Intro to Machine Learning (CS771)
Mini-Project 2

Ayush Meena
220268

Archit Atrey
220195

Rishikesh Dargad
220891

Patil Piyush Shivaji
220759

Group Name - PaaaR

Group No - 5

Instructor : Dr. Piyush Rai

Introduction

In recent years, semi-supervised learning (SSL) has emerged as a crucial paradigm in machine learning, allowing models to leverage both labeled and unlabeled data effectively. This report focuses on implementing a **Learning with Prototypes (LwP)** framework to solve a semi-supervised image classification problem using subsets of the **CIFAR-10** dataset. The project aims to sequentially train models using limited labeled data and an abundance of unlabeled data while preserving performance across previously encountered datasets.

Problem Statement

The project involves 20 training subsets $\mathbf{D_1}, \mathbf{D_2}, \ldots, \mathbf{D_{20}}$ of the **CIFAR-10** dataset, where:

- The first 10 classes $\mathbf{D_1}$ to $\mathbf{D_{10}}$ share a common input distribution.
- The remaining 10 datasets $\mathbf{D_{11}}$ to $\mathbf{D_{20}}$ originate from different input distributions, yet retain a degree of similarity with the first 10 datasets.

Crucially, only $\mathbf{D_1}$ is labeled, while the remaining datasets are unlabeled. For evaluation, 20 labeled held-out subsets $\hat{\mathbf{D}}_1, \hat{\mathbf{D}}_2, \ldots, \hat{\mathbf{D}}_{20}$ are provided, ensuring that performance can be objectively measured without influencing model training.

Task 1.1

In Task 1, the goal is to train a sequence of models $\mathbf{f_1}, \mathbf{f_2}, \ldots, \mathbf{f_{10}}$ on the first 10 subsets of the CIFAR-10 dataset ($\mathbf{D_1}, \mathbf{D_2}, \ldots, \mathbf{D_{10}}$) in a semi-supervised manner. The task begins by training the first model $\mathbf{f_1}$ on the labeled dataset $\mathbf{D_1}$ using a **Learning with Prototypes (LwP)** classifier. For each subsequent dataset $\mathbf{D_i}$, the model $\mathbf{f_{i-1}}$ is used to predict pseudo-labels, which are then used to update the model to $\mathbf{f_i}$. This process is repeated for all 10 datasets.

Each model is evaluated on its corresponding held-out dataset $\hat{\mathbf{D}}_i$ and all prior held-out datasets to ensure that performance on previous datasets does not degrade significantly.

Approach

The solution employs a **Vision Transformer (ViT)** pretrained on the **ImageNet-21k** dataset (which has low degree of similarity with CIFAR-10 dataset) to extract meaningful features from input images. A prototype-based classification scheme is then used to iteratively train and update models across datasets.

Feature Extraction

The raw input images, which are 32×32 color images, are first resized to 224×224 pixels to match the input size expected by the ViT model. The Vision Transformer model, **ViT-base-patch16-224-in21k**, was chosen for its ability to capture global image features efficiently.

The images are processed using the **ViTFeatureExtractor** from the Hugging Face transformers library. This feature extractor applies the necessary transformations including resizing, normalization using the mean and standard deviation of the pretrained ViT model, and converting the images into tensor format. These transformations ensure that the images are compatible with the pretrained ViT model.

Once the images are transformed, features are extracted by passing them through the ViTModel. Specifically, the output of the model is taken from the [CLS] token (the first token of the sequence), which is a learned representation of the entire image. These features serve as the representations of the images, capturing high-level semantic information useful for classification.

The extracted features are then used to compute class prototypes for the Learning with Prototypes (LwP) classifier, as described later in the report.

Prototype Computation

Class prototypes are computed as the mean of feature vectors belonging to each class. For the initial dataset $\mathbf{D_1}$, labeled training data is used to compute the prototypes. For subsequent datasets, pseudo-labels are generated, and confident samples are used for prototype updates.

Pseudo-labeling and Prototype Updates

To handle unlabeled datasets $\mathbf{D_2}$ to $\mathbf{D_{10}}$, pseudo-labeling is applied:

1. Compute distances between extracted features and class prototypes.

2. Assign pseudo-labels based on the nearest prototype.
3. Select samples with confidence above a predefined threshold (0.8 in this case).

Using these confident samples, prototypes are updated iteratively using momentum:

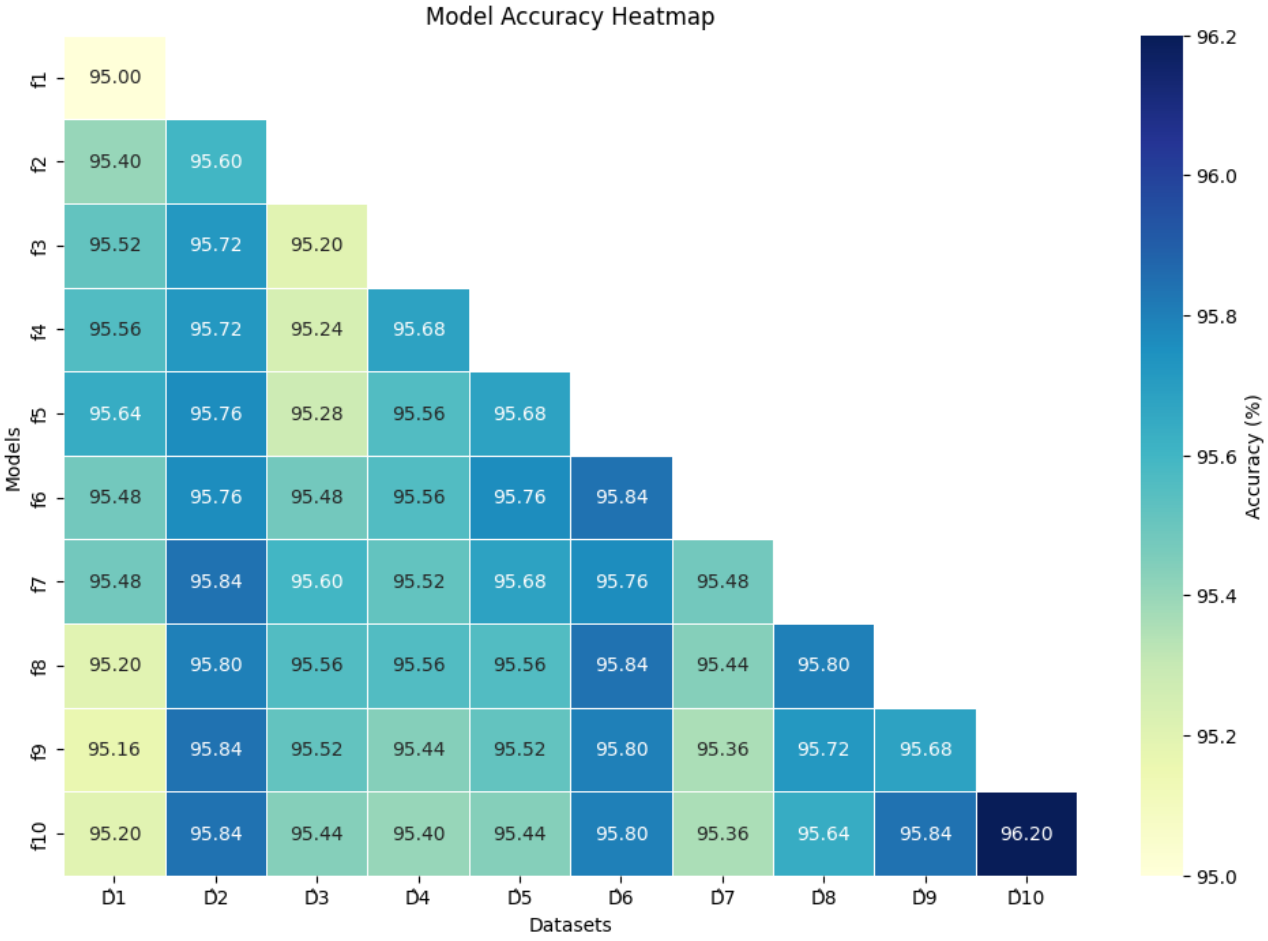
$$\text{Prototypes}_{\text{new}} = \alpha \cdot \text{Prototypes}_{\text{old}} + (1 - \alpha) \cdot \text{Prototypes}_{\text{pseudo}}$$

where $\alpha = 0.9$ is the momentum factor.

Evaluation

Each model $\mathbf{f_i}$ ($i = 1, \ldots, 10$) is evaluated on held-out datasets $\hat{\mathbf{D}}_\mathbf{1}, \ldots, \hat{\mathbf{D}}_\mathbf{i}$. The evaluation uses:

- Prototype-based classification: Query features are classified based on the nearest prototype.
- Accuracy computation: Match predicted labels with ground-truth labels.



Conclusion

The described approach successfully trains and updates models sequentially while maintaining performance on previous datasets. By leveraging ViT for feature extraction and prototype-based classification, the method achieves efficient continual learning. The implementation emphasizes simplicity and adaptability, aligning well with the objectives of Task 1.1.

Task 1.2

Task 1.2 extends the continual learning approach to datasets $\mathbf{D_{11}}$ to $\mathbf{D_{20}}$, which exhibit varying input distributions. The challenge lies in updating the models sequentially to adapt to distributional shifts while maintaining performance on previously seen datasets. This report describes the approach and methodology used to achieve this.

Approach

The methodology builds on the prototype-based classification scheme from Task 1.1, starting with the final model ($\mathbf{f_{10}}$) from Task 1.1 as the initial set of prototypes. Specific adjustments are made to address the challenges posed by distributional shifts in the new datasets.

Prototype Initialization

The prototypes learned from Task 1.1 ($\mathbf{f_{10}}$) are loaded as the starting point. These prototypes are iteratively updated as new datasets are introduced, while leveraging confident pseudo-labels.

Feature Extraction and Pseudo-labeling

For each dataset $\mathbf{D_{11}}$ to $\mathbf{D_{20}}$, features are extracted using the Vision Transformer (ViT) model. Given the potential distributional differences, the pseudo-labeling strategy is adjusted to be more conservative:

1. Compute distances between extracted features and prototypes from the previous model.
2. Assign pseudo-labels based on the nearest prototype.
3. Retain samples with confidence exceeding a predefined threshold (e.g., 0.8).

Prototype Updates with Momentum

Prototypes are updated using the pseudo-labeled samples and a momentum-based approach:

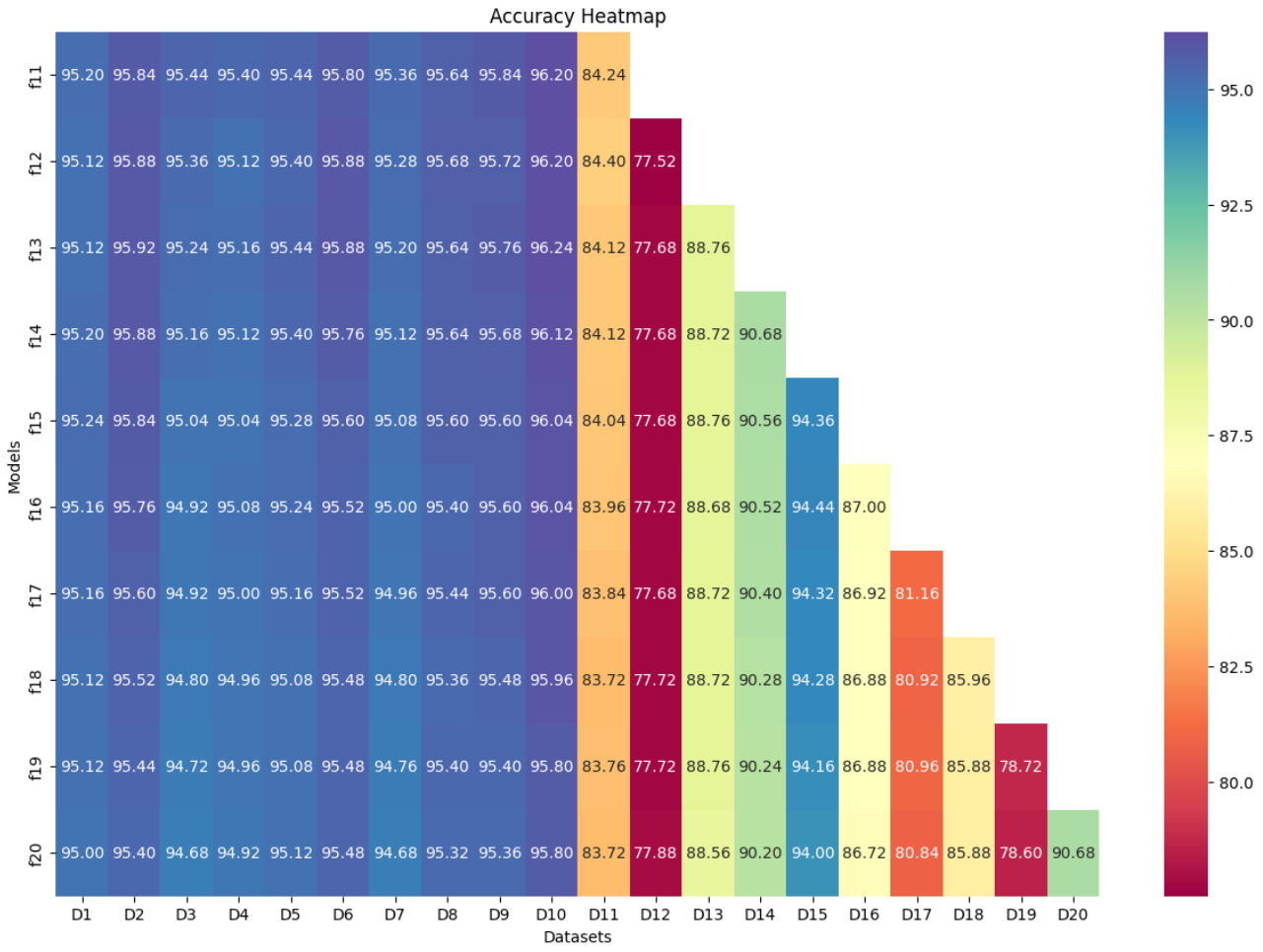
$$\text{Prototypes}_{\text{new}} = \alpha \cdot \text{Prototypes}_{\text{old}} + (1 - \alpha) \cdot \text{Prototypes}_{\text{pseudo}}$$

where $\alpha = 0.9$ is the momentum factor. This slower update rate ensures stability and avoids overfitting to the new datasets, which may differ significantly in distribution.

Evaluation

Each updated model $\mathbf{f_i}$ ($i = 11, \ldots, 20$) is evaluated on all held-out datasets $\hat{\mathbf{D}}_\mathbf{1}, \ldots, \hat{\mathbf{D}}_\mathbf{i}$. The evaluation process uses:

- Prototype-based classification: Query features are classified based on the nearest prototype.
- Accuracy computation: Match predicted labels with ground-truth labels.



Conclusion

The described approach successfully adapts models to datasets with varying input distributions. By leveraging Vision Transformer features, momentum-based updates, and cautious pseudo-labeling, the methodology achieves a balance between adapting to new datasets and retaining performance on earlier ones. The results demonstrate the effectiveness of the prototype-based framework in continual learning scenarios with distributional shifts.

Task 2

Video Link - https://drive.google.com/file/d/17fVb7OPKXkSDXCVckIrGIjqejPoc-XhA/view?usp=sharing

Dependencies –

In this project, the following libraries and tools were mainly used:

Python Libraries:

- **`PyTorch`**: For training and evaluating the machine learning models, specifically for handling model training, prototype updates, and dataset manipulation.
- **`TorchVision`**: For dataset preprocessing and transformations.

Deep Learning Framework:

- **`Transformers (HuggingFace)`**: For feature extraction with ViT (Vision Transformer), which was used in the model for embedding images.

CUDA and GPU Libraries:

- **`CUDA 12.4`**: To leverage GPU acceleration for faster training.
- **`cuDNN 9.1.0`**: For optimized deep learning operations on NVIDIA GPUs.
- **`NVIDIA GeForce RTX 3050`** GPU was used.

References

- PyTorch Documentation. *PyTorch: Deep Learning Framework*. Available at: https://pytorch.org/docs/stable/.
- Chollet, F. (2017). *Deep Learning with Python*. Manning Publications. Available at: https://www.manning.com/books/deep-learning-with-python.
- Hugging Face. *Transformers Library*. Available at: https://huggingface.co/transformers/.
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press. Available at: https://www.deeplearningbook.org/.