**Submitted by -** *Shivani Sharma*

**SIAMESE NETWORK**

A Siamese network is a type of neural network architecture designed for tasks that involve measuring similarity or dissimilarity between pairs of data points. It's commonly used in applications like face recognition, signature verification, and similarity-based recommendation systems.

The name "Siamese network" is derived from the concept of "Siamese twins," who are identical or very similar twins. In a Siamese network, two identical subnetworks (or twin networks) share the same architecture and weights. These subnetworks are used to process two input data points simultaneously, and the goal is to learn embeddings (feature representations) for each input that are suitable for measuring similarity or dissimilarity.

**TRIPLET LOSS FUNCTION -**

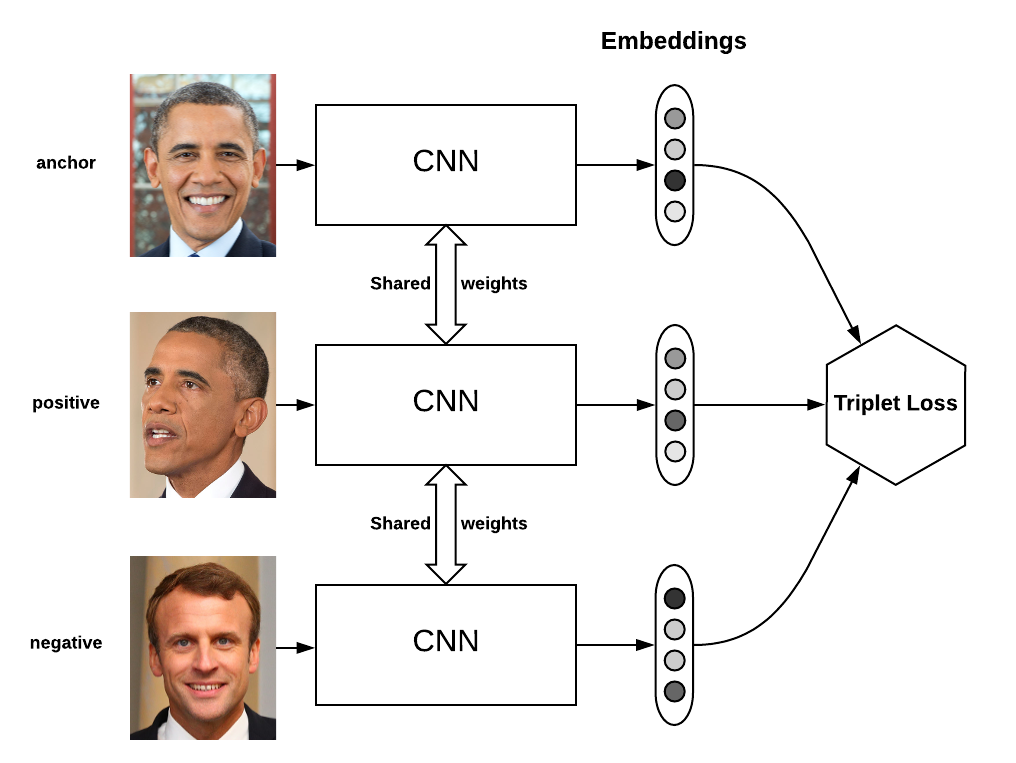
The triplet loss function is a key component of Siamese networks and is used to train these networks for tasks involving similarity or dissimilarity comparisons between data points. It encourages the network to embed similar data points close together in the embedding space while pushing dissimilar data points apart. It's commonly used in applications like face recognition and image retrieval.

The **mathematical formulation** of the triplet loss function:

Let:

* A : be the anchor data point.
* P : be the positive data point (similar to the anchor).
* N : be the negative data point (dissimilar to the anchor
* f(x): be the embedding function that maps input data x to a feature vector in an embedding space.

The triplet loss function L can be defined as:

L(A, P, N) = max(‖f(A) - f(P)‖² - ‖f(A) - f(N)‖² + margin, 0)

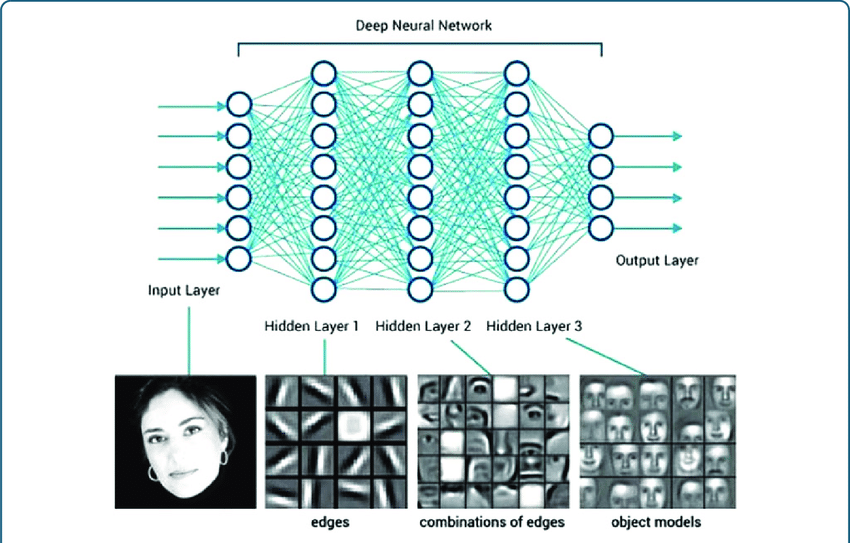
**FACE RECOGNITION:**

Face recognition is a biometric technology that involves identifying or verifying an individual's identity by analysing and comparing their facial features. It is a computer vision and machine learning application used to automatically detect and recognize faces in images or videos.

Face recognition is a technology that enables the automatic identification, verification, or authentication of a person based on their facial characteristics. It involves capturing an individual's facial image, extracting relevant features from that image, and comparing these features to a database of known individuals to determine if a match exists.

**Key components of face recognition include:**

* Face Detection: The process of locating and identifying faces within an image or video frame. It involves identifying the regions of an image that contain faces.
* Feature Extraction: Extracting key facial features, such as the position of the eyes, nose, and mouth, as well as the overall facial structure. These features are used to create a unique representation of an individual's face.
* Face Representation: Transforming the extracted facial features into a numerical representation or feature vector that can be used for comparison and analysis.
* Database Matching: Comparing the facial feature vector of the person in question to a database of known individuals. This step determines whether there is a match or similarity between the input face and the stored reference faces.
* Decision-Making: Making a decision based on the comparison results, which can involve recognizing the person if a match is found or verifying their identity.



**Applications of face recognition technology include:**

* Access Control: Granting or denying access to secure areas or devices based on facial recognition, commonly used in security systems.
* Authentication: Verifying a person's identity for various applications, such as unlocking smartphones or authorising financial transactions.
* Surveillance: Monitoring and identifying individuals in public places or within large crowds for security and law enforcement purposes.
* Attendance Tracking: Automating attendance records by recognizing individuals as they enter or exit a location.
* Emotion Analysis: Analysing facial expressions to determine emotional states for applications in human-computer interaction and market research.

**FACE RECOGNITION USING SIAMESE NETWORK :**

**Importing Libraries:**

It is important to import all libraries. Libraries used in this project are:

[Libraries included](https://colab.research.google.com/drive/1In3-KDlLhXP5Ykfm1WT5bRTiPVx8iqXo#scrollTo=HDa6aoQi3w3P&line=3&uniqifier=1)

1. **matplotlib.pyplot:**

Matplotlib is a widely-used Python library for creating static, animated, and interactive visualisations in Python. `pyplot` is a subpackage within Matplotlib that provides a MATLAB-like interface for creating plots and charts.It's used for plotting and visualising data. For example, you can use it to display images, plot training curves, or visualise any data as needed.

1. **`numpy`:**

NumPy is a fundamental library for numerical computing in Python. NumPy is used for various numerical operations, array manipulations, and mathematical computations. In this code, it's likely used for handling and processing data.

1. **`os`:**

The `os` module provides a way to use operating system-dependent functionality, such as interacting with the file system, working with directories, and managing file paths.

1. **`random`:**

The `random` module is used for generating random numbers and performing randomization operations. It's likely used for randomization tasks, such as shuffling data or generating random indices.

1. **`cv2` (OpenCV):**

OpenCV (Open Source Computer Vision Library) is a popular computer vision library that provides tools and functions for image and video analysis, including image processing, computer vision tasks, and more.

1. **`tensorflow`:**

TensorFlow is an open-source machine learning framework developed by Google. It's used for building and training machine learning and deep learning models.It's used for defining neural network architectures, optimising models, and managing training and evaluation processes.

1. **`pathlib.Path`:**

The `pathlib` module provides an object-oriented interface for working with file system paths. The `Path` class allows you to manipulate file paths in a more intuitive way.

1. **`tensorflow.keras`:**

Keras is a high-level neural networks API that runs on top of TensorFlow. It provides a user-friendly interface for building and training deep learning models.

1. **`sklearn.utils`:**

Scikit-learn is a machine learning library that provides tools for data analysis and modelling.In this code, `sklearn.utils` is used for shuffling data. It's often used to randomise the order of data samples to prevent any bias during training.

1. **`tensorflow.keras.applications.resnet`:**

TensorFlow's Keras applications include pre-trained models for various computer vision tasks. `resnet` is a submodule that provides pre-trained ResNet models, which are popular for image classification and feature extraction.

1. **`tensorflow.keras.layers`:**

This module from TensorFlow's Keras API provides a wide range of layers for building neural network architectures. These layers include convolutional layers, dense layers, recurrent layers, and more.It's used to define the layers of the neural network architecture, specifying the structure of the model, such as convolutional layers for feature extraction and dense layers for classification or regression tasks.

1. **`tensorflow.keras.losses`:**

TensorFlow's Keras API includes a `losses` module that provides various loss functions used for training neural networks. Loss functions measure the difference between predicted values and true target values and are used to update the network's weights during training.

1. **`tensorflow.keras.optimizers`**:

The `optimizers` module in TensorFlow's Keras provides various optimization algorithms used to minimise the loss function during training. These algorithms include SGD (Stochastic Gradient Descent), Adam, RMSprop, and more.

1. `**tensorflow.keras.metrics`:**

TensorFlow's Keras API includes a `metrics` module that provides predefined metrics for evaluating the performance of trained models. Metrics are used to measure the accuracy, precision, recall, etc., of the model on the validation or test data.

1. **`tensorflow.keras.Model`:**

The `Model` class in TensorFlow's Keras API serves as the base class for defining neural network models. It allows you to specify the model's architecture and training configuration.It's used to create and configure the neural network model by specifying the input and output layers, as well as any custom layers or operations that are part of the model.

**Working on Dataset:**

1. **Creating a Utility Function:**

[**Utility Function**](https://colab.research.google.com/drive/1In3-KDlLhXP5Ykfm1WT5bRTiPVx8iqXo#scrollTo=wR68cuz64giK&line=11&uniqifier=1)

‘generate\_image\_pairs’, is designed to generate pairs of images and their corresponding labels for use in training a Siamese network or a similar model for similarity-based tasks.

The two pairs that are generated are:

* Positive Pairs (Pairs with Label 1)
* Negative Pairs(Pairs with Label 0)

1. **Preaparing Dataset:**

[**Data Preparation**](https://colab.research.google.com/drive/1In3-KDlLhXP5Ykfm1WT5bRTiPVx8iqXo#scrollTo=3fSVQuBoVHLy&line=1&uniqifier=1)

After executing these codes, you should have a dataset of shuffled image pairs (images\_dataset) and their corresponding labels (labels\_dataset) ready for training a Siamese network or a similar model for tasks like face recognition or similarity-based learning.

**Visualising the Data:**

[Data Visualisation](https://colab.research.google.com/drive/1In3-KDlLhXP5Ykfm1WT5bRTiPVx8iqXo#scrollTo=R_asQLviWW9r&line=1&uniqifier=1)

This code will display visual examples of positive image pairs, helping in inspecting and understanding the data used for training or evaluation.

**Network Architecture:**

1. **Embedded Layers:**

[Embedded Layers](https://colab.research.google.com/drive/1In3-KDlLhXP5Ykfm1WT5bRTiPVx8iqXo#scrollTo=xrkb1AEkWyL8&line=1&uniqifier=1)

The provided code defines a neural network model called "Embedding" using the Keras API in TensorFlow. This model is designed for feature extraction or embedding of input images. This model is designed to take grayscale images of size 64x64 as input and produce embeddings of size 1024 as output. The convolutional layers learn hierarchical features from the input images, and the fully connected layers further process these features before producing the final embeddings.

You can use this "Embedding" model as part of a larger architecture for various tasks, such as image classification, face recognition, or similarity-based learning, by connecting it to other layers as needed.

1. **Siamese Network:**

[Siamese Network](https://colab.research.google.com/drive/1In3-KDlLhXP5Ykfm1WT5bRTiPVx8iqXo#scrollTo=-IxLaFr9XP0B&line=1&uniqifier=1)

1.Distance Layer (`DistanceLayer`):

- This is a custom Keras layer that calculates the Euclidean distance between two embeddings. It's used to compute the distance between the

anchor and comparison embeddings.

- The `call` method takes two input embeddings, computes the squared sum of differences, takes the square root, and returns the result.

2. Siamese Model (`siamese\_model`):

- This section defines the architecture of the Siamese network.

- It defines two input layers (`anchor\_input` and `compare\_input`) for the anchor and comparison images. These inputs have the same shape as the target image shape.

- The `embedding` model, defined previously, is used to embed the anchor and comparison images.

- The `DistanceLayer` is used to calculate the distance between the anchor and comparison embeddings.

- Finally, a dense layer with a single unit and sigmoid activation is used to produce an output, which represents the similarity score between the anchor and comparison images.

**Training:**

[Training](https://colab.research.google.com/drive/1In3-KDlLhXP5Ykfm1WT5bRTiPVx8iqXo#scrollTo=ASbrJ33fYIEA&line=1&uniqifier=1)

The provided code compiles and trains the Siamese network using binary cross-entropy loss and the Adam optimizer.The history variable will contain information about the training process, including training and validation loss and accuracy for each epoch. You can use this information to visualise the training progress and evaluate the model's performance.Then there is training plot in which trained data is plotted using plot.

**Testing:**

[Testing](https://colab.research.google.com/drive/1In3-KDlLhXP5Ykfm1WT5bRTiPVx8iqXo#scrollTo=tNv29COJaOVH&line=1&uniqifier=1)

This code essentially selects a test image, generates a set of comparison images, visualises both the test and comparison images, and then uses your Siamese network to predict similarity scores between the test image and each comparison image.