CS771: Introduction to Machine Learning

Mini Project 2

SynthAdaptMix

Group Number 100: Learners

Shivam Sharma (221016) Banoth Saidishanth (220282)

Abstract

This document presents our project, an implementation of the **Lifelong Domain Adaptation Using Consolidated Internal Distribution (LDAuCID)** and **Continual Domain Shift Learning (CDSL)** algorithms for various datasets which are subsets of the CIFAR-10 image classification dataset, utilizing a custom classifier based on Learning with prototypes (LwP). There are 20 datasets with the first 10 of these databases being from the same distribution, while others belong to different distributions, still having some similarity. We have proposed an algorithm **SynthAdaptMix** for this, which is a modified version of LDAuCID and CDSL. The main challenge to this is not saving any important data points into a memory buffer. As the research papers, we used as our reference stored important data points. So, we combined a method of synthetic data generation based on previous data distribution to tackle this challenge. These proposed methods are evaluated on all the datasets, demonstrating their effectiveness in adapting to new domains while maintaining accuracy. Key results include accuracy metrics across various models and datasets, visualized through heatmaps.

1 Introduction

Imagine a world where models could adapt like humans. Faced with a changing environment, they wouldn't stumble—they will learn, flex, and keep going. This project explores that very concept: **continual learning** in machine learning. Here, our model adjusts with every new data batch, evolving to handle *domain shift*—those tricky scenarios where the training environment doesn't match the data.

Why does this matter? Because most machine learning models perform well only in stable environments. When data sources change—say, different weather, lighting, or device—the model can get thrown off. But real-world data isn't static. So our goal? Using synthetic data and adaptive training, we're tackling domain shift without storing any datasets.

1.1 Background

Traditional machine learning follows this "train-once" approach, using all labeled data in one go, assuming data distribution won't change. In practice, though, new data often does look different.

So what can be done? **Domain Adaptation**. The idea is to keep the model agile, allowing it to generalize better across different scenarios without labeled data in every new domain. In this project, we blend domain adaptation with continual learning. We use synthetic and augmented data generation and prototype-based classification to teach the model to adapt in real-time.

1.2 Objective

Develop a method that lets the model:

- Handle shifting data environments with *minimal retraining*.
- Use augmented data to "fill in the blanks" when real data falls short.
- Generalize across domains without needing labeled data for each.

1.3 Significance

So, why does this matter? Adaptability. Picture this: A self-driving car or medical diagnostic tool. These systems can't afford to be thrown off by a new scenario. They need to handle changes gracefully. By working on continual adaptation, we're developing a framework that could be used in any system facing changing data.

2 Code Overview

2.1 Data Preprocessing

Each dataset contains 2500 raw images (32×32x3 size color images). The data is clean, balanced among all classes, and has no Null values. To extract features, we used **ResNet-50** network which is pre-trained on the ImageNet dataset to get 1000 features from each dataset.

2.2 Libraries

- Deep Learning Framework: torch, & torchvision.
- Data processing and analysis: numpy, scikit-learn, & scipy.
- Data Visualization: matplotlib & seaborn.

2.3 Model Architecture

Our model architecture includes the following components:

- **LwP Classifier Class:** It is a prototype-based classifier that iteratively refines class prototypes to represent clusters in labeled data. The classifier initializes these prototypes as class-wise centroids and then optimizes them through an iterative process that employs distance-weighted averaging and momentum-based updates. During optimization, it computes Euclidean distances between training samples and current prototypes, assigns samples to their nearest prototype, and updates the prototypes using a weighted scheme.The update process incorporates a **decay factor** to balance between new and historical prototype positions, while an **adaptive tolerance** ensures stable convergence. For classification, it employs a nearest-prototype strategy, assigning test samples to the class of their closest prototype based on Euclidean distance. This approach is particularly effective in continual learning scenarios where class distributions may evolve, as it maintains stable decision boundaries while adapting to distribution shifts.
- **Synthetica Class:** It implements a data synthesis framework that leverages **Kernel Density Estimation (KDE)** to model and generate synthetic data samples that maintain the underlying distribution characteristics of the original dataset. The class utilizes **StandardScaler** for feature normalization and employs a **Gaussian** kernel-based KDE with an adjustable **bandwidth** parameter to estimate the probability density function for each class independently. The **analyze distribution** method first normalizes the input data and then fits separate KDE models for each class, storing both the fitted KDE model and the scaler for later use. The **synthesize data** method generates synthetic samples by sampling from these learned distributions, with the capability to specify the number of samples per class. The generated samples are then inverse-transformed using the stored scaler to restore them to the original feature space. This approach enables the generation of synthetic data that preserves the statistical properties and class-specific characteristics of the original dataset, making it mainly useful for data augmentation and handling class imbalance in machine learning applications, especially in scenarios involving continual learning where maintaining distribution knowledge is crucial.
- **SynthAdaptMix:** It implements a comprehensive framework for synthetic data generation and adaptation in continual learning scenarios. The class combines multiple data manipulation strategies, with its core functionality centered around the **RandMix** method, which performs feature-level mixing of samples within each class. This method generates synthetic samples by randomly pairing features within the same class and combining them using a uniformly sampled **mixing ratio**, effectively creating new samples that maintain class-specific characteristics while introducing controlled variability. The class also incorporates a **combine data** method that uses **K-means clustering** to select representative samples from combined datasets, helping to manage memory efficiency while preserving data distribution characteristics. The framework includes integrated functionality for classifier training through the **train classifier** method, which initializes and updates a Learning with Prototypes (LwP) classifier, along with methods for **pseudo-label prediction** and model evaluation that calculates accuracy scores. This comprehensive approach enables adaptive synthetic data generation while maintaining class consistency, making it particularly effective for handling domain shift and class evolution in continual learning settings, where the balance between preserving previous knowledge and adapting to new data distributions is crucial.

2.4 Methodology

For this project, we decided to use a modified version of **LDAuCID** and **CDSL** algorithms, with data analysis and synthesis. Initially, the labeled training features (D1) are augmented using **RandMix**, which gives new samples through intra-class data mixing, helping in data generalization. The combine data method then merges these augmented samples with the original labeled data, yielding a balanced feature-label set. A KDE-based distribution is calculated using the Synthetica class

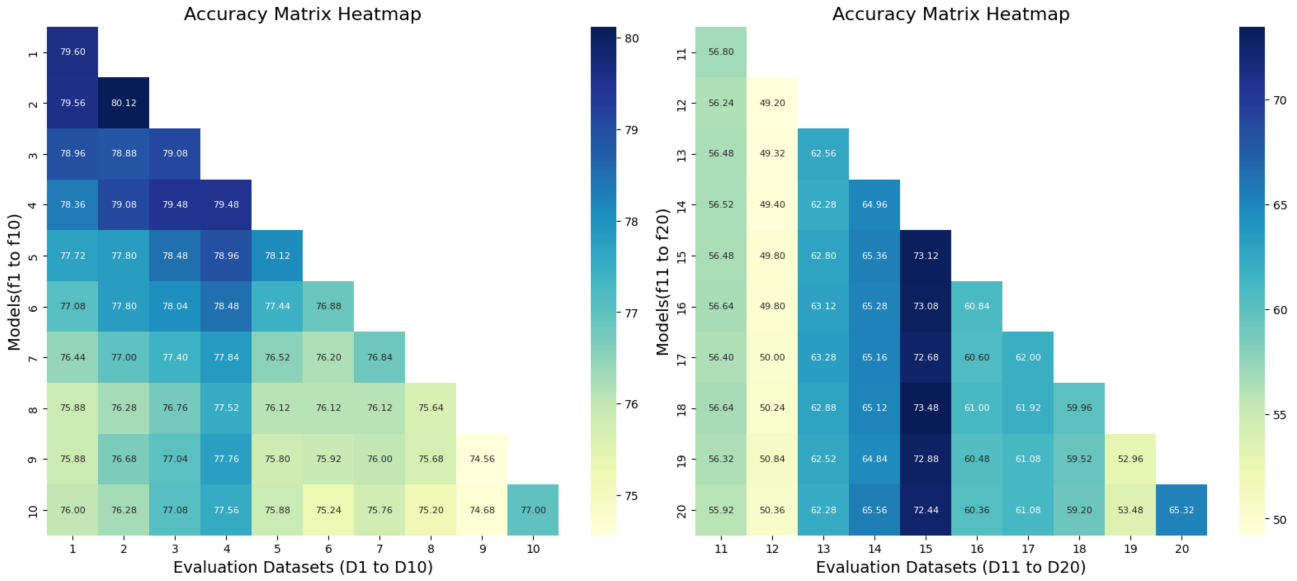
to represent class data density. The first classifier (f1) is trained on the combined data. For subsequent domains, the last classifier predicts pseudo-labels on the new data, and synthetic samples are created based on previous data density. This data is also passed through RandMix to get augmented data. Data diversity is further increased by mixing augmented, synthetic, and current samples, then selecting representative clusters through K-Means. This refined dataset is used to retrain the classifier, which adapts progressively with each new domain. Bandwidth hypermeter in the Synthetica class is adaptively lowered to improve density estimates in later stages, allowing for fine-grained distribution modeling in complex domain shifts.

2.5 Evaluation Metrics and Visualization

The primary metrics used in this study are:

- Metrices: Mean Accuracy, Standard Deviation, Minimum and Maximum Accuracy.
- Visualization: Accuracy Matrix Heatmap.

3 Results



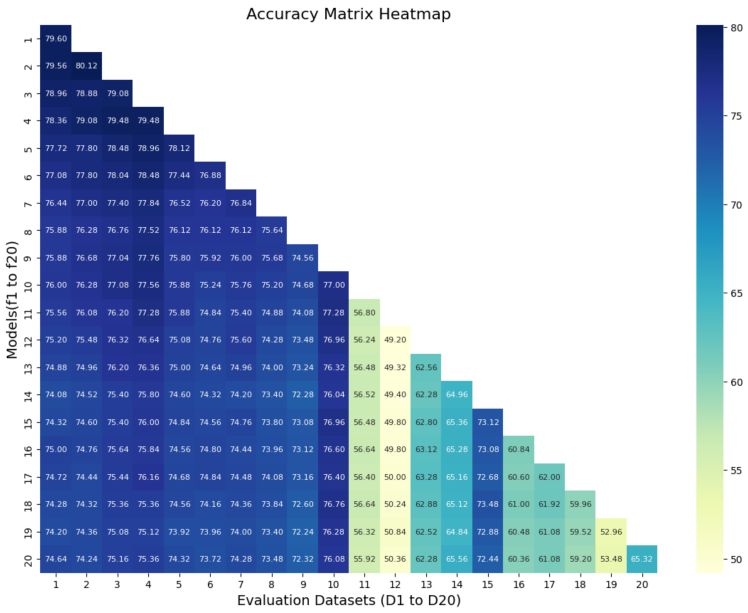


Table 1: Performance Index for Each Model

Models	Mean Accuracy (%)	Max Accuracy (%)	Models	Mean Accuracy (%)	Max Accuracy (%)
f1	79.60	79.60	f11	74.03	77.28
f2	79.84	80.12	f12	71.60	76.96
f3	78.97	79.08	f13	70.69	76.36
f4	79.10	79.48	f14	69.84	76.04
f5	78.22	78.96	f15	70.39	76.96
f6	77.62	78.48	f16	69.84	76.60
f7	76.89	77.84	f17	69.32	76.40
f8	76.31	77.52	f18	68.71	76.76
f9	76.15	77.76	f19	67.58	76.28
f10	76.07	77.56	f20	67.48	76.08

Table 2: Performance Results

Models	Mean Accuracy(%)	Standard Deviation(%)	Minimum Accuracy(%)	Maximum Accuracy(%)
f1 to f10 on D1 to D10	77.17	1.36	74.56	80.12
f10 to f20 on D11 to D20	60.16	6.66	49.2	73.48
f1 to f20 on D1 to D20	71.60	7.75	49.2	80.12

4 Conclusion

- We can see that each model from f1 to f20, have mean accuracy of 67% to 80%. This means our algorithm has adapted well on new datasets.
- As D1 to D10 are from the same distribution, our algorithm generates great results on them. Models f1 to f10 produce an accuracy of 74% to 80% on these datasets. This suggests that this domain adaptation is very stable.
- From accuracy matrix, we can see that models f11 to f20 perform well on datasets D1 to D10. This means these models that when a model is updated, the performance on previous datasets is not degrading significantly.
- But as Datasets D11 to D20 are from different distribution, f11 accuracy drops to 56.80% on D11 from 77.28% of f10 on D10. But models slowly adapt to this new distribution as well.

5 Acknowledgments

We want to thank Professor Piyush Rai, for his guidance and support throughout this project.

6 Video Presentation

DEJA VU: Continual Model Generalization for Unseen Domains

7 Extent of contribution

- Shivam Sharma (221016): Made the entire code and this report.
- Banoth Saidishanth (220282): Made the video presentation and discussed the code.

Other members didn't contribute at all, even after multiple requests.

References

[1] Mohammad Rostami, *USC Information Sciences Institute, Los Angeles*, *Lifelong Domain Adaptation via Consolidated Internal Distribution*, Proceedings of NeurIPS 2021.

[2] Chenxi Liu, Lixu Wang, Lingjuan Lyu, *DEJA VU: Continual Model Generalization for Unseen Domains*, 2023.