **Input**:
  - Triplet loss requires three inputs (anchor, positive, and negative)
  - Contrastive loss requires only two (positive and negative) inputs
- **Distance:**
  - Triplet loss - minimize distance between anchor and positive example while raising the gap between anchor and negative example.
  - Contrastive loss - minimize distance between positive (similar) examples while increasing distance between negative (dissimilar) examples

Triplet vs Contrastive Loss

- **Use cases**:
  - Triplet loss used in problems that aim to acquire a representation space where similar cases are close together, and different examples are far apart—such as facial recognition
  - Contrastive loss commonly employed in applications such as picture categorization
- **Sensitivity**:
  - Margin parameter specifies minimum distance that has to be kept between anchor and positive example and maximum distance that has to be retained between both anchor and negative example - more dependent upon selection of triplet loss
  - Margin parameter has less of an effect on contrastive loss

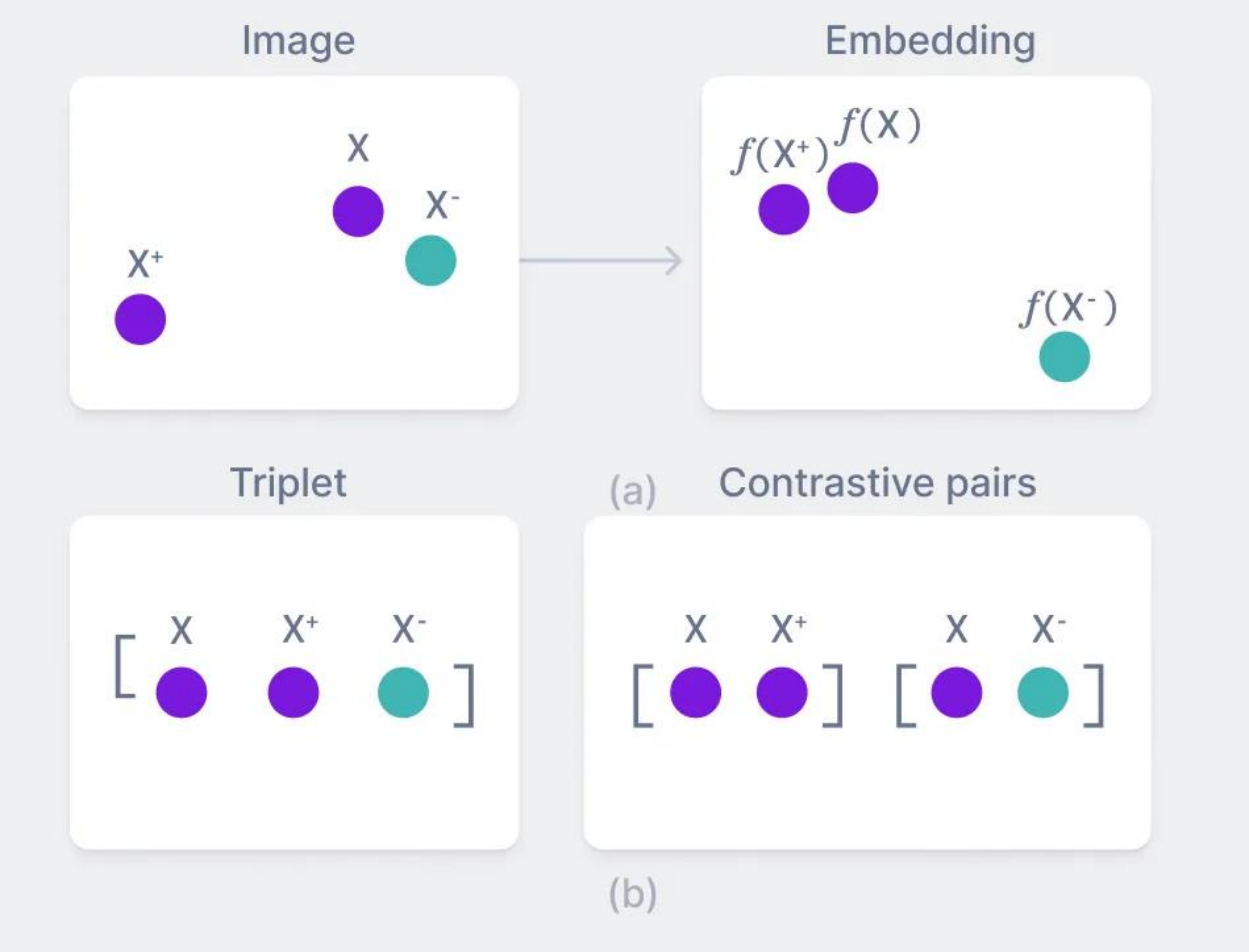
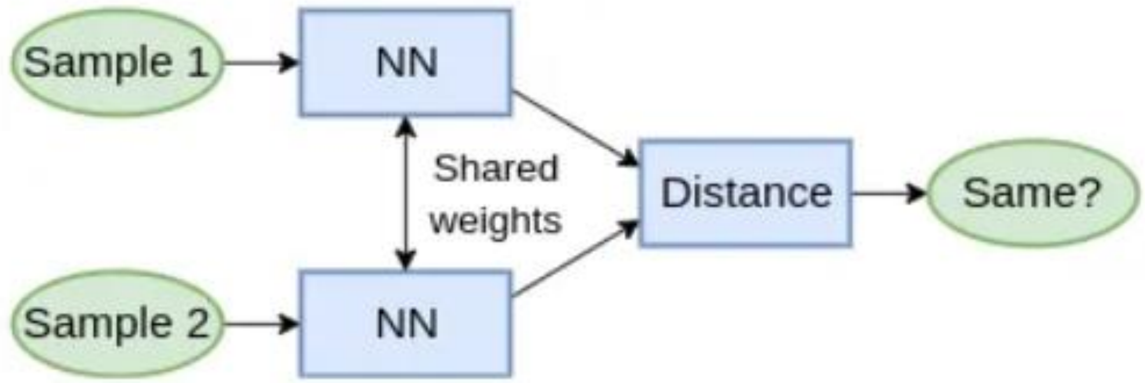


Image
Embedding
X
X-
X+
f(X+)
f(X)
f(X-)
Triplet
(a)
Contrastive pairs
X
X+
X-
X
X+
X
X-
(b)

Employee Attendance System

- Example: build an attendance system for a small organization with only 20 employees, where system has to recognize face of employee

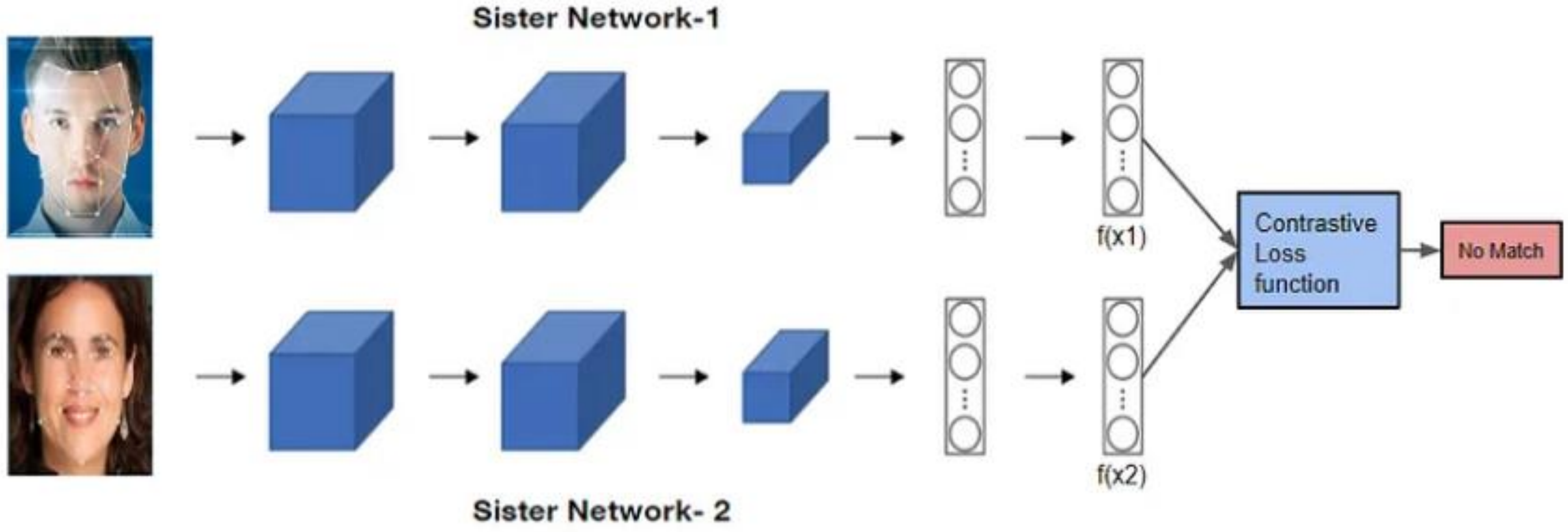


Employee Attendance System

- First problem will be train data images
  - Require a lot of **different** images of each of employees in the organization
- When a new employee joins or leaves organization need to collect data again and **re-train** entire model
  - Not efficient for a scalable system, especially for large organizations like MNCs
- For such a scenario where a scalable system is needed, SNN can be a great solution

Employee Attendance System

- Instead of classifying a test image to one of 20 people in organization:
  - Take a reference image of person as input
  - Generate a similarity score denoting probability that two input images are of same person
- Similarity score lies between 0 and 1 using a sigmoid function
  - Similarity score 0 denotes no similarity
  - Similarity score 1 denotes full similarity
  - Any number between 0 and 1 is interpreted accordingly



Sister Network-1
f(x1)
Contrastive
Loss
function
No Match
f(x2)
Sister Network- 2

Example

- Google CoLab