Exploring Face Similarity with Siamese Networks

Understanding Siamese Networks for Face Similarity Detection

In this project, I implemented a **Siamese Network** to detect face similarity by learning to differentiate between similar and dissimilar face pairs. This architecture is particularly suited for **metric learning** tasks where the goal is to measure similarity or dissimilarity between inputs, such as face verification or one-shot learning. The focus was both on theoretical understanding and practical implementation, covering key aspects such as data preprocessing, model architecture, training, and evaluation.

Siamese Network Architecture

A **Siamese Network** consists of two identical subnetworks f_θ that share the same weights. Given two input face images x_1 and x_2, the network computes their feature embeddings $f_\theta(x_1)$ and $f_\theta(x_2)$. These embeddings are compared using a distance function, typically the **Euclidean distance**:

$$d(x_1, x_2) = \|f_\theta(x_1) - f_\theta(x_2)\|_2$$

The goal of the network is to ensure that the distance d is small for similar pairs and large for dissimilar pairs.

Data Preprocessing and Pair Generation

The dataset preparation involved creating pairs of images:

- **Positive pairs**: Images of the same person.
- **Negative pairs**: Images of different people.

Each image was resized, normalized, and converted into tensors for compatibility with the model. The dataset was split into training, validation, and test sets to ensure proper evaluation.

Loss Function: Contrastive Loss

The Siamese Network was trained using the **contrastive loss function**, which encourages similar pairs to have a small distance and dissimilar pairs to have a large distance. The contrastive loss L is defined as:

$$L(y, d) = yd^2 + (1 - y)\max(0, m - d)^2$$

where:

- y is the label ($y = 1$ for similar pairs, $y = 0$ for dissimilar pairs).
- d is the Euclidean distance between the embeddings.
- m is a margin hyperparameter that defines the minimum distance for dissimilar pairs.

This loss function penalizes the network when similar pairs have a large distance or when dissimilar pairs have a distance smaller than the margin m.

Model Training

During training, the network minimizes the contrastive loss using **backpropagation** and an optimizer such as **Adam**. Key parameters included:

- **Learning rate**: $\eta = 0.001$
- **Batch size**: 32
- **Epochs**: 20

The shared weights of the subnetworks ensure that both inputs are processed identically, enabling the model to learn a consistent feature space.

Model Evaluation

After training, the model was evaluated on the test set by computing the distances between the embeddings of face pairs. A **threshold** τ was determined to classify pairs:

$$\text{Prediction} = \begin{cases} \text{Similar} & \text{if } d \leq \tau \\ \text{Dissimilar} & \text{if } d > \tau \end{cases}$$

Performance was measured using metrics such as:

- **Accuracy**
- **Precision**

- **Recall**
- **F1 Score**

The **Receiver Operating Characteristic (ROC) curve** was also used to visualize the trade-off between true positive and false positive rates.

Theoretical Insights and Applications

Implementing a Siamese Network provided deeper insights into how **metric learning** works. By learning an embedding space where similar faces cluster closely and dissimilar faces are far apart, the model can generalize well to unseen data. This approach is particularly effective for:

- **Face verification** systems (e.g., login authentication).
- **One-shot learning** tasks where only a few examples per class are available.
- **Duplicate detection** in datasets.

Understanding the theoretical underpinnings of Siamese Networks opens the door to more advanced models like **Triplet Networks** and **Quadruplet Networks**.

Conclusion

This project was a valuable exploration of deep learning for face similarity detection, combining mathematical rigor with practical implementation.