CS771- Intro to ML (Autumn 2024): Mini-project 2
Group 53

Authors- Anya Rajan 220191 | Ananya Baghel 220136 | Nandini Akolkar 220692 | Shriya Garg 221038 | Shreya Shree 221029

November 26, 2024

1 Problem 1

In this task we were given 20 datasets consisting of 32x32 input images. The first 10 datasets (D1 to D10) share a common input distribution, while the remaining 10 datasets (D11 to D20) come from distinct but somewhat similar input distributions to D1-D10. Only D1 is labeled, while the others are unlabeled. Our first task was to check the distribution of the labels in dataset 1, and check if the distribution is skewed, which it is not. Ref. Fig1.

1.1 Task 1

In this task we were required to train an LwP classifier f_1 on labeled dataset D_1, then iteratively update f_i using predictions on D_{i+1} while ensuring model size remains constant. Evaluate models f_1 to f_{10} on held-out datasets.

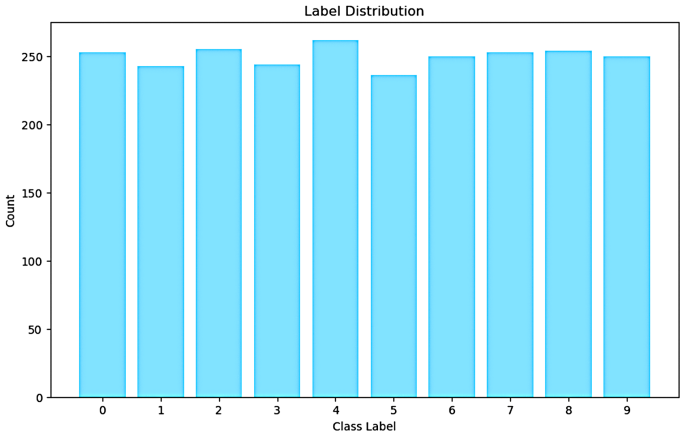

Figure 1: Label frequency in training data

1.1.1 Pre-processing

- Features for datasets D_1 to D_{10} were extracted using the pre-trained ResNet50 model trained on ImageNet.
- The fully connected layer was removed, and the penultimate layer's output was used as feature representation. Images were resized to 224×224, normalized using ImageNet statistics (mean =[0.485, 0.456, 0.406] , std=[0.229, 0.224, 0.225], and converted to tensors.
- While alternatives like VGG16 or handcrafted features were considered, ResNet50 was chosen for its proven performance. Extracted features were stored to optimize iterative training.
- PCA is used to reduce the dimensions and speed up the model.

1.1.2 Training the Lwp Classifier

- The LWP Classifier was implemented to iteratively train and update models f_1 to f_{10}. For the first dataset D_1, the classifier was trained using the extracted features and labels. Subsequently, the trained model f_i was used to predict labels for D_{i+1}.
- These predicted labels, along with the features from D_{i+1}, were used to update the classifier to f_{i+1}.
- The updated models were evaluated on held-out datasets D_1 to D_{10}, with the results recorded in an accuracy matrix.
- A Mahalanobis distance-based classifier is employed to classify images. This classifier works by computing the distance of each feature vector to the centroids of each class, using the inverse of the pooled covariance matrix as the metric. The class with the smallest distance is selected as the predicted class.
- We also tried using Euclidean Distances in LwP, but got 79 % accuracy. This shows that the shapes of the classes may not be uniform.

1.1.3 Final Result

The final result is presented as an accuracy matrix (Figure 2) where rows represent the models (f_1 to f_{10}) and columns represent the datasets (D_1 to D_{10}). The matrix provides insights into how well each model generalizes across datasets, highlighting any potential degradation in performance on prior datasets as models are updated. There is no significant degradation of accuracy at each iteration and the accuracy is 85-88% across all datasets and models.

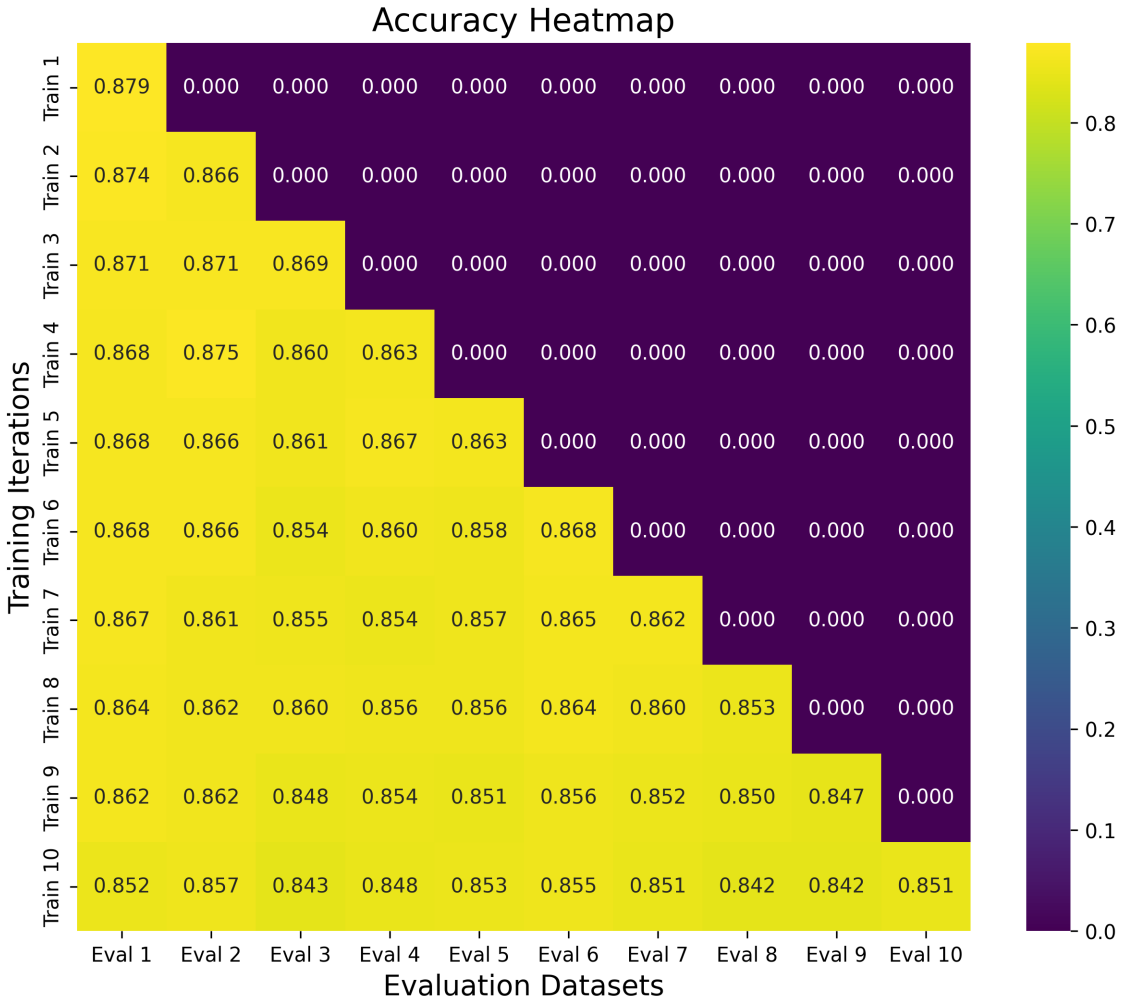

Figure 2: Accuracy Matrix

1.2 Task 2

In Task 2, starting with the model f_{10} from Task 1, the goal is to iteratively update the model using datasets D_{11} to D_{20}, which may have different input distributions. The updated models f_{11} to f_{20} should be evaluated on the held-out datasets, ensuring that performance on previous datasets does not degrade significantly. Accuracy results should be presented in a matrix format with models and datasets.

1.2.1 Pre-processing

- Data has been pre-processed the same way as done in task 1.
- The data has been resized and normalized.
- Here we have used ConvNeXt as a feature extractor in a transfer learning setup, enabling the extraction of meaningful, high-level embeddings (features) from images. These embeddings are then used by the downstream classifier (e.g., the LearningWithPrototypes classifier) to perform the task of classification, this was not used in task 1 beacuse ResNet gave a better output.

1.2.2 LwP classifier

- **Base Model**: The model f10, trained in Task 1 on datasets D1 to D10, served as the starting point for Task 2.
- The **Learning with Prototypes (LwP)** classifier was used. This lightweight framework represents each class using prototypes, computed as the mean of the embeddings for samples belonging to that class.

- The dataset is in the form of raw images (32x32 pixels), and transformations are typically required to prepare the data for neural network training. The permute ensures that the data has the correct dimension order for the model to process.
- In the incremental learning setup, pseudo-labels are generated using the most recent model's predictions. This allows you to iteratively update the model by training it on unlabeled data with the generated pseudo-labels.
- Acknowledging the distinct distributions of D11 to D20, the reliance on compact, generalizable embeddings ensured robustness during pseudo-labeling.
- Each model is updated incrementally, and the new model is trained with the latest pseudo-labels. This helps in adapting to new data while preserving the previous models' knowledge.

1.3.3 Results

- Models trained later in the sequence (e.g., f20) retained performance on earlier datasets, showcasing the robustness of the LwP framework.
- Performance dips were observed for datasets with significant differences in $p(x)$ (e.g., D12, D15), emphasizing the challenges of pseudo-labeling in such cases.
- The accuracies of the the D11-D20 dataset was lower as compared to those of D1-D10, due to more variation in thier shapes.
-

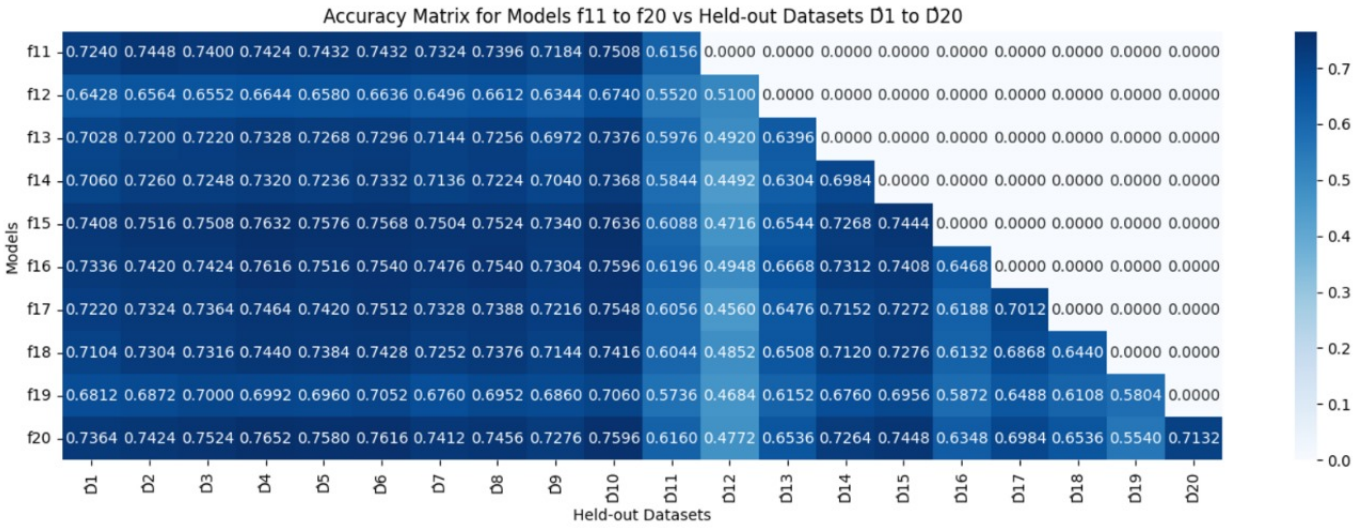

Figure 3: Accuracy Matrix for Task 2

2 Problem 2

The video link

3 References

References

[1] ResNet. https://www.kaggle.com/code/shamiulislamshifat/playing-with-resnet50-full-tutorial

[2] https://medium.com/@meetkalathiya1301/feature-extraction-using-pre-trained-models-for-image-classification-16e6ff43f268

[3] https://www.geeksforgeeks.org/how-to-draw-2d-heatmap-using-matplotlib-in-python/

[4] https://tinyurl.com/cs771-mp2-data.

[5] Pytorch Documentation https://pytorch.org/docs/stable/index.html

[6] TorchVision https://pytorch.org/vision/stable/index.html

[7] ConvNeXt Torchvision Models https://pytorch.org/vision/main/models/convnext.html

[8] Scikit-learn: https://scikit-learn.org/stable/

[9] TQDM: https://tqdm.github.io/