Machine Learning Mini Project-2

TEAM: Parameter Hunters

Author	**Roll No**
Pokala Dattatreya	241110048
Voora Nagendra Bhaskara Swamy	241110044
Telugu Sudhakar	241110077
Edula Vinay Kumar Reddy	241110024
Tippireddy Yashwanth	241110082

1 Problem 1

1.1 TASK 1

1.1.1 Features Extraction

In Features Extraction, We Resize each image to 224*224 because input size for feature extractors like ViT or Resnet is 224*224. Then we normalized it using mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225] because it helps model to learn better. This normalizes the tensor by subtracting given mean and dividing by the given standard deviation for each color channel(RGB).
We have evaluated various feature extraction methods on dataset-1 to determine the most effective model for our task. Below are the respective accuracies achieved by each method: Among all the

Method	**Accuracy**
Swin-Transformer	0.24%
AlexNet	57.36%
VGG16	60.72%
DPN92	61.68%
InceptionV3	62.28%
ResNet50	63.52%
DenseNet121	63.96%
RegNetY-400MF	64.00%
MobileNetV2	66.48%
ResNet101	67.32%
EfficientNet-B3	69.76%
SENet154	69.24%
ConvNeXt-Base	75.80%
EfficientNet-B0	79.88%
ViT-Base	**84.68%**

Table 1: Performance of Various Feature Extraction Methods

models tested, **ViT-Base** achieved the highest accuracy of **84.68%**, demonstrating its superior performance in feature extraction for our task and this ViT-Base model was not trained on the CIFAR-10 image dataset. Therefore, we have selected **ViT-Base** as the preferred model for feature extraction in the subsequent stages of our analysis.
Features Extraction Using Vision Transformer: The task begins with the extraction of features from the dataset using the **Vision Transformer (ViT)** model. This model processes input data (e.g., images) and converts them into feature vectors that can be used for further modeling.
Our feature extraction file has been uploaded to Google Drive. Please find it: **here**

1.1.2 LWP variants exploration

1. **LWP Hard on model:**
   Using Learning with prototypes hard method where means is calculated based on points belonging to that class only would result in 83.84% accuracy for dataset1. While testing here each point is assigned to class based on distance to each class. It is assigned to the class whose mean is less distant to the data point.

2. **LWP Soft on model:**
   Using Learning with prototypes, a soft method where probability of each point belonging to each class is calculated and used for calculation of mean resulted in 74.04% accuracy. While testing here each point is assigned to class based on distance to each class. It is assigned to the class whose mean is less distant to the data point.

3. **LWP Hard with Mahalanobis distance:**
   Using Learning with prototypes hard method for calculating mean and used mahalanobis distance for the calculation of predicted class resulted in 100% accuracy for the model1.

1.1.3 Final Model:

- **LwP Hard (Model f1)**:
  - We used above LWP Hard method for model1.
  - To simulate randomness in the data Gaussian noise is added to the features of each class.
  - While calculating the mean/prototype for each class, smoothing is applied by incorporating the global mean (i.e., the mean of all data points across all classes). The purpose of this smoothing is to prevent the prototype from being overly sensitive to noisy or specific characteristics of the data, which could make it less generalizable.
  - A small regularization value is applied to the covariance matrix. This prevents the covariance matrix from becoming singular (non-invertible), which could occur if the matrix is poorly conditioned. This ensures that the covariance matrix remains stable and preserves the original dependencies or correlations between features without distorting them.

- **Model Update (f2 to f10) LwP Soft**
  - From model f2 to f10, the LwP Soft approach is used. Here, each data point has a probability of belonging to all classes, rather than being restricted to a single class.
  - The probability of each data point belonging to each class is computed based on a soft assignment. These probabilities are used to update the mean and covariance for each class.
  - Here mean/prototype is calculated as weightage average of each point feature values where particular point probability of belonging to particular class is taken as weight.
  - Same approach is applied while calculating the covariance matrix of each class.

$$Z_{nk} = \frac{\exp(-\|x_n - \mu_k\|^2)}{\sum_{l=1}^{K} \exp(-\|x_n - \mu_l\|^2)} \quad \forall n, k$$

$$\mu_k = \frac{\sum_{n=1}^{N} Z_{nk} x_n}{\sum_{n=1}^{N} Z_{nk}} \quad \forall k$$

Z_{nk} : probability of the n-th data point belonging to the k-th class

μ_k : mean of the k-th class

  - **Updating the Mean:**

* The mean/prototype for each class is updated by combining the previous mean with the new mean calculated using the current data. The update is done with a weighting scheme where:
  - 0.8 weightage is given to the previous mean (to retain information from earlier models).
  - 0.2 weightage is given to the new mean (to incorporate new information).

  Note : these weights are found optimal after making trials using different alpha values(weightage given to new class mean).

- **Updating the Covariance Matrix:**
  * The covariance matrix is also updated using the same weighting method. This helps to smoothly incorporate new information from the data while retaining the old data's influence.
  * Like in the LwP Hard case, a small regularization value is applied to the covariance matrix to prevent it from becoming singular. This ensures that the covariance matrix remains well-conditioned, which is crucial for accurately capturing the dependencies and correlations between the features in the dataset.

- **Prediction**
  - In the prediction stage, the Mahalanobis distance is used to calculate how close each data point is to each class prototype.
  - For each data point, the distance to each class is computed. The class with the minimum Mahalanobis distance (i.e., the class with the closest prototype) is predicted as the correct class for the data point.

1.1.4 Result

- **Accuracy Matrix:**

Model/Dataset	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9	D_{10}
f_1	99.96									
f_2	99.96	86.88								
f_3	99.96	86.96	87.32							
f_4	99.96	86.84	87.16	87.48						
f_5	99.96	86.80	87.32	87.40	87.44					
f_6	99.96	86.72	87.44	87.16	87.52	87.36				
f_7	99.96	86.72	87.40	87.16	87.48	87.40	86.24			
f_8	99.96	86.64	87.20	87.12	87.32	87.36	86.16	86.92		
f_9	99.96	86.64	87.28	87.20	87.44	87.32	86.20	86.88	86.12	
f_{10}	99.96	86.64	87.20	87.04	87.32	87.60	86.04	86.96	86.00	87.56

- **Confusion Matrix**(by running f_{10} model on D_{10} Dataset):

$$\begin{bmatrix} 228 & 2 & 1 & 1 & 1 & 0 & 1 & 0 & 7 & 3 \\ 4 & 226 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 15 \\ 10 & 1 & 202 & 6 & 6 & 4 & 4 & 1 & 0 & 0 \\ 2 & 1 & 3 & 211 & 5 & 31 & 16 & 0 & 0 & 1 \\ 2 & 1 & 5 & 6 & 189 & 1 & 12 & 3 & 0 & 0 \\ 2 & 0 & 0 & 27 & 2 & 212 & 2 & 6 & 0 & 0 \\ 0 & 0 & 8 & 15 & 3 & 6 & 220 & 0 & 0 & 0 \\ 2 & 2 & 1 & 6 & 7 & 8 & 1 & 211 & 1 & 2 \\ 9 & 6 & 0 & 1 & 0 & 0 & 0 & 0 & 264 & 4 \\ 5 & 20 & 2 & 3 & 0 & 0 & 0 & 0 & 2 & 226 \end{bmatrix}$$

1.2 TASK 2

1.2.1 First Trial (Generative replay):

In this method, we use a functionality that generates new data samples based on the prototypes (means) and covariance matrices of each class. These prototypes help the model retain knowledge about older data points as new data points are introduced. During training, we append these generated features to the new features and train the model using the same procedure as in Task 1.By using this method we achieved accuracy of 60.3% for data set 1 on model f_{20}.

1.2.2 Second Trial (T^2PL & RandMix):

In this method, T^2PL selects samples based on the confidence level of each class, which is measured by computing softmax scores. We then select the top samples with the highest confidence to construct class means. Using these selected samples, we calculate the mean of their representations and generate pseudo labels for the remaining data points by measuring their similarity to the means. Next, we apply **RandMix**, a data augmentation technique that creates new training samples by combining two or more existing samples using a weighted average. Finally, we train the model again, following the same procedure as in Task 1..By using this method we achieved accuracy of 81.7% for data set 1 on model f_{20}.

1.2.3 class_weighted_clustering :

We updated the logic while training the model that uses the class_weighted_clustering functionality to compute a weight for each feature based on the inverse of the squared Mahalanobis distance.

$$\text{Min\_distance} = \min_{c \in \{1,2,\ldots,10\}} \left(\sqrt{(x - \mu_c)^T \Sigma^{-1} (x - \mu_c)} \right)$$

$$\text{Weight} = \frac{1}{1 + (\text{Min\_distance})^2}$$

This approach provides the advantage that more importance is given to the points closer to the prototypes while updating the prototypes and covariance matrices. In contrast, points that are farther away, or potential outliers, receive less influence. This ensures that the model focuses more on the data points representative of each class, leading to more accurate updates. we train the model following the same procedure as in Task 1 (i.e LwP Hard for f_1 and LwP Soft for f_2 to f_{20} with modification of updating the mean and covariance matrix we use the weight which we got from class_weighted_clustering).By using this method we achieved accuracy of 95.72% for data set 1 on model f_{20} and compared to other two methods this method outperformed even in remaining data sets while updating model from f_2 to f_{20}.

- **Updating the Mean and Covariance Matrix :**
  - The mean/prototype for each class is updated by combining the previous mean with the new mean calculated using the current data. The update is done with a weighting scheme where:
    - 0.9 weightage is given to the previous mean (to retain information from earlier models).
    - 0.1 weightage is given to the new mean (to incorporate new information).

    The covariance matrix is also updated using the same weighting method. This helps to smoothly incorporate new information from the data while retaining the old data's influence.

    Note : these weights are found optimal after making trials using different alpha values(weightage given to new class mean).

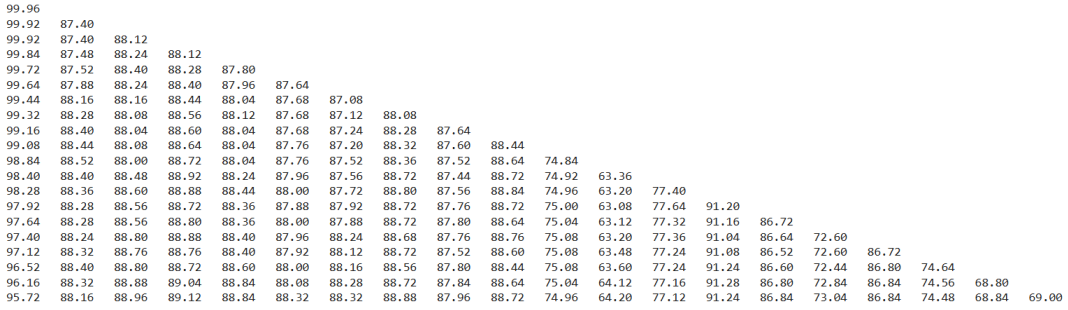

Figure 1: Accuracy Matrix for Task-2.

1.2.4 Result

- **Accuracy Matrix:**
- **Confusion Matrix**(by running f_{20} model on D_{20} Dataset):

$$\begin{bmatrix} 190 & 7 & 2 & 5 & 1 & 0 & 7 & 0 & 22 & 4 \\ 5 & 218 & 1 & 11 & 0 & 0 & 10 & 0 & 12 & 5 \\ 30 & 1 & 157 & 13 & 5 & 0 & 48 & 1 & 1 & 0 \\ 8 & 4 & 4 & 183 & 2 & 19 & 28 & 1 & 1 & 5 \\ 14 & 9 & 14 & 20 & 118 & 3 & 75 & 6 & 9 & 0 \\ 6 & 3 & 10 & 51 & 1 & 146 & 25 & 1 & 3 & 0 \\ 6 & 3 & 5 & 10 & 2 & 2 & 196 & 0 & 1 & 0 \\ 15 & 10 & 2 & 15 & 10 & 14 & 15 & 168 & 6 & 4 \\ 15 & 12 & 0 & 5 & 1 & 1 & 3 & 0 & 197 & 2 \\ 10 & 51 & 0 & 18 & 0 & 2 & 6 & 0 & 16 & 152 \end{bmatrix}$$

Note : For Task-2 we used GPU and it may give warning which can be ignored as it doesn't affect the final output.

2 Problem 2

For this task, we read and analyzed the paper titled **"Deja vu: Continual Model Generalization for Unseen Domains"** (ICLR 2023). we created a presentation summarizing the key problem studied in the paper, the novel ideas proposed, and the results presented by the authors.

The presentation includes slides along with a voice-over, providing a concise overview of the paper's contributions and its significance in the field of continual/lifelong unsupervised domain adaptation.

The video has been uploaded to YouTube, and can be accessed via the following link: **Click here to view the video**

3 Submission

We have given python notebooks for Task-1, Task-2 and feature extraction and also PPT.