Kasukabe_Defence_Group(Group 8) Mini-project 2

Anurag Gupta
220185

Kushagra Gupta
220571

Mayank Agrawal
220636

Pushpdeep Teotia
220840

Video Link for presentation LDAuCID

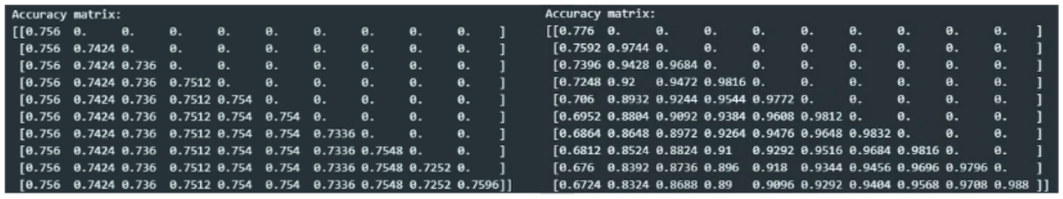
1 Overview

This project aims to develop a machine learning model for lifelong learning and domain adaptation, particularly in image classification, where the model must adapt to new datasets while retaining knowledge from previous ones. Our approach utilizes a prototype-based classifier that iteratively updates class prototypes as new datasets are introduced. The datasets D1 to D10 are drawn from similar distributions, while D11 to D20 are from significantly different distributions, presenting a challenge of domain shift. To address this, we incorporate domain adaptation techniques to minimize the impact of this shift, ensuring the model maintains strong performance across all datasets. Additionally, PCA-based dimensionality reduction is used to efficiently handle the high-dimensional data from image features. This strategy allows the model to learn continuously from new data without forgetting prior knowledge, enabling it to adapt seamlessly to both similar and diverse domains, making it well-suited for real-world applications requiring robust and ongoing adaptation to new data distributions.

2 Feature Extraction

In the initial stages of the project, we used MobileNet for feature extraction from the images. MobileNet is a lightweight neural network architecture designed for efficient computation, especially in resource-constrained environments. While MobileNet provided reasonable feature extraction, it did not offer the optimal performance for our task.

To improve the feature extraction process, we switched to using ResNet, a deeper and more complex network known for its ability to handle a wide range of image classification tasks. Along with ResNet, we incorporated a regularizer to prevent overfitting and to enhance the model's ability to generalize across different datasets. The results showed that ResNet, with regularization, outperformed MobileNet in terms of accuracy, as it was better able to capture and extract relevant features from the images. The accuracy improvement was evident, demonstrating that the more complex architecture and the regularization technique allowed the model to handle the domain shift and varying dataset distributions more effectively.



```
Accuracy matrix:
[[0.756  0.     0.     0.     0.     0.     0.     0.     0.     0.    ]
 [0.756  0.7424 0.     0.     0.     0.     0.     0.     0.     0.    ]
 [0.756  0.7424 0.736  0.     0.     0.     0.     0.     0.     0.    ]
 [0.756  0.7424 0.736  0.7512 0.     0.     0.     0.     0.     0.    ]
 [0.756  0.7424 0.736  0.7512 0.754  0.     0.     0.     0.     0.    ]
 [0.756  0.7424 0.736  0.7512 0.754  0.754  0.     0.     0.     0.    ]
 [0.756  0.7424 0.736  0.7512 0.754  0.754  0.7336 0.     0.     0.    ]
 [0.756  0.7424 0.736  0.7512 0.754  0.754  0.7336 0.7548 0.     0.    ]
 [0.756  0.7424 0.736  0.7512 0.754  0.754  0.7336 0.7548 0.7252 0.    ]
 [0.756  0.7424 0.736  0.7512 0.754  0.754  0.7336 0.7548 0.7252 0.7596]]
```

```
Accuracy matrix:
[[0.776  0.     0.     0.     0.     0.     0.     0.     0.     0.    ]
 [0.7592 0.9744 0.     0.     0.     0.     0.     0.     0.     0.    ]
 [0.7396 0.9428 0.9684 0.     0.     0.     0.     0.     0.     0.    ]
 [0.7248 0.92   0.9472 0.9816 0.     0.     0.     0.     0.     0.    ]
 [0.706  0.8932 0.9244 0.9544 0.9772 0.     0.     0.     0.     0.    ]
 [0.6952 0.8804 0.9092 0.9384 0.9608 0.9812 0.     0.     0.     0.    ]
 [0.6864 0.8648 0.8972 0.9264 0.9476 0.9648 0.9832 0.     0.     0.    ]
 [0.6812 0.8524 0.8824 0.91   0.9292 0.9516 0.9684 0.9816 0.     0.    ]
 [0.676  0.8392 0.8736 0.896  0.918  0.9344 0.9456 0.9696 0.9796 0.    ]
 [0.6724 0.8324 0.8688 0.89   0.9096 0.9292 0.9404 0.9568 0.9708 0.988 ]]
```

MobileNet vs Resnet

Figure 1: MobileNet vs Resnet

3 Approach for Task 1

For Task 1, we focus on training the model iteratively on a sequence of datasets D_1 to D_{10}, where the datasets D_1 to D_{10} share similar distributions. The approach consists of the following steps:

- **Initial Training:** We begin by training the model f_1 on the first dataset D_1, which is a labeled dataset. A prototype-based classifier is used, where class prototypes are initialized based on the average feature vectors of the labeled data. This model is then used to predict the labels for the subsequent unlabeled datasets.
- **Pseudo-Labeling and Iterative Training:** For datasets D_2 to D_{10}, we do not have ground truth labels. Therefore, we employ a pseudo-labeling strategy where the model f_1, trained on D_1, is used to predict the labels for these unlabeled datasets. The pseudo-labeled data is then used to train a new model, f_2, which updates the class prototypes. This process continues iteratively, where each new model f_i is trained using the previous model's pseudo-labels.
- **Regularization and Prototype Update:** The class prototypes are updated after each training step to refine the model's knowledge. We also incorporate regularization terms during the prototype update to ensure that the model does not overfit the pseudo-labeled data. This step ensures that the classifier retains the knowledge from previous datasets while adapting to the new data.
- **Evaluation:** After training each model f_i, we evaluate the performance on all previous datasets $D_1, D_2, \ldots, D_i$, using the updated model. The accuracy matrix is constructed by comparing the model's predictions with the true labels for each dataset.

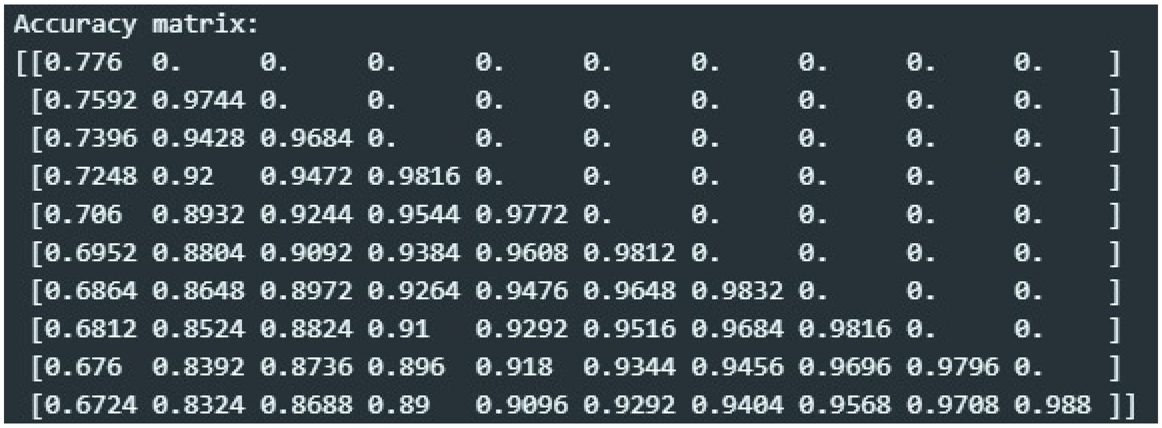
4 Approach for Task 2

Task 2 involves extending the learned model to handle datasets D_{11} to D_{20}, where these datasets have different distributions compared to D_1 to D_{10}. The key challenge in Task 2 is to adapt the model to the domain shift while maintaining performance on previously seen datasets. The approach for Task 2 is as follows:

- **Domain Adaptation for Models f_{10} to f_{20}:** Since datasets D_{11} to D_{20} have different distributions, we start with the model f_{10} (the model trained on datasets D_1 to D_{10}) and adapt it for the new domain. A domain adaptation strategy is employed to update the class prototypes by adjusting them to minimize the domain shift. This ensures that the model can handle the new datasets without significant performance degradation.
- **Iterative Training on New Datasets:** For each new dataset D_{11} to D_{20}, the model f_{i-1} is used to generate pseudo-labels for the unlabeled data in the current dataset. The classifier is then updated using these pseudo-labels, and the class prototypes are refined accordingly. The domain adaptation step ensures that the prototypes evolve to account for the new distribution, while retaining knowledge from the previous datasets.
- **Regularization and Prototype Adaptation:** Similar to Task 1, regularization is applied to avoid overfitting on the new datasets. Additionally, domain adaptation is used to ensure

the prototypes are adjusted to minimize domain shift. The prototypes are adapted using a weighted average of the old and new prototypes, which helps in reducing the impact of domain shift.

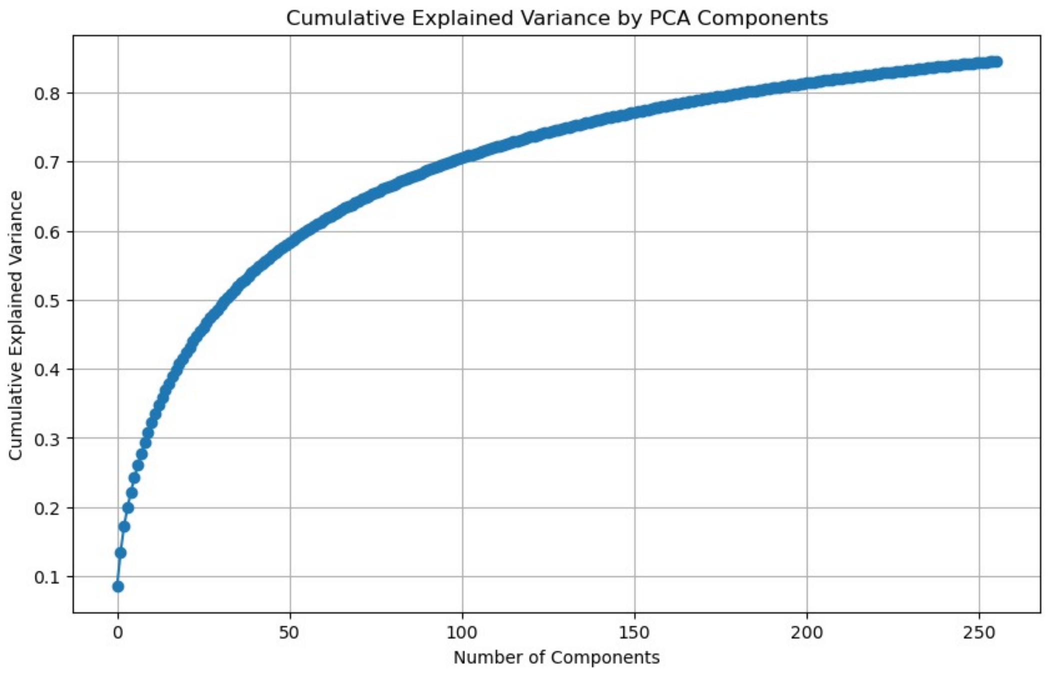
- **Evaluation on All Datasets:** After training each model f_{11} to f_{20}, the performance is evaluated not only on the new dataset but also on all previous datasets D_1 to D_{10}. The accuracy matrix is updated to track the model's performance across all datasets, ensuring that the model can retain and transfer knowledge across both similar and different distributions.



```
Accuracy matrix:
[[0.776  0.     0.     0.     0.     0.     0.     0.     0.     0.    ]
 [0.7592 0.9744 0.     0.     0.     0.     0.     0.     0.     0.    ]
 [0.7396 0.9428 0.9684 0.     0.     0.     0.     0.     0.     0.    ]
 [0.7248 0.92   0.9472 0.9816 0.     0.     0.     0.     0.     0.    ]
 [0.706  0.8932 0.9244 0.9544 0.9772 0.     0.     0.     0.     0.    ]
 [0.6952 0.8804 0.9092 0.9384 0.9608 0.9812 0.     0.     0.     0.    ]
 [0.6864 0.8648 0.8972 0.9264 0.9476 0.9648 0.9832 0.     0.     0.    ]
 [0.6812 0.8524 0.8824 0.91   0.9292 0.9516 0.9684 0.9816 0.     0.    ]
 [0.676  0.8392 0.8736 0.896  0.918  0.9344 0.9456 0.9696 0.9796 0.    ]
 [0.6724 0.8324 0.8688 0.89   0.9096 0.9292 0.9404 0.9568 0.9708 0.988 ]]
```

Figure 2: 10 x 10 matrix for task 1.1

Graphs for task 1.1



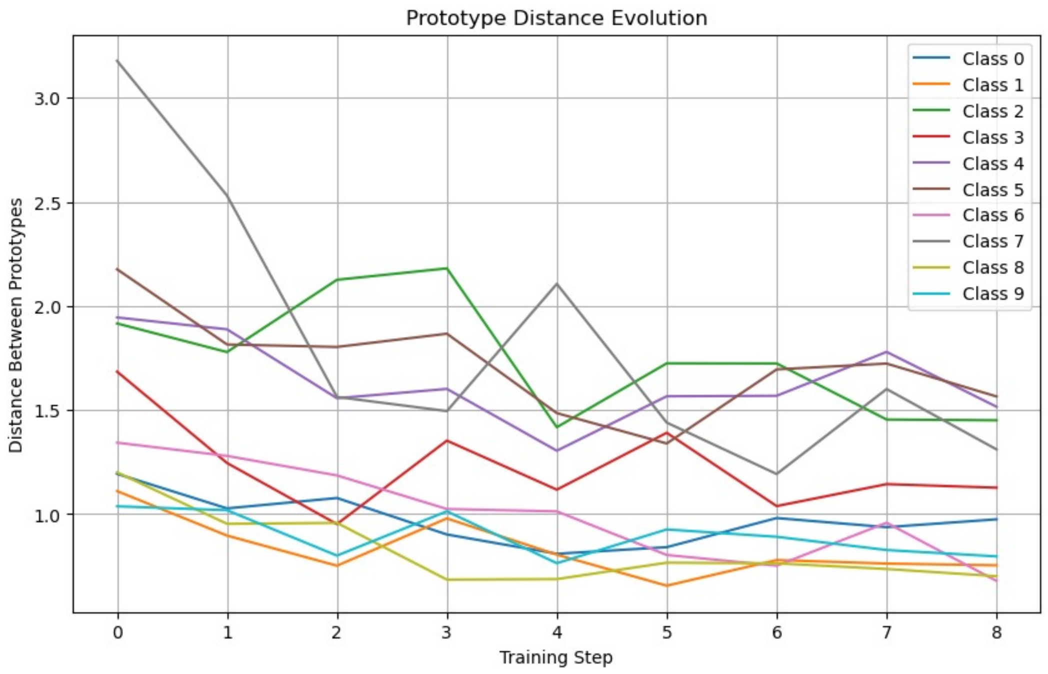


Figure 3: Prototype Distance Evolution

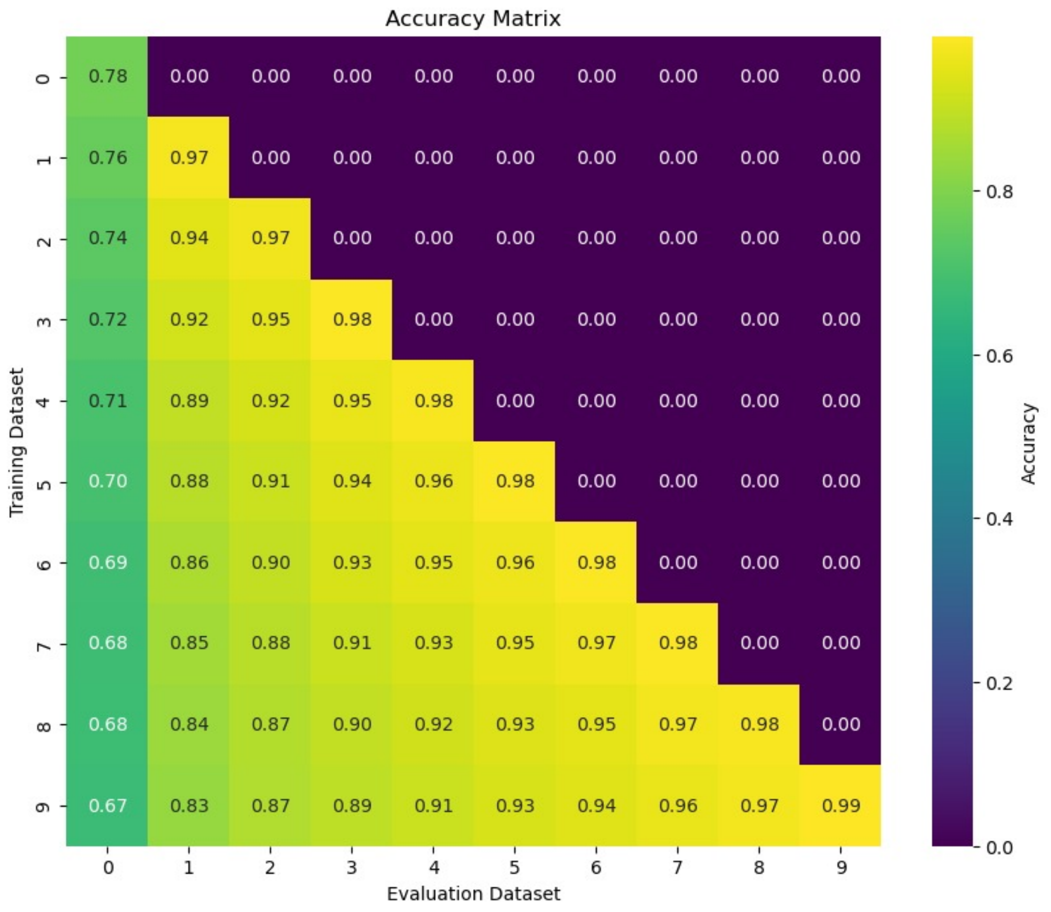


Figure 4: Heatmap for accuracies

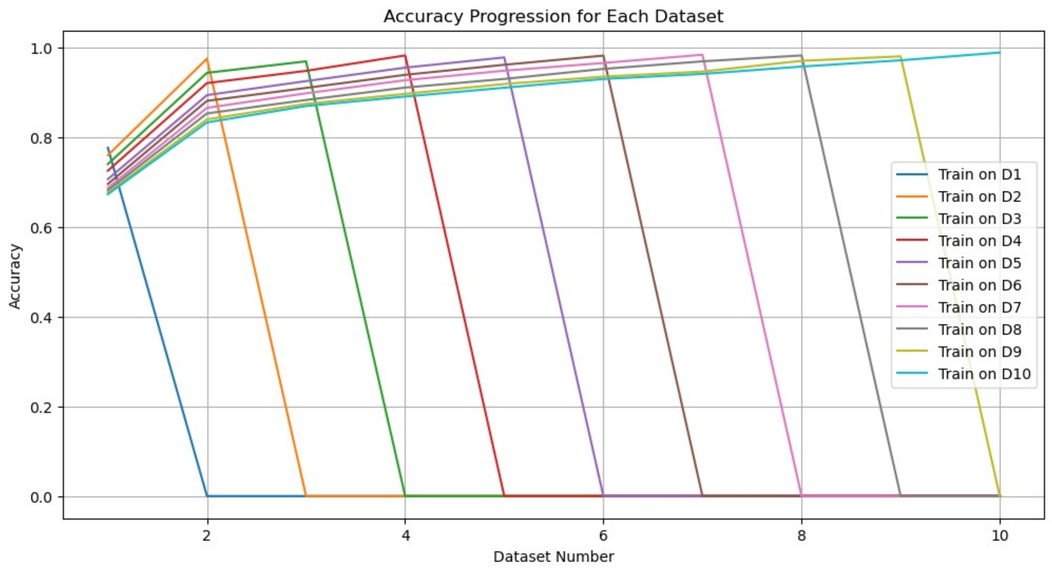
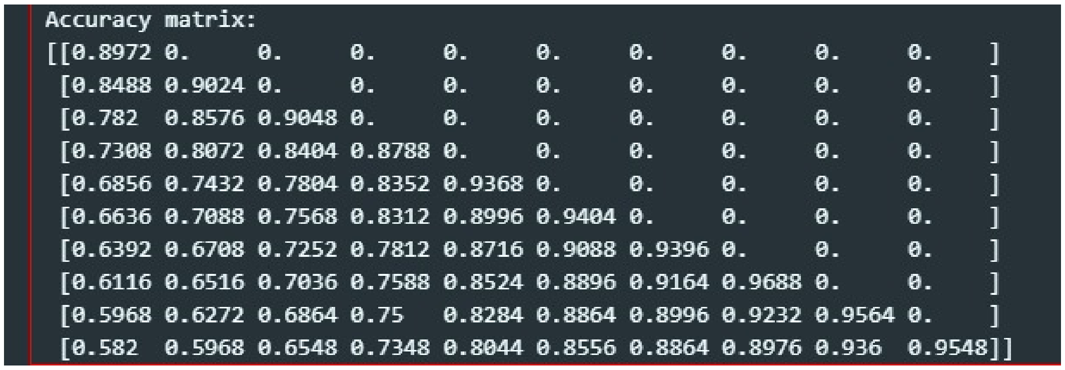


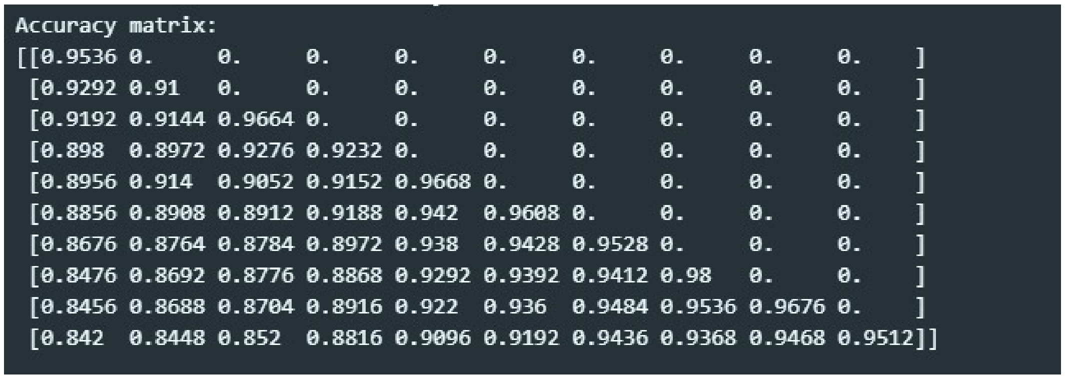
Figure 5: Accuracy Progression

Graphs for task 1.2



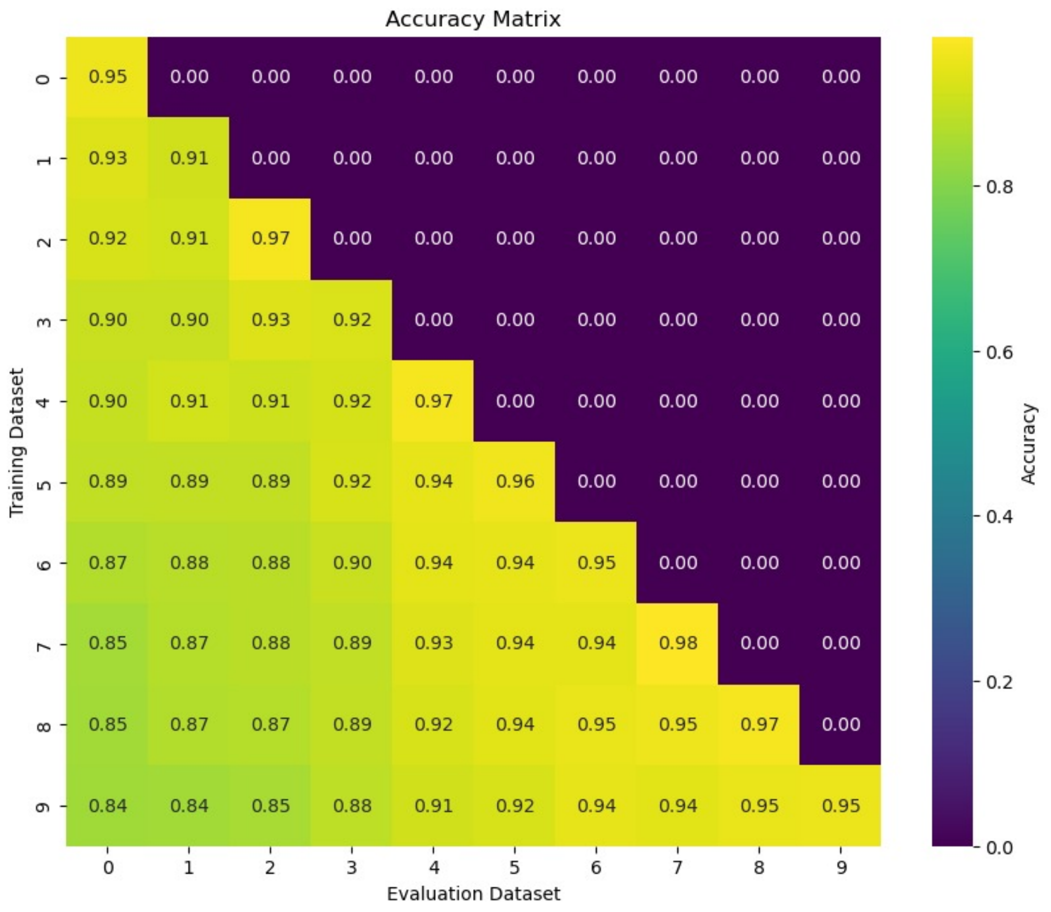
```
Accuracy matrix:
[[0.8972 0.     0.     0.     0.     0.     0.     0.     0.     0.    ]
 [0.8488 0.9024 0.     0.     0.     0.     0.     0.     0.     0.    ]
 [0.782  0.8576 0.9048 0.     0.     0.     0.     0.     0.     0.    ]
 [0.7308 0.8072 0.8404 0.8788 0.     0.     0.     0.     0.     0.    ]
 [0.6856 0.7432 0.7804 0.8352 0.9368 0.     0.     0.     0.     0.    ]
 [0.6636 0.7088 0.7568 0.8312 0.8996 0.9404 0.     0.     0.     0.    ]
 [0.6392 0.6708 0.7252 0.7812 0.8716 0.9088 0.9396 0.     0.     0.    ]
 [0.6116 0.6516 0.7036 0.7588 0.8524 0.8896 0.9164 0.9688 0.     0.    ]
 [0.5968 0.6272 0.6864 0.75   0.8284 0.8864 0.8996 0.9232 0.9564 0.    ]
 [0.582  0.5968 0.6548 0.7348 0.8044 0.8556 0.8864 0.8976 0.936  0.9548]]
```

Figure 6: 10 x 10 matrix without any updation of weights i.e. catastrophically loss



```
Accuracy matrix:
[[0.9536 0.     0.     0.     0.     0.     0.     0.     0.     0.    ]
 [0.9292 0.91   0.     0.     0.     0.     0.     0.     0.     0.    ]
 [0.9192 0.9144 0.9664 0.     0.     0.     0.     0.     0.     0.    ]
 [0.898  0.8972 0.9276 0.9232 0.     0.     0.     0.     0.     0.    ]
 [0.8956 0.914  0.9052 0.9152 0.9668 0.     0.     0.     0.     0.    ]
 [0.8856 0.8908 0.8912 0.9188 0.942  0.9608 0.     0.     0.     0.    ]
 [0.8676 0.8764 0.8784 0.8972 0.938  0.9428 0.9528 0.     0.     0.    ]
 [0.8476 0.8692 0.8776 0.8868 0.9292 0.9392 0.9412 0.98   0.     0.    ]
 [0.8456 0.8688 0.8704 0.8916 0.922  0.936  0.9484 0.9536 0.9676 0.    ]
 [0.842  0.8448 0.852  0.8816 0.9096 0.9192 0.9436 0.9368 0.9468 0.9512]]
```

Figure 7: 10 x 10 matrix after LDAuCID



Training \ Evaluation	0	1	2	3	4	5	6	7	8	9
0	0.95	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1	0.93	0.91	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	0.92	0.91	0.97	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3	0.90	0.90	0.93	0.92	0.00	0.00	0.00	0.00	0.00	0.00
4	0.90	0.91	0.91	0.92	0.97	0.00	0.00	0.00	0.00	0.00
5	0.89	0.89	0.89	0.92	0.94	0.96	0.00	0.00	0.00	0.00
6	0.87	0.88	0.88	0.90	0.94	0.94	0.95	0.00	0.00	0.00
7	0.85	0.87	0.88	0.89	0.93	0.94	0.94	0.98	0.00	0.00
8	0.85	0.87	0.87	0.89	0.92	0.94	0.95	0.95	0.97	0.00
9	0.84	0.84	0.85	0.88	0.91	0.92	0.94	0.94	0.95	0.95

Figure 8: Heatmap for Distribution

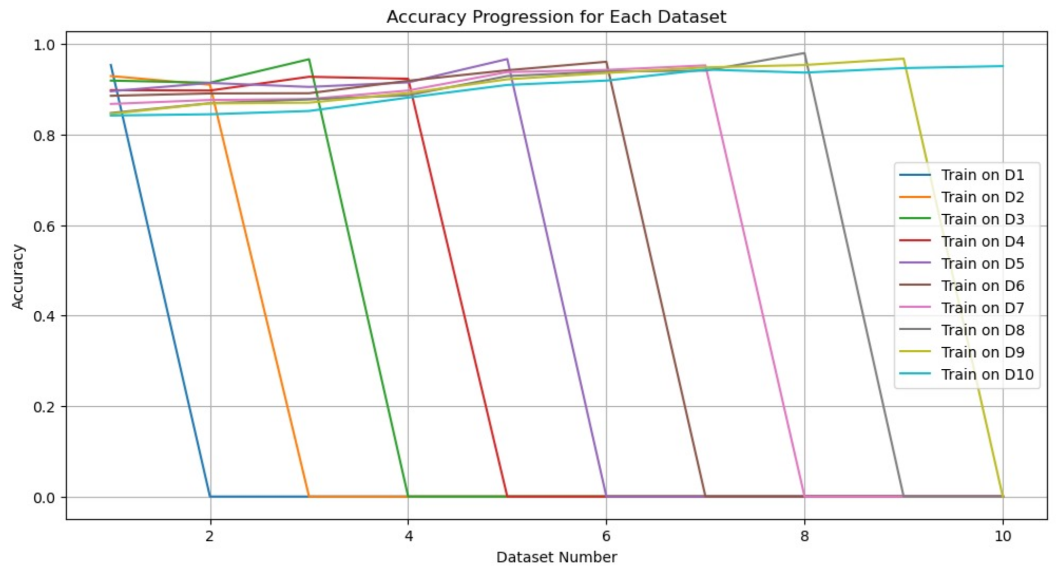


Figure 9: Progression of accuracy on 1.2

5 Learning

During the experimentation, we observed a slight loss in accuracy in Task 1.1 when evaluating the performance on previous datasets after updating the weights. However, the loss was not significant. In Task 1.2, we encountered a more pronounced issue: there was a catastrophic loss in accuracy for the previously trained datasets when using the new weights. This behavior was expected, as outlined in the referenced paper, and is evident in the accuracy matrices presented above.

To address this, we incorporated several enhancements, such as the LDAuCID approach, as suggested in the paper. Instead of completely changing the weights after each iteration, we opted for a more gradual adjustment. The updated weights were then used in the subsequent iteration for testing. This strategy allowed us to mitigate the domain shift between datasets, especially considering that the datasets in Task 1.2 have different distributions. As a result, even though the distribution of the datasets in Task 1.2 was different from those in Task 1.1, we were able to maintain relatively stable accuracy. The final accuracy matrix shows only a minor dip in the accuracy of previously trained datasets, as shown in the final matrix above.