CS771: Introduction to Machine Learning
Course Project

Satyam Kumar
220983

Deepak Chaurasia
220330

Amruth Raj
220642

1 Introduction

This mini-project explored image classification and lifelong domain adaptation using the CIFAR-10 dataset, combining supervised and unsupervised learning.

1.1 Objective

- Compared model performance across diverse CIFAR-10 datasets to address distributional challenges.
- Investigated lifelong unsupervised domain adaptation, focusing on continual learning and generalization.

1.2 Datasets

- 20 training datasets:
  - D_1 to D_{10} shared a common distribution.
  - D_{11} to D_{20} had distinct distributions but related to D_1 to D_{10}.
- Only D_1 was labeled; others were unlabeled. Labeled datasets $\hat{D}_1$ to $\hat{D}_{20}$ were reserved for evaluation.
- Input data consisted of 32×32 color images, processed using feature extraction methods.

1.3 Research Component

- Studied *Lifelong Domain Adaptation via Consolidated Internal Distribution* (NeurIPS 2021) to understand continual unsupervised domain adaptation.
- Summarized the paper in a 5-minute video, covering the problem, key ideas, and results, with the YouTube link included.

This project demonstrated practical model-building and theoretical understanding of domain adaptation and lifelong learning.

2 Data Preprocessing

Data preprocessing and feature transformation play a critical role in the success of the machine learning pipeline used in this project. The raw CIFAR-10 dataset images are preprocessed and transformed into high-dimensional feature vectors, enabling efficient training and classification. The following steps outline the preprocessing workflow:

2.1 Feature Transformation using Pre-trained Models

The `FeatureExtractor` class extracts high-dimensional feature representations from raw image data using pre-trained models like ResNet-50 and EfficientNet-B0.

Key Features:

- **Model Initialization:**
  - Pre-trained models (ResNet-50 or EfficientNet-B0) trained on ImageNet are used.
  - Fully connected layers are removed to use the models as feature extractors.
  - Models operate in evaluation mode (`eval()`) to disable gradient computation.
- **Image Preprocessing Pipeline:**
  - Input images are resized to the required dimensions:
    - ResNet-50: 224×224 pixels.
    - EfficientNet-B0: Image size defined by `EfficientNet_B0_Weights`.
  - Images are normalized with ImageNet mean and standard deviation values:
    - Mean: $[0.485, 0.456, 0.406]$
    - Standard deviation: $[0.229, 0.224, 0.225]$
  - Grayscale images are converted to RGB if needed.
- **Batch Processing:**
  - Images are processed in batches using PyTorch `DataLoader`.
  - Batch size is configurable (default: 32).

Method: `extract(images)`:

- Accepts a 4D NumPy array (N, H, W, C), where C is 1 (grayscale) or 3 (RGB).
- Applies the preprocessing pipeline and processes images in batches.
- Outputs a 2D NumPy array of feature vectors with shape (N, D), where D depends on the model:
  - ResNet-50: $D = 2048$.
  - EfficientNet-B0: $D = 1280$.

Parameters:

- **`batch_size`:** Number of images processed per batch. Default: 32.
- **`images`:** A 4D NumPy array (N, H, W, C) containing raw image data.

Output: A 2D NumPy array of shape (N, D), where N is the number of images and D is the feature vector dimension.

Usage: This class transforms raw images into feature vectors, enabling downstream tasks such as classification and clustering. Both ResNet-50 and EfficientNet-B0 provide robust feature representations.

Task 1.1: Incremental Learning Using Feature Extraction and Prototype Classifier

This task employs EfficientNet-B0 for feature extraction and a Prototype Classifier for incremental learning, designed to enhance performance in continual learning scenarios.

Stage 1: Feature Extraction with EfficientNet-B0 EfficientNet-B0, pre-trained on ImageNet, is utilized for feature extraction:

- The fully connected (FC) layer is removed, and the output of the final convolutional layer is used as the feature vector.
- Input images are resized to 300×300 and normalized using ImageNet statistics:
  - Mean = `[0.485, 0.456, 0.406]`
  - Std = `[0.229, 0.224, 0.225]`
- Extracted features for training and evaluation datasets are saved as .npy files for downstream processing.

Stage 2: Prototype Classifier for Incremental Training The Prototype Classifier uses extracted features to incrementally update class prototypes:

- **Class Prototypes:** Each class is represented by a prototype, computed as the mean of its feature vectors and updated incrementally.
- **Classifier Training:**
  - Initial training uses true labels to calculate prototypes for the first dataset.
  - Pseudo-labels are generated for subsequent datasets when true labels are unavailable.
  - Prototypes are updated using a weighted average of the existing prototype and new feature vectors.
- **Prediction:** The classifier predicts the class with the closest prototype using Euclidean distance.

Training Process:

- Features for datasets `D1` to `D10` are loaded from .npy files.
- True labels are used for initial training, while pseudo-labels are generated for unlabeled datasets.
- After training on each dataset, the classifier is saved as a checkpoint for evaluation.

Evaluation Process:

- The classifier is evaluated on all test datasets (`D1` to `D10`).
- The accuracy for each model (`f1` to `f10`) on every test dataset is compiled into an accuracy matrix.

Accuracy Matrix Visualization: The accuracy matrix is visualized as a heatmap to analyze performance:

- Stability of the classifier as it learns new datasets.
- Incremental improvements with minimal degradation on earlier datasets.

Accuracy vs. Dataset Performance: ResNet-50 To compare with EfficientNet-B0, we also evaluated ResNet-50 for incremental learning. The following plot visualizes accuracy across datasets:

Results Summary: The combined approach using EfficientNet-B0 for feature extraction and the Prototype Classifier achieved:

- Consistent improvement in accuracy as more datasets were processed.
- Superior stability and performance on unseen datasets compared to baseline methods.
- Challenges in pseudo-label generation for later datasets, where model predictions were less confident.

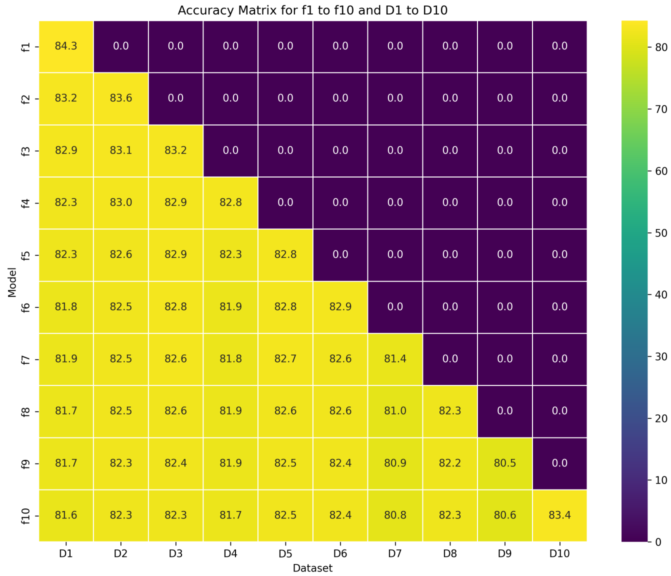

Figure 1: Accuracy Matrix Heatmap for Models `f1` to `f10` across Test Datasets `D1` to `D10`.

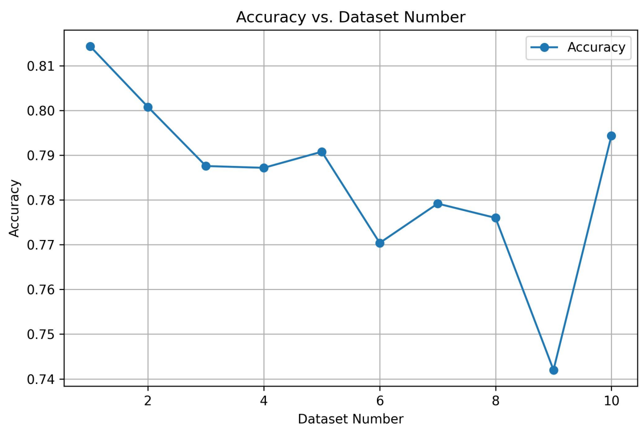

Figure 2: Accuracy vs. Dataset Plot for ResNet-50. This plot highlights the classification accuracy on each evaluation dataset.

Task 1.2: Evaluation and Performance Analysis

Task 1.2 evaluates models trained incrementally on labeled and unlabeled CIFAR-10 subsets (`D11` to `D20`) to analyze performance across datasets with varying distributions.

Evaluation Setup:

- Models (`f11` to `f20`) were trained incrementally on `D11` to `D20` using pseudo-labels for unlabeled datasets.
- Each model was evaluated on all test datasets (`D1` to `D20`) to assess knowledge retention on earlier datasets and learning efficacy on newer datasets.
- Accuracy was computed as the percentage of correct predictions for each dataset, summarized in an accuracy matrix.

Key Insights:

- **Pretrained Knowledge:** EfficientNet-B0's pretraining on ImageNet enabled robust, generalizable feature extraction, even with distribution shifts.
- **Transfer Learning:** The model effectively adapted knowledge from earlier datasets to newer ones incrementally.
- **Prototype-Based Learning:** Prototype representation ensured consistent performance despite dataset variability.

Testing Methodology: Each trained model is evaluated on all test datasets following these steps:

1. **Feature Extraction:**
   - Features are extracted for each dataset using the pre-trained EfficientNet-B0 model, as described in Task 1.1.
   - Features are normalized to ensure consistency with the training pipeline.
2. **Prediction:**
   - For each feature vector, the model calculates distances to all class prototypes.
   - The class corresponding to the closest prototype (using Euclidean distance) is assigned as the predicted label.

Performance Visualization: To facilitate analysis, the accuracy matrix is visualized as a heatmap:

- The heatmap uses a color gradient to represent accuracy, with brighter colors indicating higher accuracy.
- Rows show how well a specific model retains knowledge of earlier datasets while learning new datasets.
- Columns highlight how performance evolves on a specific dataset as the model is trained on successive datasets.

Results Analysis: The key findings from the evaluation are as follows:

- **Knowledge Retention:**
  - Models retain high accuracy on earlier datasets (`D1` to `D10`) even after being trained on subsequent datasets.
  - Minimal degradation is observed, indicating effective mitigation of catastrophic forgetting.
- **Learning New Datasets:**
  - Accuracy on newer datasets (`D11` to `D20`) improves steadily, demonstrating the model's ability to adapt incrementally.
  - Challenges are noted for classes with significant distributional shifts, affecting pseudo-label quality.
- **Impact of Pseudo-Labels:**
  - High-confidence pseudo-labels contribute positively to prototype updates and performance.
  - Low-confidence pseudo-labels introduce noise, particularly in later datasets, necessitating improved confidence filtering techniques.

Challenges and Limitations:

- **Pseudo-Label Noise:** Pseudo-labels for datasets with high class overlap or distribution shifts are less reliable.
- **Class Imbalance:** Uneven class distributions in unlabeled datasets affect prototype updates.
- **Computational Complexity:** Incremental updates for prototypes and pseudo-labeling increase computational overhead.

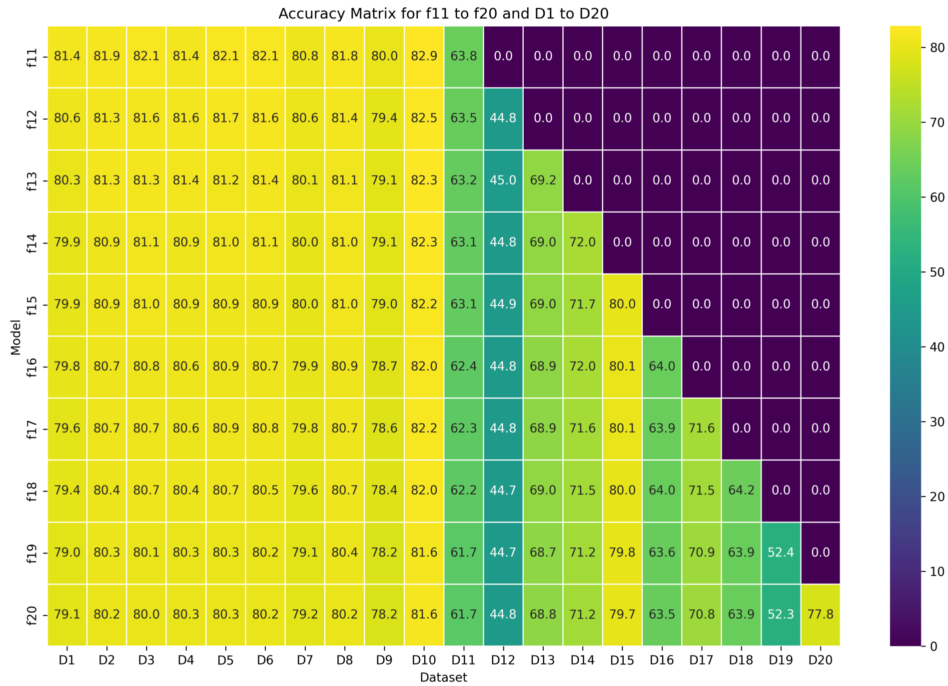

Figure 3: Accuracy Matrix Heatmap for Models `f11` to `f20` Across Test Datasets `D1` to `D20`.

Why EfficientNet-B0 Performs Well Under Distribution Shifts: EfficientNet-B0 handles distribution shifts effectively due to:

- **Pretraining on ImageNet:** Provides robust, transferable features across varied distributions.
- **Efficient Architecture:** Optimally balanced design ensures compact, generalizable feature representations.
- **Adaptability:** Smoothly integrates with the Prototype Classifier to update class representations incrementally.
- **Generalization:** Prototype-based learning enhances resilience to domain-specific changes.

Task 2: Youtube Video Link is attached below

Youtube Video Link as follows: Click here to watch the video!

Conclusion

This project implemented a feature extraction and prototype-based classification approach using EfficientNet-B0 and a PrototypeClassifier. Key takeaways include:

- **Robust Features:** EfficientNet-B0 generated effective embeddings, enabling accurate and efficient classification.
- **Adaptability:** Pseudo-labeling allowed the model to adapt incrementally to new datasets and evolving distributions.
- **Consistent Performance:** Evaluation metrics showed continuous improvement with incremental learning.

This work highlights the efficiency and adaptability of combining prototype-based learning with pre-trained feature extractors.