Deep Learning - Spring 2025

Assignment 2

Image Retrieval with Contrastive Learning

Due Date: 11:59 PM on Sunday, 23-March-2025 **Total Marks:** 100

Submission:
Submit all of your codes and results in a single zip file with the name FirstName_RollNumber_01.zip

- Submit a single zip file containing
  (a) codes (b) report (c) Saved Models (d) Readme.txt
- There should be **Report.pdf** detailing your experience and highlighting any interesting results. Kindly don't explain your code in the report, just explain the results. Your report should include your comments on the results of all the steps, with images, for example, what happened when you changed the learning rate, etc.
- Readme.txt should explain how to run your code, preferably it should accept the command line arguments e.g dataset path used for training the model.
- The assignment is only acceptable in .py files. No Jupyter notebooks.
- In the root directory, there should be 2 python files, a report, and a folder containing saved models.
- Root directory should be named as **FirstName_RollNumber_02**
- Your code script files should be named as **'rollNumber_02_task1.py'**
- You have to submit a .py code file**.** Follow all the naming conventions.
- For each convention, there is a 3% penalty if you don't follow it.
- Email the instructor or TA if there are any questions. You cannot look at others' code or use others' code, however, you can discuss it with each other. **Plagiarism will lead to a straight zero with additional consequences as well.**
- 10% (of obtained marks) deduction per day for a late submission.
- The late submissions will only be accepted till **23 March at midnight.**
- **DON'T RESUBMIT THE DATASETS PROVIDED IN YOUR SUBMISSION.**
- Use the provided dataset for this assignment, do not use any other dataset.

Note: For this assignment (and for others in general) **you are not allowed to search online for any kind of implementation.** Do not share code or look at the other's code. You should not be in possession of any implementation related to the assignment, other than your own. In case of any confusion please contact the TAs (email them or visit them during their office hours).

Objectives: In this assignment, you will write the code implement an image retrieval system using contrastive learning. The goals of this assignment are as follows.

- Modify pre-trained ResNet-50 to generate embeddings.
- Train using contrastive loss with dynamic pair sampling.

⚠ **Important:** With your submission, you also need to submit one .py file named **'rollNumber_02_allCode.py'** in your zip folder. This file should contain the code of all 'code files' in your project. NO SUBMISSION WILL BE ACCEPTED WITHOUT IT.

Siamese Network

1. **Dataset**

   In this assignment, the "Caltech-101" dataset is used and shared with you. This dataset contains 102 classes, with a total of 9,145 images distributed across these classes. For an in-depth understanding of the dataset, refer to the following link: Caltech-101 Dataset.

2. **Split Data:**

   The dataset is stored in a single directory. Ensure that it is split into three sets: training, validation, and test. Maintain the same proportion of each class across all sets.

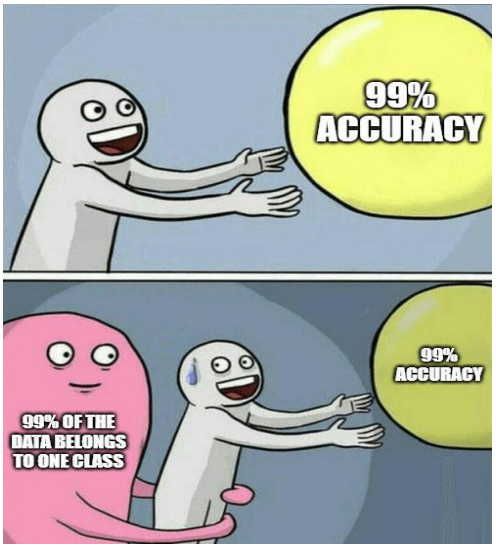
   For example, if you are splitting 10% of the data, ensure that 10% is taken from each class. Suppose there are 100 images across 10 classes, with each class containing 10 images—in this case, select 1 image per class to form the 10% split.

   You can achieve this using two approaches:

   - Copy Data: Write a script to copy the split data into separate train and test directories.
   - Dataframe Approach: A memory-efficient method where you create a dataframe, split it, and reference image paths and labels during training and testing instead of physically duplicating data.

2. **Data Loader**

   The data loader for this task is slightly complex. Initially, you can select two random images. If both images belong to the same class, assign a label of 0; otherwise, assign a label of 1. However, using this approach may lead to data imbalance, where label 1 samples (dissimilar pairs) dominate the dataset.



   To handle this imbalance, ensure that your data is balanced for both classes. While loading batches, you need to carefully select samples. The number of paired samples can range from 750 to 1500, and you can choose any value within this range. You may use PyTorch libraries to create the data loader efficiently.

Data Preprocessing

1. **Normalization**: Apply ImageNet mean/std normalization to images.

```
transform = transforms.Compose([
    transforms.Resize(224),
    transforms.ToTensor(),
    transforms.Normalize(
        mean=[0.485, 0.456, 0.406],   # ImageNet stats
        std=[0.229, 0.224, 0.225]
    )
])
```

2. **Verification**: Ensure that the normalized tensors have approximately mean ($\mu \approx 0$) and standard deviation ($\sigma \approx 1$).

Baseline Architecture

For Task 01, you must use the **ResNet-50** model as your backbone. Analyze its behavior with and without freezing the layers.

Feature Extraction

You will extract features using the backbone and further process them for similarity measurement.

```
model = torchvision.models.resnet50(pretrained=True)
model.fc = nn.Identity() # Remove classification head
```

```
class ModifiedResNet(nn.Module):
    def __init__(self):
        super().__init__()
        self.backbone = resnet50(pretrained=True)
        self.backbone.fc = nn.Identity()
        # Student adds: 2048->512->128 FC layers
```

Model Creation

Figure 1 provides an overview of the Siamese network architecture to be built in this task. The following conventions are used in the figure:

- **FC** → Fully Connected layer
- **BN** → Batch Normalization layer

You may skip batch normalization (BN) layers or fully connected (FC) hidden layers to ensure better generalization of the model.

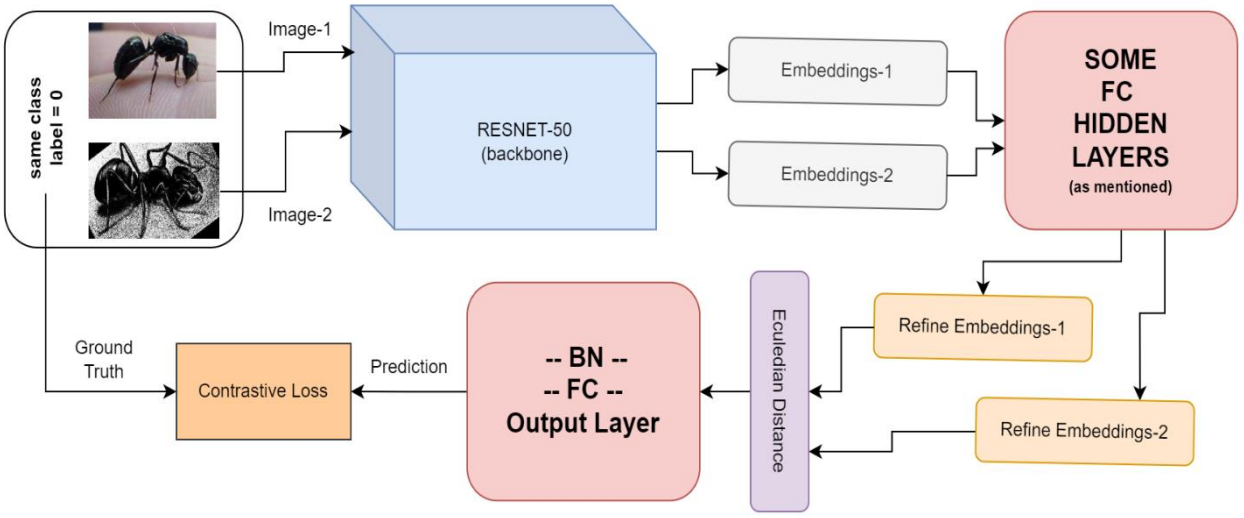


Figure 1: A sample diagram to make a siamese network workflow.

Contrastive Loss

Define a method named as contrastive loss and implement the equation as it is. Make sure your operations going smooth.

$$L = \frac{1}{2N}\sum_{i=1}^{N}[y_i d_i^2 + (1 - y_i)\max(\alpha - d_i, 0)^2]$$

Where, $d_i = \|e_a^i - e_p^i\|_2$.

Make sure your loss function is not like as:

Weights Update

Use either SGD or Adam optimizer. Initially, a higher learning rate may be beneficial for convergence, but ensure that the model achieves the global minimum and does not get stuck due to large step sizes.

Callbacks

To improve training stability, implement the following callbacks:

1. **Learning Rate Decay**: If the loss does not decrease, reduce the learning rate automatically.
2. **Save Best Weights**: Save or overwrite model weights whenever validation loss improves.

Metrics

Define separate methods to compute evaluation metrics. Specifically, implement:

- **Accuracy** (custom method)
- **Recall** (custom method)
- **Precision** (custom method)
- **F1-score** (custom method)
- **Classification Report** (use scikit-learn for this)
- **Confusion Matrix** (write a custom method to compute **TP, TN, FP, FN** using y_true and y_pred)

Be prepared to explain your implementation during evaluation.

Train & Test Methods

Write separate methods for **training** and **testing**. Ensure that your code is modular by dividing it into functions. If any operation is used multiple times, create a reusable

function and call it where necessary.

Evaluation

Create a separate file named test_eval.py, where:

- The best model is loaded.
- The user provides two image paths as input.
- The model predicts the similarity and displays results in a graph.
  While you may use Jupyter Notebook for experimentation, ensure that test_eval.py is included in your submission.

Visualization

- **Data Embeddings**: Generate embeddings with labels and visualize them before and after feature refinement.
- **Performance Comparison**: Create performance tables comparing **baseline vs. learned embeddings**.
- **t-SNE/PCA Visualizations**: Use dimensionality reduction techniques to visualize learned embeddings.
- **Graphs**: Make sure to add loss, accuracy, recall, precision training and validation graphs for each experiment.

Implementation Hints

- Use torch.nn.functional.normalize() for stable training.
- Start with a **margin (α) of 1.0**.
- Test embeddings with dummy input: print(output.shape).

References

- **Contrastive Loss**: Arxiv Paper
- **ResNet-50**: Arxiv Paper

Deliverables

```
project/
|-- rollNumber_02_task1.py'      # All code from data-loader to test & visualization.
|-- train.py                     # Training script
|-- test.py                      # Testing Script
|-- model.py                     # Siamese class
|-- data_utils.py                # Dataset/dataloaders
|-- weights                      # If weights are too large then upload it on your drive
                                 and share link in submission.
|-- Report.pdf                   # Detailed report including analysis.
|-- graphs                       # contain all the graphs as mentioned.
|-- requirements.txt             # Dependencies
'-- README.md                    # Execution instructions
```

Good Luck