

Lab Task

Objective

In this lab session, you will deepen your understanding of Siamese Networks by visualizing data pairs, experimenting with different optimization algorithms, and upgrading the model architecture from Dense Neural Networks (DNN) to Convolutional Neural Networks (CNN).

Task 1: Visualization (Data & Predictions)

Understanding the input and output of a Siamese Network is crucial.

1. **Visualize Data Pairs:** Write a code snippet to display **3 pairs** from the training set. Show the images and their labels (1 for similar, 0 for different).
2. **Visualize Model Predictions:**
 - Pick two random pairs from the test set (one similar, one different).
 - Pass them through your trained model to calculate the **distance**.
 - Display the images with the **Calculated Distance** and the **Model's Prediction** (Similar/Different) based on a threshold (e.g., 0.5).

Task 2: Impact of Optimizers

The choice of optimizer significantly affects convergence speed and final accuracy.

1. Using the baseline Siamese Network code (Contrastive or Triplet Loss), retrain the model **two separate times** using the following optimizers:
 - **RMSprop**
 - **SGD (Mini-Batch Gradient Descent)**
2. Compare these results with the **Adam** optimizer results.
3. **Deliverable:** A brief comment on which optimizer converged faster and achieved higher accuracy.

Task 3: From DNN to CNN (Architecture Upgrade)

Dense layers (Flattened input) lose spatial information. CNNs are superior for image tasks.

1. Modify the **Base Network** architecture in your code. Replace the Flatten and Dense layers with **Convolutional Layers (Conv2D)** followed by MaxPooling2D.

2. **Important:** You must reshape your MNIST data from (N, 784) to (N, 28, 28, 1) to fit the CNN input.
 3. Train this new architecture using either **Contrastive Loss** or **Triplet Loss**.
 4. **Deliverable:** Report the final accuracy. Did using CNN improve the performance compared to the Dense network?
-

Hint for Task 3:

Python

Reshaping data for CNN

```
x_train = x_train.reshape(-1, 28, 28, 1)
```

Input shape for the model becomes (28, 28, 1)