Mini-Project 2- Image Classification- Report

Aryan Agarwal
241110012

Sahil Basia
241110061

Harsh Baid
241110026

Tanuj Agarwal
241110076

Aditya Azad
241110002

1 Problem 1

We were given image data for 10 datasets, and we had to train Learning with Prototype models only in sequence of these datasets. Only first dataset was labeled. Clearly this was a Continual Learning paradigm, with Unsupervised Domain Adaptation (UDA) techniques like Domain-Adversarial Neural Networks (DANN) or Maximum Mean Discrepancy (MMD) between distributions of features set 1 and 2. in Task 2. But we were given a limitation that we cannot use previous dataset features in anyway in modeling Model 2. Thus we went with different approaches to handle Continual Learning including feature extraction from ResNet50, InceptionV3 and finally EfficientNet deep neural network which are pre-trained, as explained in each tasks below.

1.1 Task 1

- We used Resnet50 first with different techniques like Batch Normalization, Average and Max pooling through different layers of resnet50 model like conv4_block6_out and conv5_block3_out layers (Please refer to the "rough_trials_approaches.ipynb" file for these approaches).
- We also tried regularization on the prototypes and covariance matrix we made in training each model, and finally applied weighted (this was hyperparamter tuned and also based on problem statement) prototype update from previous model in updating next. All of this was done to handle Catastrophic Information loss in subsequent update of class prototypes by each dataset and to avoid overfitting. Since Task 1 datasets were supposed to be from same distributions, our models performed well as well.
- We also performed fine tuning resenet50 model by adding additional layers for getting better features, but that didn't perform well as better as EfficientNet model.
- We also tried to model a Gausian Mixture Model (GMM) with initial means and covariances initialized form dataset 1, so as to learn different class distributions and use its prediction further. But we found other models performed better than this (mentioned later)
- For final model we used EfficientNet for feature extraction, and we then used a Principal Component Analysis (PCA) which was hyperparameter tuned to deliver us desired features which we used in our LWP (learning with Prototypes) models. This is also helped us visualize the high dimensional features onto 2D space to see how data looks like and if we can get some distribution to fit it using LWP variant (mahalanobis distance or cosine similarity based)
- We performed L2 Regularization with lambda=0.01 (again hyperparameter tuned) (this was done primarily in Task 2, but we tested it in Task 1 as well) with weighted update of class prototypes and inverse of class specific covariance matrixes which are to be used in Mahalanobis Distance calculations (our chosen distance metric in LWP Models. We tried with other metrics as well which can be seen in the rough trials approach sheet). We chose this distance because it captures the class distribution shape as well in the distance calculation, hence is more robust for our dataset and model accuracies.

Figure 1 is the final models result table as given instructions. Each row represents respective model's performance (accuracy) on different datasets (represented by columns) (please note the value in ratio, example 0.7 represent 70% accuracy, and so on). We got a 95%+ training accuracy and a general

85% Evaluation Dataset accuracy. There is scope for improvement, which can be done using different techniques like Contrastive Learning, Domain-Adversarial Neural Networks etc which make use of the target domain and source domain features (which was not allowed in this project).Further if we make use of other non linear machine learning techniques like Artificial Neural Networks or Decision Trees etc, the models accuracy can be improved significantly.

	dataset 1	dataset 2	dataset 3	dataset 4	dataset 5	dataset 6	dataset 7	dataset 8	dataset 9	dataset 10
model 1	0.8652									
model 2	0.8648	0.8672								
model 3	0.8600	0.8628	0.8648							
model 4	0.8564	0.862	0.86	0.8512						
model 5	0.8556	0.8584	0.8564	0.8496	0.856					
model 6	0.8512	0.8568	0.8544	0.8492	0.8552	0.8664				
model 7	0.8484	0.854	0.852	0.8484	0.854	0.8632	0.8408			
model 8	0.8452	0.85	0.848	0.844	0.8504	0.8588	0.8384	0.8408		
model 9	0.8412	0.846	0.8468	0.844	0.8488	0.8568	0.8348	0.8404	0.8392	
model 10	0.8392	0.844	0.8432	0.842	0.8432	0.8552	0.8336	0.8356	0.8376	0.8472

Figure 1: Final accuracy Table for Task 1

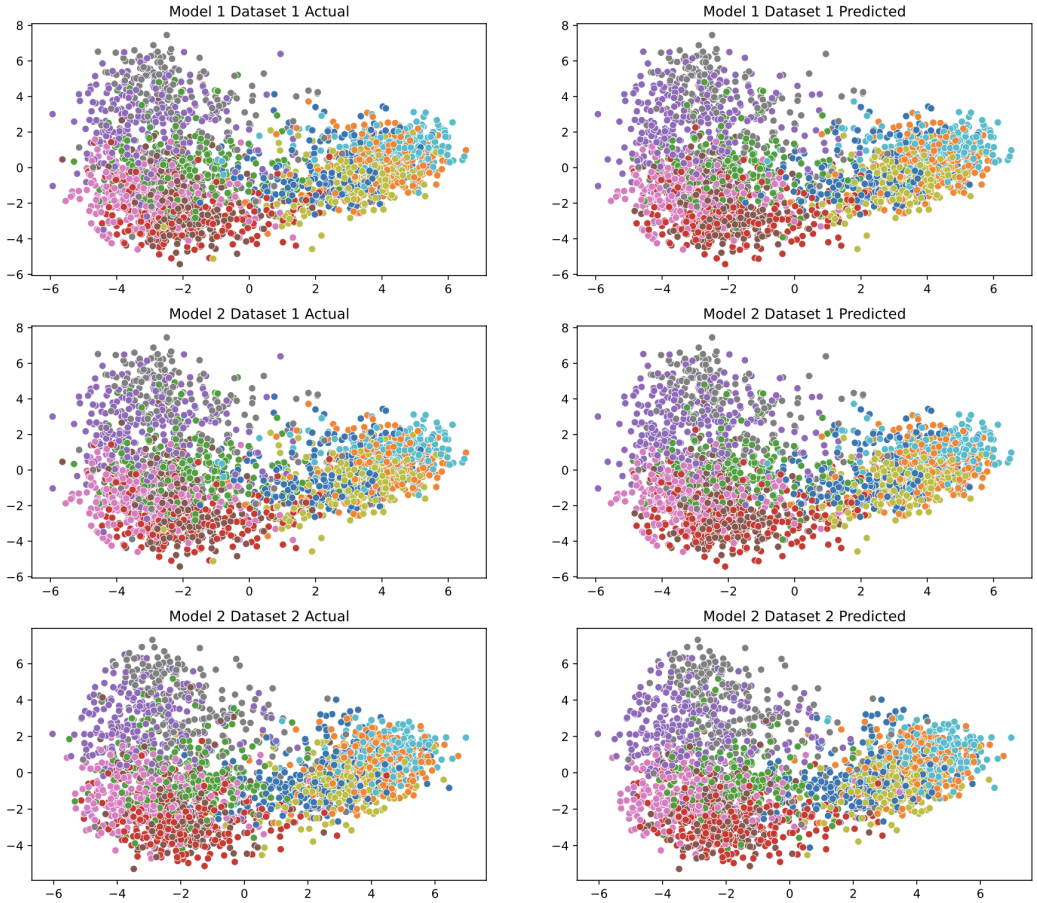

Figure 2: Actual vs Predicted visualization of High dimensional data to 2D feature space: Task 1

We also plotted different Model's performance across different Datasets (all 55 plots are present in the Jupyter notebook output, and is also submitted along with this report) A sample of such plots are shown in Figure 2. Here we see 10 different colors representing different classes/labels of the dataset. Left columns represent actual labels and left columns represent counterpart prediction la-

bels for each dataset tested with each model (mentioned in respective title). We can see models learn different distributions of non spherical shapes thanks to Mahalanobis distance covariance matrix.

1.2 Task 2

	DS 1	DS 2	DS 3	DS 4	DS 5	DS 6	DS 7	DS 8	DS 9	DS 10	DS 11	DS 12	DS 13	DS 14	DS 15	DS 16	DS 17	DS 18	DS 19	DS 20
model 11	0.678	0.682	0.678	0.673	0.688	0.700	0.670	0.665	0.670	0.696	0.571									
model 12	0.676	0.687	0.682	0.678	0.698	0.701	0.670	0.671	0.670	0.698	0.564	0.37								
model 13	0.675	0.682	0.678	0.678	0.697	0.700	0.670	0.668	0.670	0.696	0.563	0.37	0.546							
model 14	0.673	0.681	0.675	0.675	0.692	0.698	0.666	0.668	0.667	0.696	0.560	0.369	0.544	0.64						
model 15	0.671	0.680	0.672	0.672	0.687	0.693	0.663	0.666	0.665	0.693	0.558	0.366	0.542	0.638	0.688					
model 16	0.670	0.676	0.670	0.669	0.680	0.691	0.662	0.665	0.664	0.691	0.554	0.363	0.538	0.636	0.683	0.51				
model 17	0.665	0.672	0.668	0.665	0.676	0.690	0.658	0.662	0.662	0.687	0.551	0.364	0.536	0.632	0.682	0.505	0.633			
model 18	0.665	0.671	0.666	0.662	0.674	0.686	0.655	0.661	0.659	0.685	0.549	0.364	0.536	0.632	0.68	0.503	0.63	0.508		
model 19	0.662	0.669	0.662	0.662	0.672	0.684	0.654	0.659	0.656	0.683	0.548	0.363	0.538	0.632	0.679	0.5	0.626	0.507	0.502	
model 20	0.660	0.666	0.657	0.660	0.669	0.684	0.650	0.658	0.651	0.681	0.544	0.363	0.534	0.63	0.674	0.5	0.624	0.506	0.503	0.651

Figure 3: Final accuracy Table for Task 2

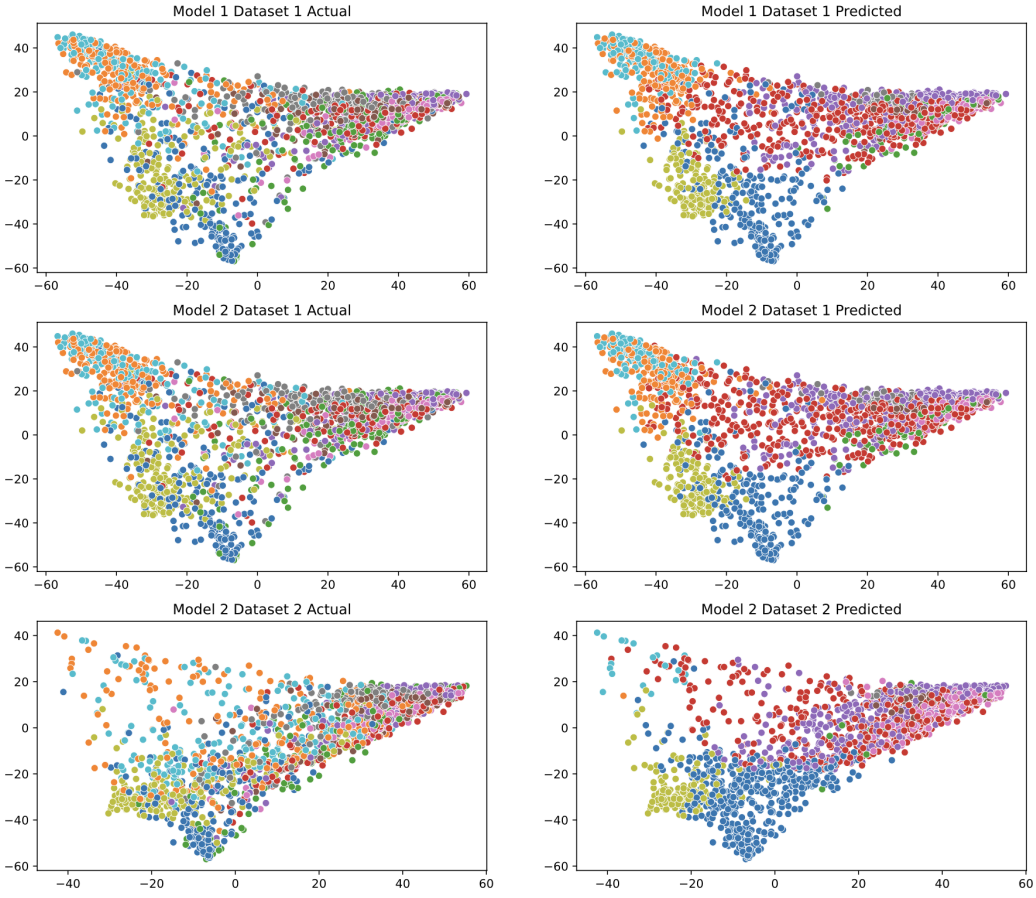

Figure 4: Actual vs Predicted visualization of High dimensional data to 2D feature space: Task 2

For Task 2, we first got our prediction labels from Task 1 Model 10 as per features extracted using same feature extractor as Task 1. For Task 2, we knew that all datasets are from different distributions. Hence to handle it (while maintaining the constraint to not use two dataset features simultaneously in Domain Adaption techniques), we used a Uniform Manifold Approximation and Projection (UMAP) model (we tried T-SNE as well to compare the clustering results done by both, but we could not fit a tsne model and use it for feature transformation, hence stayed with umap. we also Kernel PCA with RBF kernel and polynomial kernel with different degrees and sigmoid as

well to see if we can separate the different classes in a more general manner to handle the different distributions across datasets) with hyperparameters tuned to give us final features for our datasets. With this we were able to tackle the different distribution issue to some extent, and further we used weighted update of class prototypes and covariance matrix (which was calculated for each dataset as each dataset is different distribution, hence calculating its covariance matrix and weighted averaging it, will help generalize the models better).

This was further coupled with L2 regularization on class prototypes. On modeling the LWP models sequentially, following accuracies and table in Figure 3 was formed.(please note the value in ratio, example 0.7 represent 70% accuracy, and so on. Also Column "DS x" means "Dataset number"). We got a 95%+ training accuracy, but not so good test accuracy because the labels we got were from a model trained on different distribution and with 85% test accuracy, hence as a waterfall effect it affected Task 2 accuracies as well. There is scope for improvement same as task 1. We also plotted different Model's performance across different Datasets. A sample of such plots are shown in Figure 4. In these plots, same format for different classes and models is followed as done in Task 1. In the image, Model 1 represents first model in Task 2 i.e Model 11 to be precise, likely Model 2 is Model 12 (only in Figure 4) Clearly since the actual data is very much scattered, our predictions limited to learning from Model 10 and LWP variant with not much of domain shift handling techniques applied, gave clear boundaries as shown.

2 Problem 2

Below is the link to the Video Presentation as instructed for Problem 2 of the project. Please note we have used equations and reference images from the paper itself, and have given reference in the video, its description and here in report as well for same. [3]
Group 13 Video presentation on Paper: Deja vu: Continual model generalization for unseen domains

Acknowledgments

different libraries(sklearn,scipy,keras, umap,torch, tensorflow) [6],[7],[2],[4],[5],[1], [8] used in coding

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Francois Chollet et al. Keras, 2015.

[3] Chenxi Liu, Lixu Wang, Lingjuan Lyu, Chen Sun, Xiao Wang, and Qi Zhu. Deja vu: Continual model generalization for unseen domains, 2023.

[4] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[5] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[6] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Per-

rot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[7] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[8] Ross Wightman. Efficientnet pytorch. `https://github.com/lukemelas/EfficientNet-PyTorch`, 2019. Accessed: 2024-11-26.