CS771 - Intro to ML: Mini-project 2

Group Members

Kirandeep Parida
241110036

Arnika Kaithwas
241110011

Prabhakar Pandey
241110049

Tsewang Chukey
241110092

Tsewang Namgail
241110093

Department of Computer Science and Engineering

IIT Kanpur

November 2024

1 Problem 1: Task 1 - Model Training and Evaluation

1.1 Objective

The primary goal of Task 1 is to iteratively train and evaluate a model using datasets D1 to D10, where each dataset shares the same input distribution. The task involves calculating prototypes for the classes, updating them iteratively, and logging accuracy on the lower triangular matrix of evaluation datasets.

1.2 Methodologies

1.2.1 EfficientNet-B0 Pre-Training

A pre-trained EfficientNet-B0 model is used as a feature extractor. The classification head is removed, and all layers are unfrozen to allow fine-tuning. The model is moved to GPU for efficient computation.

1.2.2 Data Transformation and Loading

CIFAR-10 images are resized to 224x224 to match EfficientNet's expected input size. Images are normalized using ImageNet-specific means and standard deviations.A custom function load_train_dataset loads datasets D1 to D10, applying transformations and converting data into PyTorch DataLoader objects for batch processing.

1.2.3 Feature Extraction

The function extract_features processes data batches through EfficientNet to generate feature vectors using global average pooling. This reduces the data dimensionality while retaining essential information.

1.2.4 Euclidean LwP Classifier

The LwP (Learning with Prototypes) Classifier is defined with the following functionalities:

- **Prototype Calculation:** Computes class prototypes as the mean of feature vectors for each class.
- **Prediction:** Uses Euclidean distance between feature vectors and class prototypes to predict labels.
- **Prototype Update:** Dynamically updates prototypes with a weighted average of existing and new features.

1.2.5 Iterative Training and Evaluation

The function evaluate_and_log_lower_triangle iteratively trains and evaluates the model: For each dataset D_i:

Prototypes are calculated using labeled data.

The model is evaluated on all previous datasets D_1 to D_i (lower triangular).

Accuracy for each evaluation is stored in an accuracy matrix.

This ensures the model retains knowledge from earlier datasets while adapting to new data.

1.2.6 Result Logging

Accuracy is logged in a lower triangular matrix, where entry (i, j) represents the accuracy on dataset D_j after training with dataset D_i.

1.2.7 Model and Prototype Saving

The trained EfficientNet model (after all datasets) is saved as f10_model_final.pth. Prototypes for the final model f10 are saved in f10_prototypes.npy for reuse in subsequent tasks.

1.3 Results

The lower triangular accuracy matrix shows how well the model generalizes to earlier datasets after being trained on subsequent datasets.The final model f10 and its prototypes are stored for use in Task 2, ensuring continuity and adaptability.

The accuracy matrix for this task is shown below:

Accuracy Matrix for Task 1 (F1–F10)

Model	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10
F1	0.8544	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
F2	0.8412	0.8620	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
F3	0.8436	0.8532	0.8552	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
F4	0.8444	0.8568	0.8440	0.8700	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
F5	0.8420	0.8532	0.8436	0.8524	0.8564	0.0000	0.0000	0.0000	0.0000	0.0000
F6	0.8384	0.8492	0.8416	0.8556	0.8500	0.8624	0.0000	0.0000	0.0000	0.0000
F7	0.8380	0.8460	0.8404	0.8516	0.8452	0.8484	0.8556	0.0000	0.0000	0.0000
F8	0.8412	0.8524	0.8424	0.8540	0.8424	0.8444	0.8432	0.8568	0.0000	0.0000
F9	0.8464	0.8560	0.8480	0.8608	0.8524	0.8496	0.8524	0.8464	0.8564	0.0000
F10	0.8388	0.8472	0.8404	0.8488	0.8460	0.8468	0.8452	0.8436	0.8324	0.8628

Table 1: Accuracy Matrix for Task 1 (F1 to F10 across D1 to D10). Rows correspond to models F1 to F10, and columns correspond to datasets D1 to D10.

2 Problem 1: Task 2 - Iterative Model Update and Evaluation

2.1 Objective

The goal of Task 2 is to extend the classifier trained in Task 1 (model f10) to handle new datasets D'1 to D'10 (or D'11 to D'20, using adjusted indexing). These datasets have

different input distributions compared to D1 to D10. The task involves updating the classifier prototypes dynamically and evaluating performance on both the old and new datasets.

2.2 Methodologies

2.2.1 Starting Point: Task 1 Outputs

The EfficientNet-B0 model and prototypes from Task 1 are loaded:

- EfficientNet-B0 acts as a feature extractor with a pre-trained head removed.
- Prototypes for f10 are initialized using previously saved values.
- This ensures continuity in the learning process while retaining knowledge from Task 1

2.2.2 Dataset Preparation

Each dataset (D'11 to D'20) is loaded iteratively using load_train_dataset. Images are resized to 224x224 and normalized with ImageNet-specific means and standard deviations. Placeholder labels are used for datasets without ground-truth labels.

2.2.3 Feature Extraction

Features are extracted from EfficientNet-B0 using the extract_features function. EfficientNet-B0 generates feature vectors for all images in a dataset. Global average pooling reduces the dimensionality to 1280 while preserving relevant information.

2.2.4 Iterative Prototype Updates

The EuclideanLwPClassifier updates prototypes dynamically for each new dataset:

- **Prediction:** Labels are predicted using Euclidean distance from class prototypes.
- **Prediction:** Labels are predicted using Euclidean distance from class prototypes.

  Old Prototype Contribution (70%): Retains knowledge from previous datasets.

  New Features Contribution (30%): Adapts prototypes to the new dataset.

2.2.5 Evaluation Strategy

Each updated model (f11 to f20) is evaluated on: D1 to D10 (datasets from Task 1) to measure generalization on old data. D'1 to D'i (current and previous datasets from Task 2) to assess adaptability.

2.2.6 Accuracy Matrix

Rows of the matrix represent the accuracy of each updated model (f11 to f20) on all datasets. Diagonal values show the performance on the dataset used for updating the respective model.

2.2.7 Logging and Saving Results

The accuracy matrix for Task 2 is saved as accuracy_matrix_task2.npy for further analysis. Updated prototypes for each model are dynamically recalculated but not explicitly saved for each step.

2.3 Results

The accuracy matrix for this task is shown below:

Accuracy Matrix for Task 2 (F11–F20)

Model	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15	D16	D17	D18	D19	D20
F11	0.8352	0.8432	0.8368	0.8464	0.8460	0.8440	0.8412	0.8336	0.8292	0.8576	0.7196	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
F12	0.8252	0.8376	0.8324	0.8384	0.8396	0.8444	0.8304	0.8240	0.8240	0.8492	0.7144	0.6460	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
F13	0.8252	0.8368	0.8288	0.8384	0.8352	0.8440	0.8312	0.8212	0.8232	0.8456	0.7136	0.6348	0.7612	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
F14	0.8248	0.8316	0.8272	0.8380	0.8312	0.8396	0.8256	0.8220	0.8208	0.8420	0.7164	0.6224	0.7608	0.7860	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
F15	0.8340	0.8420	0.8368	0.8448	0.8416	0.8428	0.8344	0.8316	0.8304	0.8508	0.7176	0.6136	0.7632	0.7876	0.8312	0.0000	0.0000	0.0000	0.0000	0.0000
F16	0.8292	0.8372	0.8292	0.8416	0.8392	0.8380	0.8336	0.8280	0.8244	0.8508	0.7096	0.6032	0.7580	0.7860	0.8292	0.7424	0.0000	0.0000	0.0000	0.0000
F17	0.8288	0.8332	0.8284	0.8364	0.8388	0.8376	0.8324	0.8256	0.8204	0.8440	0.7192	0.6000	0.7580	0.7864	0.8264	0.7348	0.7524	0.0000	0.0000	0.0000
F18	0.8296	0.8296	0.8256	0.8336	0.8332	0.8364	0.8216	0.8232	0.8160	0.8404	0.7160	0.6032	0.7588	0.7872	0.8256	0.7348	0.7520	0.7400	0.0000	0.0000
F19	0.8268	0.8280	0.8268	0.8276	0.8360	0.8360	0.8208	0.8216	0.8164	0.8372	0.7024	0.5808	0.7460	0.7892	0.8160	0.7272	0.7448	0.7376	0.6656	0.0000
F20	0.8352	0.8412	0.8364	0.8396	0.8452	0.8432	0.8316	0.8296	0.8236	0.8476	0.7040	0.5864	0.7552	0.7876	0.8264	0.7264	0.7420	0.7412	0.6584	0.8132

Table 2: Accuracy Matrix for Task 2 (F11 to F20 across D1 to D20). Rows correspond to models F11 to F20, and columns correspond to datasets D1 to D20.

3 Problem 2 : Paper Presentation

Below is the you tube link of our paper presentation:

Title: Lifelong Domain Adaptation via Consolidated Internal Distribution
https://youtu.be/QksW3GMKEso