Painter Classification

By Lillian Mansour

ID- 211623806

The main goal was to examine pairs of paintings and predict whether the paintings are by the same artist.

Guide to Running Assignment Files:

- Start by downloading the four files included in the assignment.
- Change the paths to fit your needs – in the sixth cell.
- Run all.

Data Processing, Model Training, and Evaluation

Step 1-

Creating Dataset:

- Initial Splitting
  - Start with a dataset from the "all_data_info" CSV file.
  - Split the dataset into train and test datasets based on the value in the "in_train" column. Place images with the value TRUE in the train dataset and images with the value FALSE in the test dataset.
- Validation Dataset Creation
  - From the train dataset, identify all unique artists.
  - From the test dataset, identify all unique artists.
  - Create the validation dataset:
    - ✓ Split the artists from the train dataset into two groups: one group containing 80% of the artists and another containing 20%.
    - ✓ Use the 20% group of artists to construct the validation dataset.
- Triple Dataset Construction
  - For each artist in the train, test, and validation datasets:
    - ✓ Select at most 10 images belonging to that artist.
    - ✓ For each pair of images from the same artist (2-combinations), randomly select another image from a different artist to create a triplet.
    - ✓ Repeat this process for all possible pairs within the same artist.
- CSV File Saving
  - After constructing the triple datasets for train, test, and validation, save each dataset into separate CSV files for later use.

Step 2-

Image Dataset Class:

Designed to load images dynamically, meaning it fetches them from storage as they're requested rather than loading them all at once. This reduces memory usage, making it suitable for large datasets on systems with limited RAM.

Step 3-

Data Augmentation:

- For Training:
  - **Random Rotation** (15 degrees): Rotating images randomly within a range of ±15 degrees helps the model learn to recognize objects from various angles, improving its robustness.
  - **Random Horizontal and Vertical Flip:** Flipping images horizontally or vertically randomly introduces variations in the dataset, aiding the model in learning invariant features across different orientations.
  - **Random Affine Transformation:** Affine transformations, including scaling, translation, rotation, and shearing, can simulate different perspectives of the images, enhancing the model's ability to generalize to unseen variations.
- For Training, Testing and Validation:
  - **Normalization:** Normalizing the images by subtracting the mean and dividing by the standard deviation across all pixels ensures consistency in data representation and helps the model learn more effectively.
  - **Center Crop to 224 Dimensions:** Center cropping all images to 224x224 pixels. (I have cropped all the images and saved them in a new zip file, so the images I used are already cropped).

Step 4-

Network Architecture:

- Architecture:
  - Utilizes a pre-trained ResNet-18 model.
  - Adjusts the fully connected layer (last layer) to produce embeddings of 256 dimensions.
- Forward Pass:
  - Accepts three input images (img1, img2, img3) representing an anchor, positive, and negative sample, respectively.
  - Computes embeddings for each input image using the modified ResNet-18 backbone.
  - Returns the computed embeddings for img1, img2, and img3.

Step 5-

Loss Function & Optimizer:

- Optimizer:
  - I used adam optimizer to optimize the parameters of your model with a learning rate of 0.00001.
- Triplet Loss Funtion:
  - Accepts three sets of embeddings: anchor, positive, and negative.
  - Computes the Euclidean distances between the anchor and positive embeddings, as well as between the anchor and negative embeddings.
  - Calculates the triplet loss using the margin formula.
  - Utilizes the ReLU function to ensure that only positive differences contribute to the loss.
  - Returns the mean loss value along with the distances of the anchor-positive and anchor-negative pairs.

✓ The margin represents the minimum difference between the distance of anchor-positive pairs and anchor-negative pairs.

Step 6-

Training Loop:

Iterating over each epoch, within which it loops through the training data loader. For each batch of anchor-positive-negative triplets, it computes the embeddings using the model, calculates the triplet loss, performs backpropagation, and updates the model parameters using the optimizer. It also calculates and stores the training loss and accuracy for plotting.

Step 7-

Validation Loop:

Switching the model to evaluation mode, it iterates over the validation data loader. For each batch, it computes the embeddings, calculates the loss, and evaluates the accuracy without updating the model parameters. The validation loss and accuracy are then calculated and stored for plotting.

Step 8-

Test Accuracy Computation:

Iterating through the test data loader, the model computes embeddings, calculates the triplet loss, and tracks both the loss and correct predictions. Subsequently, it computes the test accuracy and average loss to evaluate the network's performance on unseen data.

Step 9-

Training, Validation, and Test Loss and Accuracy Plots:

Each plot depicts the change in loss and accuracy as the number of epochs increases. This visualization provides insights into the model's performance during training, validation, and testing phases.

Parameters

Learning rate = 0.00001

Embedding size = 256

Margin (Triplet loss) = 4.0

Num of epochs = 12

Number of triplets in training data = 19092

Number of artists used in training = 783

Number of triplets in test data = 3883

Number of artists used in test = 198

Number of triplets in validation data = 4737

Number of artists used in validation = 195

Results

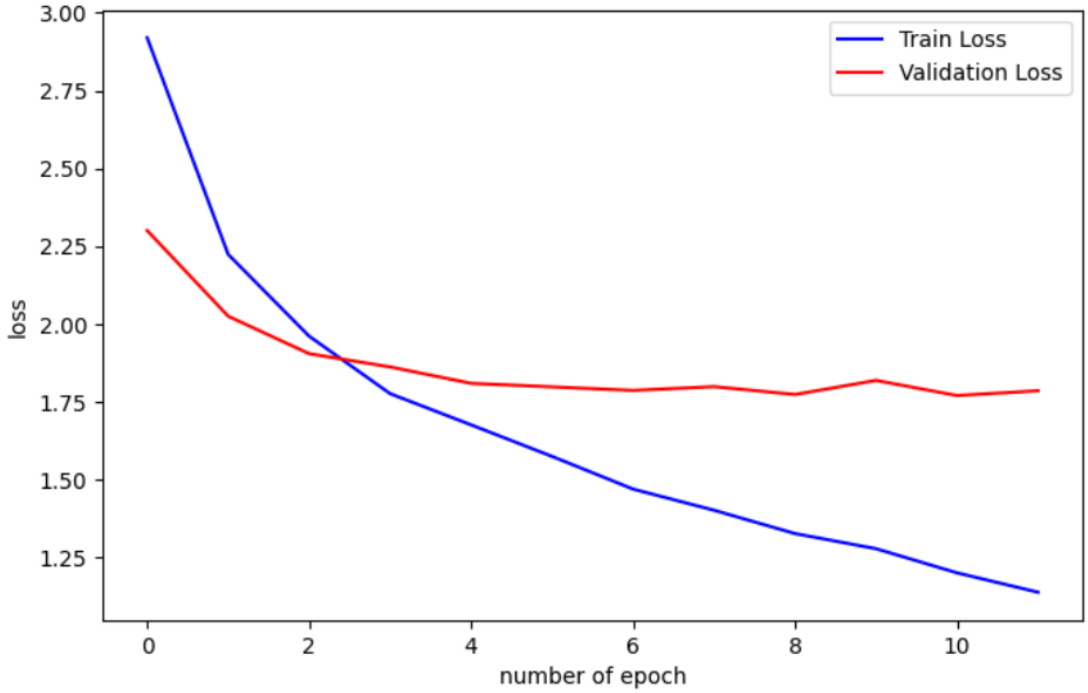
Training accuracy = 88.9011%

Validation accuracy = 81.0640%

Test accuracy = 79.5519%

Plots

Loss Plot:



Accuracy Plot: