

Understanding Siamese Networks

Learning through Similarity



Recognizing Your Friend at the Airport

You have one **old photo** of your friend.

You **scan many faces** at the airport.

How do you recognize your friend?

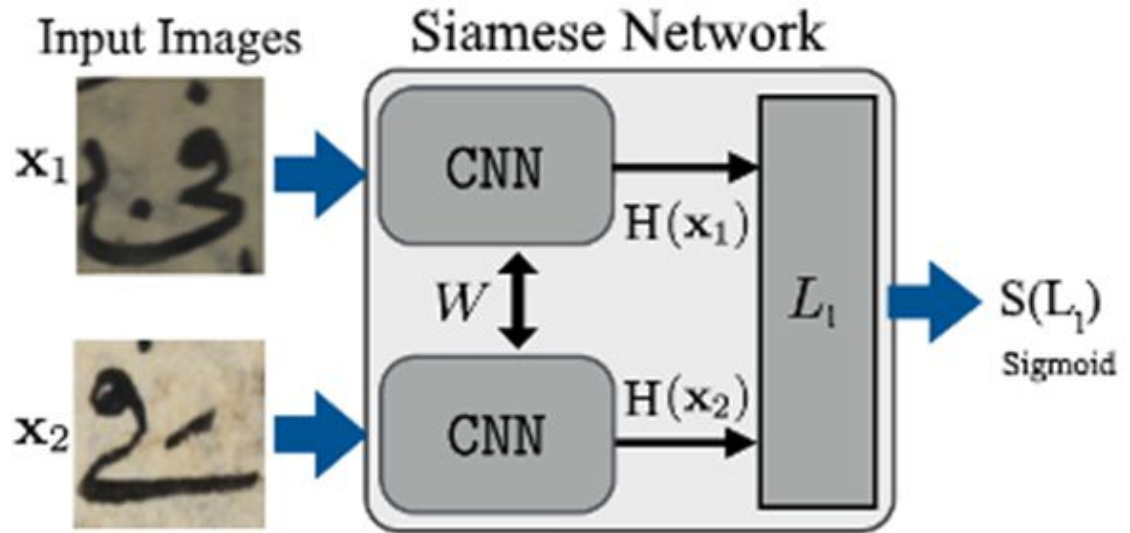


Siamese Network = Twin Learners

Two identical neural networks (shared weights)

Accept two inputs

Output a **similarity score**



Where are Siamese Networks used?

Face verification (e.g., Apple Face ID)

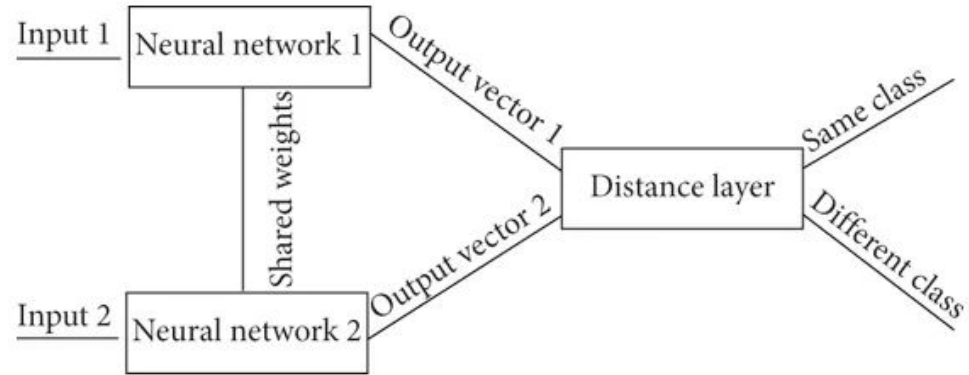
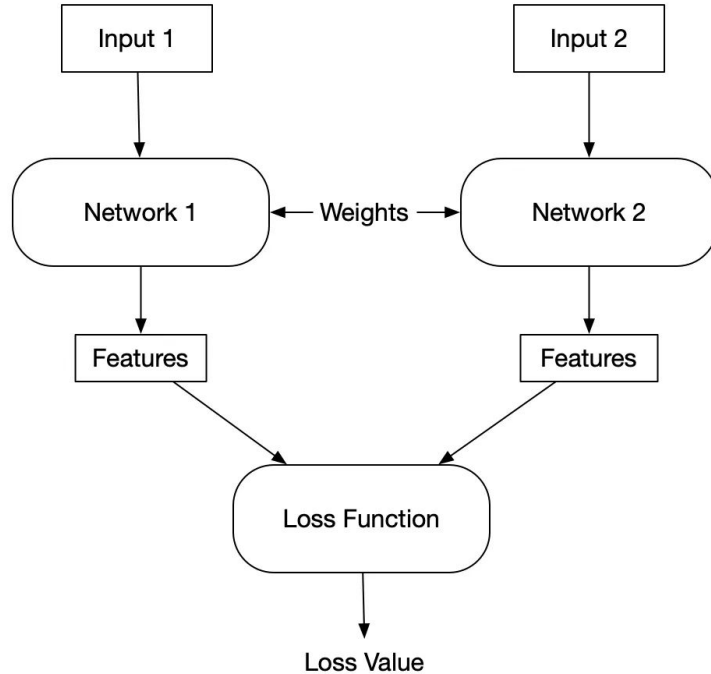
Signature verification at banks

Product recommendation systems

Medical imaging comparisons

Siamese Architecture – A Visual Overview

Generic Siamese Model

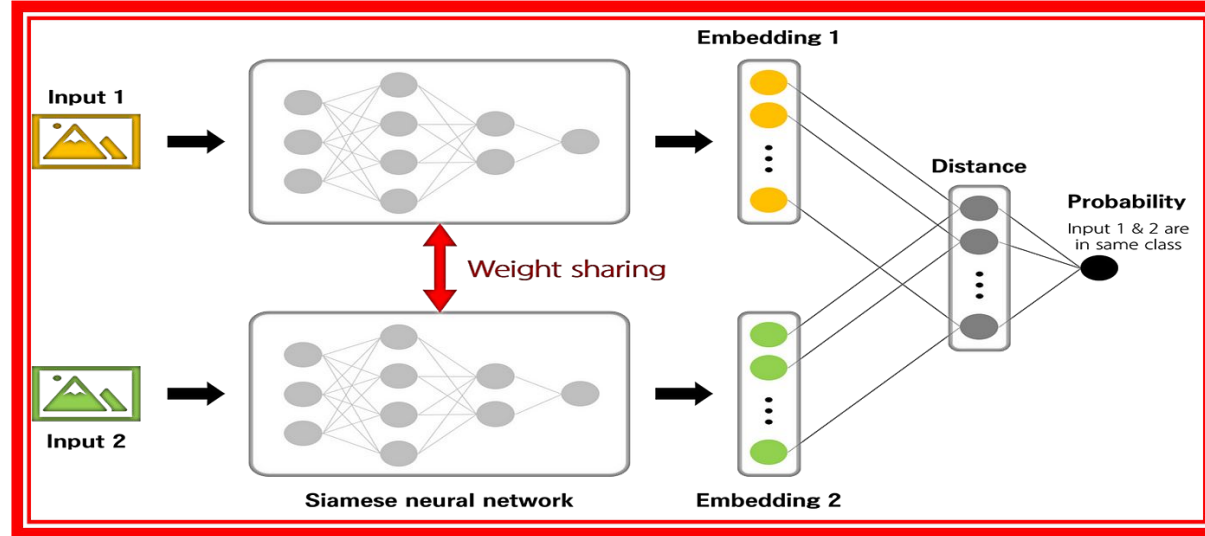


Shared Weights – Why Important?

Ensures **fair** comparison

Reduces model size

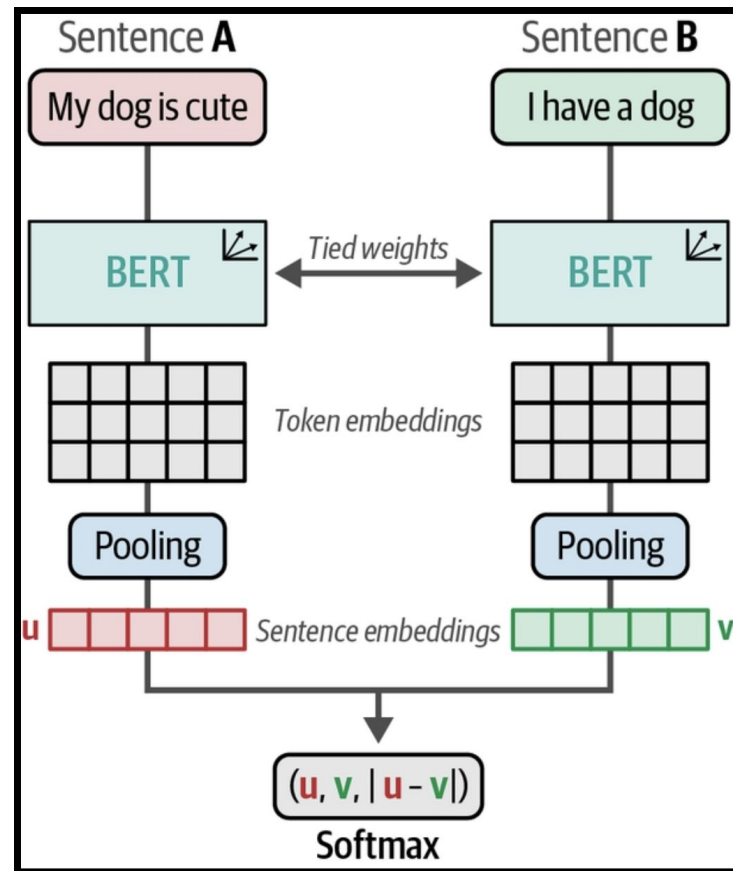
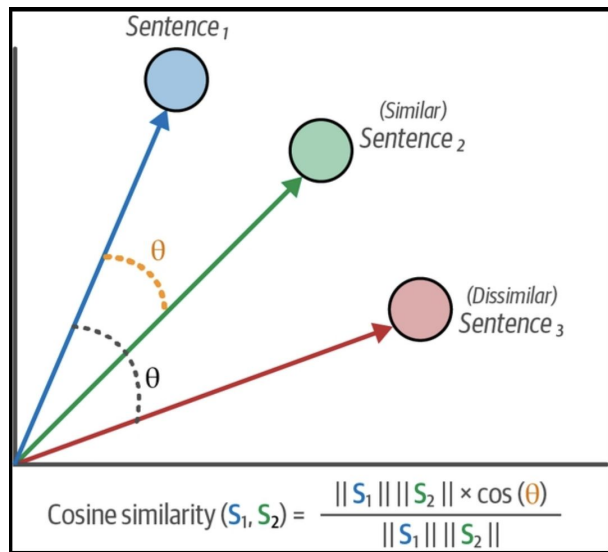
Keeps **feature extraction**
consistent



Embedding Space Intuition - Learning to Group Similar Things Together

Similar inputs → Close embeddings

Different inputs → Far apart embeddings

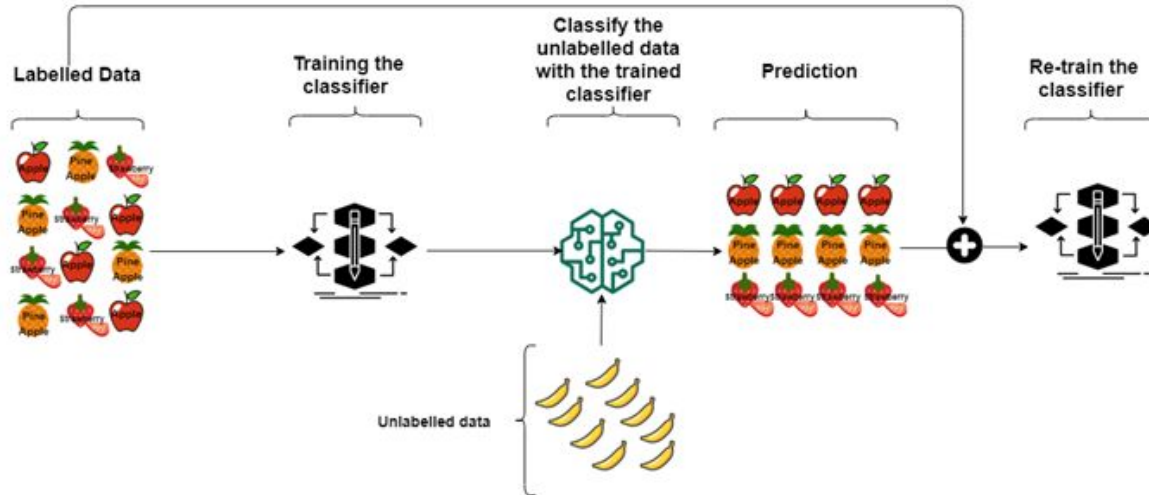


Few-Shot Learning Problem - Why Not Just Use a Classifier?

Traditional CNNs need **many examples**

Few-shot: **only 1-5 examples per class**

Need to generalize quickly



How Siamese Networks Do Few-Shot

Query image **compared** against support set

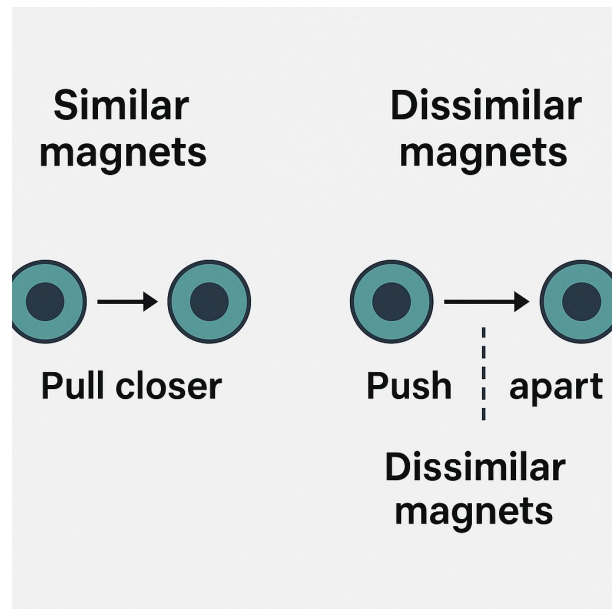
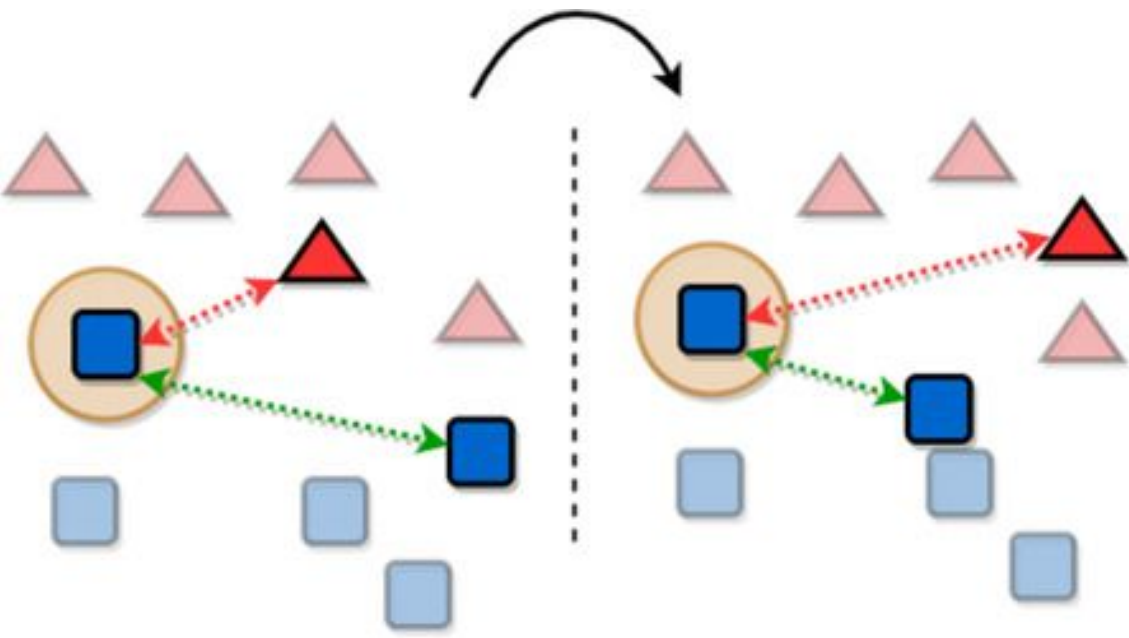
Most similar match is chosen



Contrastive Loss - Pull Together, Push Apart

Similar pairs: minimize embedding distance

Dissimilar pairs: maximize embedding distance (beyond margin)



Contrastive Loss – Formula

$$L = (1 - Y) \cdot D^2 + Y \cdot \max(0, m - D)^2$$

Where:

- $Y = 0$ for similar pairs
- $Y = 1$ for dissimilar pairs
- D is the distance
- m is the margin

Goal First: What are we trying to do?

We want a network to **compare two things** and learn whether they are **similar or dissimilar**.

- Show it pairs of inputs
 - Label them as:
 - **0 = similar**
 - **1 = dissimilar**
- Let the network learn embeddings (i.e., numeric summaries) for each input
- Use a **loss function** to:
 - Pull embeddings together if similar
 - Push them apart if dissimilar

Step-by-Step Intuition: Deriving Contrastive Loss

Step 1: Define a distance between the two embeddings

Let's say:

- The network creates embedding A for input 1
- Embedding B for input 2
- We calculate the distance D between A and B (using Euclidean, cosine, etc.)
- Think of this like the distance between two dots on a 2D map.
- Now let's ask...

Step 2: What should happen for similar pairs (label = 0)?

If two things are similar, we want their distance to be:


 As small as possible (close together)

So, the loss function should:

$$\text{Loss} = D^2 \quad (\text{for similar pairs})$$

This way:

- If the distance is large \rightarrow big penalty
- If distance is small \rightarrow small penalty

 Intuition: You're **rewarding closeness** for similar pairs.

👉 Step 3: What should happen for dissimilar pairs (label = 1)?

✅ At least some margin apart (say, 1 or 2 units)

❌ But we don't want to push them apart forever (no need to separate the moon from the earth!)

So we use:

$$\text{Loss} = \max(0, \text{margin} - D)^2 \quad (\text{for dissimilar pairs})$$

This means:

- If they are **already far enough apart** → loss = 0 (no problem!)
- If they are **too close** → penalize!



Intuition: You're telling the network:

"These are supposed to be different — make sure they don't look too close!"



Step 4: Combine both cases using the label Y

Let's say:

- $Y = 0$ for similar
- $Y = 1$ for dissimilar

Then we write a **single formula** to handle both:

$$\text{Contrastive Loss} = (1 - Y) \cdot D^2 + Y \cdot \max(0, \text{margin} - D)^2$$



Final Intuition Recap

Case	Label (Y)	Desired Action	Loss Term
Similar	0	Pull together	D^2
Dissimilar	1	Push apart (at least by margin)	$(\text{margin} - D)^2$ if $D < \text{margin}$



Step-by-Step Numerical Example

Case 1: Similar Pair ($Y = 0$)

Let's say:

- Embedding distance: $D = 0.5$
- Label $Y = 0$ (similar pair)
- Margin $m = 1$

Plug into formula:

$$L = (1 - 0) \cdot (0.5)^2 + 0 \cdot \max(0, 1 - 0.5)^2$$

$$L = 1 \cdot 0.25 + 0 \cdot 0.25 = 0.25$$

✅ So, the loss is **0.25**.

👉 This means: There's still a little distance between the similar pair, so the model gets a small penalty.

Case 2: Dissimilar Pair, Too Close ($Y = 1$)

Let's say:

- $D = 0.3$ (too close!)
- $Y = 1$
- $m = 1$

Now:

$$L = (1 - 1) \cdot (0.3)^2 + 1 \cdot \max(0, 1 - 0.3)^2$$

$$L = 0 + 1 \cdot (0.7)^2 = 0.49$$

✅ Loss = 0.49 → Higher penalty

👉 Because the dissimilar pair is not far enough apart (they're only 0.3 units away instead of at least 1).

Case 3: Dissimilar Pair, Far Enough ($Y = 1$)

Let's say:

- $D = 1.5$
- $Y = 1$
- $m = 1$

Now:

$$L = (1 - 1) \cdot (1.5)^2 + 1 \cdot \max(0, 1 - 1.5)^2$$

$$L = 0 + 1 \cdot (0)^2 = 0$$

✅ Loss = 0

👉 No penalty! The dissimilar pair is far enough apart — they're already more than the required margin.

Summary

Case	Y (Label)	Distance (D)	Margin (m)	Loss	Explanation
Similar Pair (close)	0	0.5	1	0.25	Still a bit apart, small penalty
Dissimilar Pair (too close)	1	0.3	1	0.49	Not far enough, medium penalty
Dissimilar Pair (far enough)	1	1.5	1	0	Already separated, no penalty

Burger or Pizza?



Anchor



A

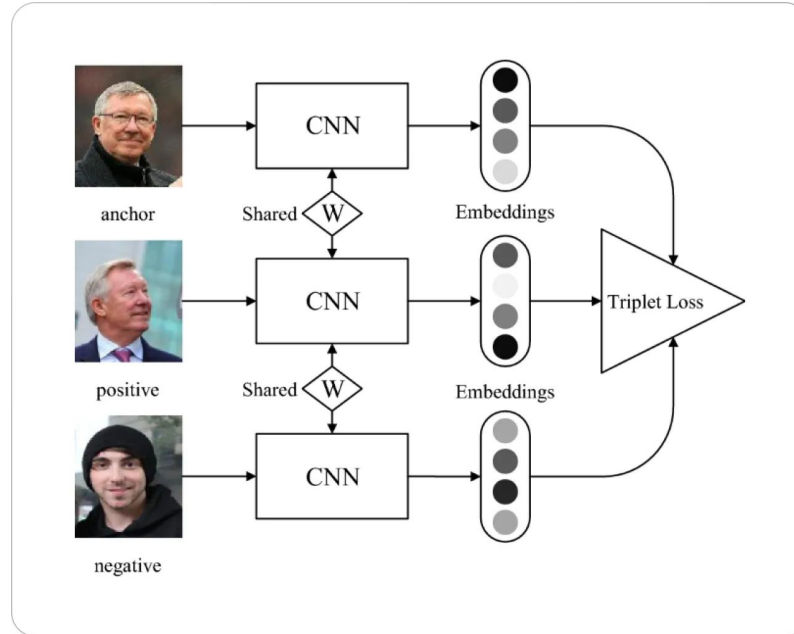


B

Triplet Loss – What's New?

Triplet: Anchor, Positive, Negative

Anchor should be closer to Positive than Negative by margin



Triplet Loss – Formula

$$L = \max(0, D_{ap} - D_{an} + m)$$

Diagram: Anchor → Positive (closer), Anchor → Negative (farther)

What It Does

If the positive is closer than the negative by at least the margin \rightarrow loss = 0
(perfect!)

If not \rightarrow the model gets a penalty, and updates itself

 Desirable case:

$$D_{AP} + \text{margin} < D_{AN} \Rightarrow \text{Loss} = 0$$

 Undesirable case:

$$D_{AN} \text{ is too close} \Rightarrow \text{Loss} > 0$$



Why Not Just Use Pairs?



Simple Numerical Example

Let's say:

- $D_{AP} = 0.5$
- $D_{AN} = 0.6$
- Margin = 0.3

Triplet Loss:

$$L = \max(0, 0.5 - 0.6 + 0.3) = \max(0, 0.2) = 0.2$$

👉 There's a small penalty because the gap isn't big enough.

Now say:

- $D_{AP} = 0.5$
- $D_{AN} = 0.9$

$$L = \max(0, 0.5 - 0.9 + 0.3) = \max(0, -0.1) = 0$$

✅ Perfect! No loss.

Siamese vs Triplet Loss - Which one to use?

Siamese (Contrastive)	Triplet Loss
Simple pairs	Triples (3 inputs)
Easier to train	More stable embeddings
Pairwise loss	Ranking loss

Quick Quiz

What is the main task of a Siamese Network?

Why do the twin networks share weights?

How does contrastive loss encourage learning?

What's the difference between contrastive and triplet loss?