
Mini-project 2

Varun Tandon
Roll No. 211151

Gary Derrick Anderson J
Roll No. 210383

Bhavesh Shukla
Roll No. 210266

1 Problem 1

We were provided with 20 training datasets, D_1, D_2, \dots, D_{20} , each being a subset of the CIFAR-10 image classification dataset. Among these, the first 10 datasets (D_1 to D_{10}) have inputs from the same distribution, implying that the input features of these datasets share similar characteristics ($p(x)$ is the same across D_1 to D_{10}). In contrast, datasets D_{11} to D_{20} are derived from different input distributions ($p(x)$ varies for these datasets). However, the input distributions of D_{11} to D_{20} exhibit a certain degree of similarity to the common input distribution shared by D_1 to D_{10} .

The datasets D_1 to D_{10} and D_{11} to D_{20} were provided in two separate folders. Each dataset contained raw 32×32 color images. We were allowed to use any suitable feature representation for the inputs, such as leveraging a pre-trained neural network for feature extraction or even employing kernel-based methods.

Notably, only the first dataset (D_1) was labeled, while the remaining 19 datasets (D_2 to D_{20}) were unlabeled. Additionally, we were provided with 20 held-out labeled datasets (D_1 to D_{20}) to evaluate the model's performance (accuracy). However, we were explicitly instructed not to use these held-out datasets for any other purpose, such as cross-validation or model tuning.

1.1 TASK 1

Role of ResNet50

ResNet50, a deep convolutional neural network pre-trained on ImageNet, is used as the feature extractor. It processes the 32×32 images and outputs a fixed-length feature vector (2048 dimensions). These features serve as inputs to the Kernel LwP classifier.

Why ResNet50?

- **Pretrained Network:** ResNet50 is trained on a massive dataset (ImageNet), giving it a strong ability to extract meaningful features across various visual tasks.
- **Generalization:** The learned representations generalize well to new datasets, such as CIFAR-10 subsets in this task.
- **Pooling Layer:** The global average pooling layer in ResNet50 reduces spatial dimensions, providing a compact 2048-dimensional feature representation, ideal for prototype-based classification.

How It's Used?

- **Feature Extraction:** The `extract_features` function processes batches of input images through ResNet50, ensuring efficient memory usage.

- **Image Preprocessing:** Images are preprocessed using `preprocess_input`, which normalizes the pixel values to match the range expected by ResNet50.

Role of Kernel Learning with Prototypes (Kernel LwP) Classifier

The Kernel LwP classifier classifies samples based on their similarity to class prototypes in the feature space. Instead of using Euclidean distance, it computes RBF kernel similarity between input features and class prototypes.

Why Kernel LwP for Task 1.1?

- **Prototype-Based Representation:** This approach aligns with the task requirement of incremental model updates without retraining from scratch.
- **Kernel Similarity:** The RBF kernel introduces non-linearity, capturing more complex relationships between data points and prototypes.
- **Incremental Updates:** The prototypes are updated incrementally as new datasets (D_2, \dots, D_{10}) are encountered, making it suitable for continual learning.

Key Components in Kernel LwP Classifier:

- **Prototypes:**
 - Each class is represented by a prototype, initialized as the mean feature vector of the class during training on D_1 .
 - Prototypes are updated incrementally using high-confidence pseudo-labeled data from D_2, \dots, D_{10} .
- **RBF Kernel Similarity:**
 - The classifier computes the similarity between input features and prototypes using the Radial Basis Function (RBF) kernel:

$$K(x, y) = \exp(-\gamma \|x - y\|^2)$$

- The hyperparameter γ controls the sensitivity of the kernel. Smaller γ values focus on global structure, while larger values emphasize local similarity.
- **Cross-Validation for Gamma Selection:**
 - The optimal γ is determined using 5-fold cross-validation on D_1 , ensuring that the kernel's behavior is well-suited to the data.
 - Candidate γ values $\{0.0001, 0.001, 0.01, 0.1, 1\}$ are evaluated, and the one yielding the highest accuracy is selected.
- **Incremental Updates:**
 - The `update_prototypes` method updates prototypes based on pseudo-labeled samples, using a running average to integrate new data without overwriting past knowledge.

	1	2	3	4	5	6	7	8	9	10
1	0.4860	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.000
2	0.4820	0.4904	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.000
3	0.4756	0.4876	0.4880	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.000
4	0.4772	0.4880	0.4868	0.4984	0.0000	0.0000	0.0000	0.0000	0.0000	0.000
5	0.4752	0.4872	0.4868	0.4980	0.4772	0.0000	0.0000	0.0000	0.0000	0.000
6	0.4732	0.4864	0.4844	0.4968	0.4736	0.4956	0.0000	0.0000	0.0000	0.000
7	0.4728	0.4856	0.4844	0.4956	0.4732	0.4940	0.4760	0.0000	0.0000	0.000
8	0.4720	0.4840	0.4836	0.4952	0.4736	0.4936	0.4768	0.4836	0.0000	0.000
9	0.4720	0.4840	0.4828	0.4948	0.4744	0.4932	0.4768	0.4836	0.4764	0.000
10	0.4716	0.4836	0.4836	0.4940	0.4732	0.4936	0.4756	0.4832	0.4756	0.492

Figure 1: Accuracy matrix for Task 1

1.2 Task 2

This section outlines the implementation of **Task 1.2**, focusing on adapting the *Kernel LwP Classifier* trained on D_1 to D_{10} to handle D_{11} to D_{20} . The classifier is evaluated incrementally on the union of all held-out datasets ($\hat{D}_1, \dots, \hat{D}_{20}$). The workflow balances model updates while ensuring prior knowledge retention.

Key Components

ResNet50 Feature Extractor

Pre-trained **ResNet50** serves as the feature extractor, processing the input images and outputting a fixed-size feature vector (2048 dimensions) for each sample. Extracted features are input to the *Kernel LwP Classifier* for classification.

Kernel LwP Classifier

Prototypes represent **class means** in the feature space. Classification is based on *Euclidean distance* or *kernel similarity* between features and prototypes. Prototypes are updated incrementally using **high-confidence pseudo-labeled data**.

Pseudo-Labeling

Assigns labels to **unlabeled data** using the current model. **Confidence-based filtering** ensures that only reliable samples are used to update prototypes.

Accuracy Matrix

Tracks **model accuracy** for f_{11} to f_{20} on the union of held-out datasets ($\hat{D}_1, \dots, \hat{D}_{20}$).

Workflow

1. Initialize ResNet50 for Feature Extraction

The ResNet50 model is initialized with:

- `weights='imagenet'`: Leverages pre-trained weights from the ImageNet dataset for robust feature extraction.
- `include_top=False`: Excludes the classification head, allowing customization for CIFAR-10.

- `pooling='avg'`: Adds a global average pooling layer, reducing the spatial dimensions to a 2048-dimensional vector.
- **Functionality:** The `extract_features` function preprocesses input images, passes them through ResNet50, and outputs feature vectors for classification.

2. Pseudo-Labeling with Confidence Filtering

[leftmargin=1.5cm]**Pseudo-labeling pipeline** (`pseudo_label_dataset`):

- – **Distance Calculation:** Computes the Euclidean distance between input features and class prototypes.
- **Confidence Scores:** Inversely proportional to the distance to the nearest prototype.
- **Filtering:** Normalizes confidence scores and selects samples with confidence above a threshold (0.8).
- **Why Confidence Filtering?** Ensures that only reliable pseudo-labels contribute to prototype updates, reducing noise.

3. Training Loop for Task 1.2

[leftmargin=1.5cm]The loop updates the model incrementally on D_{11} to D_{20} , starting from the model trained on D_1 to D_{10} . **Steps:**

- 1. **Initialization:** Prototypes are inherited from the model trained on D_1 to D_{10} .
- 2. **Iterative Updates:** For each dataset D_{i+11} ($i = 0, \dots, 9$):
 - Extract features using ResNet50.
 - Generate pseudo-labels using the classifier’s predictions.
 - Update prototypes incrementally using high-confidence pseudo-labeled samples.
- 3. **Evaluation:** After training on D_{i+11} , the updated model is evaluated on all held-out datasets ($\hat{D}_1, \dots, \hat{D}_{20}$).
- 4. Results are stored in a **10x20 accuracy matrix**, where:
 - **Row i :** Model trained on D_{i+11} .
 - **Column j :** Accuracy on \hat{D}_j .

Key Design Choices for Task 1.2


[leftmargin=1.5cm]**Kernel LwP Classifier:**

- – Retains previously learned knowledge via prototype updates.
- Effectively handles distribution shifts between D_{11}, \dots, D_{20} using kernelized similarity.
- **Incremental Updates:**
 - Prototypes are updated incrementally, avoiding the need to store or re-train on previous datasets.
- **Union of Held-Out Datasets:**
 - Evaluating on all held-out datasets tests the model’s ability to generalize while retaining prior knowledge.
- **Confidence Filtering:**
 - Ensures robust pseudo-labeling, crucial as the training data (D_{11} to D_{20}) is unlabeled.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	0.4700	0.4844	0.4840	0.4948	0.4720	0.4928	0.4748	0.4800	0.4756	0.4912	0.4288	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.00
2	0.4704	0.4840	0.4848	0.4936	0.4724	0.4924	0.4744	0.4796	0.4760	0.4904	0.4284	0.3328	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.00
3	0.4700	0.4832	0.4848	0.4940	0.4720	0.4940	0.4752	0.4788	0.4764	0.4908	0.4276	0.3332	0.4180	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.00
4	0.4700	0.4840	0.4868	0.4936	0.4720	0.4924	0.4752	0.4788	0.4760	0.4908	0.4280	0.3336	0.4180	0.3588	0.0000	0.0000	0.0000	0.0000	0.0000	0.00
5	0.4704	0.4836	0.4868	0.4936	0.4720	0.4932	0.4756	0.4792	0.4756	0.4908	0.4280	0.3336	0.4180	0.3592	0.5072	0.0000	0.0000	0.0000	0.0000	0.00
6	0.4712	0.4836	0.4860	0.4936	0.4724	0.4940	0.4752	0.4792	0.4764	0.4900	0.4292	0.3336	0.4192	0.3588	0.5068	0.4024	0.0000	0.0000	0.0000	0.00
7	0.4712	0.4832	0.4860	0.4932	0.4716	0.4936	0.4764	0.4788	0.4772	0.4900	0.4296	0.3332	0.4200	0.3596	0.5068	0.4020	0.3720	0.0000	0.0000	0.00
8	0.4716	0.4832	0.4868	0.4932	0.4712	0.4928	0.4764	0.4780	0.4772	0.4900	0.4292	0.3332	0.4196	0.3592	0.5068	0.4020	0.3720	0.3872	0.0000	0.00
9	0.4716	0.4836	0.4864	0.4932	0.4716	0.4932	0.4760	0.4776	0.4768	0.4896	0.4292	0.3336	0.4188	0.3596	0.5072	0.4024	0.3716	0.3872	0.4296	0.00
10	0.4716	0.4840	0.4864	0.4932	0.4712	0.4928	0.4764	0.4776	0.4768	0.4900	0.4296	0.3336	0.4188	0.3600	0.5072	0.4020	0.3716	0.3864	0.4296	0.48

Figure 2: Accuracy matrix for Task 2

2 Problem 2

Click the following icon to watch the related video:  **Problem 2 video presentation**