CS771 : Intro. to ML Project

Sankul	Bhavishya Gupta	Harshit Gupta	Gautam Kumar
Roll No. 220965	Roll No. 220295	Roll No. 220439	Roll No. 220407

Ayush Patel
Roll No. 220269

1 Brief Introduction

We outline various methods and approaches employed towards incrementally learning across several subsets drawn from differing distributions of the CIFAR-10 dataset. Task 1 of the project (presented here) consists of two main subtasks:

- **Task 1.1**: Incrementally training models on datasets with a shared distribution.
- **Task 1.2**: Incrementally training models on datasets with varying distributions.

1.1 Problem Statement

We are given 20 training datasets $D_1, D_2, \ldots, D_{20}$ from CIFAR-10. The datasets are structured as follows:

- **Task 1**: Datasets D_1 to D_{10} have inputs from a similar distribution $p(x)$.
- **Task 2**: Datasets D_{11} to D_{20} have inputs from different distributions $p(x)$.

The first dataset, D_1, is labeled, while the remaining datasets are unlabeled. Our goal is to incrementally train models $f_1, f_2, \ldots, f_{10}$, where each model is trained using the predictions of the preceding model.

2 Our approach for Task 1 and Task 2.

2.1 Dataset Overview

The dataset is organized into two parts: *part One* and *part Two*. Each part contains several sub-datasets, denoted as D_1 to D_{20}. These datasets are in the form of .tar.pth files, containing image data and labels. The dataset is processed in batches, and for each batch, features are extracted using the ResNet-152 model.

2.2 Custom Dataset Class

A custom dataset class, `CIFARDataset`, is implemented to handle datasets with or without labels. The class provides methods to load and transform the data. The `transform` parameter allows for preprocessing the images, such as resizing, normalization, and converting to tensors.

- `CIFARDataset`: A custom PyTorch dataset for loading images and labels (if available).
- `DataLoader`: Used to load the dataset in batches for training and evaluation.

2.3 Feature Extraction with ResNet-152

To extract features, we leverage the pre-trained **ResNet-152** model from `torchvision.models`. The ResNet-152 model is a deep convolutional neural network that includes residual connections to address the vanishing gradient problem. We use the model as a feature extractor by removing the final classification layer, resulting in a feature vector for each input image.

2.4 ResNet-152 Architecture Overview

ResNet-152 is a deep network with 152 layers, utilizing residual blocks that help mitigate training challenges in very deep networks. Each residual block contains:

- Two convolutional layers with batch normalization and ReLU activation.
- A skip connection that adds the input directly to the output after the convolution operations.

The output of the feature extractor is a high-dimensional feature vector that represents the image in a more compact form.

2.5 Feature Extraction Process

The feature extraction is done in two stages:

1. **Training Features**: The training data is passed through the ResNet-152 model, and the output from the penultimate layer is collected as the feature vector.
2. **Unlabeled Data Features**: For unlabeled datasets, the feature extractor processes the images and outputs the feature vectors, which are later used for training the student model.

The extracted features are stored as tensors and used for subsequent classification tasks.

2.6 Model Distillation

The main training process involves the concept of *knowledge distillation*, where the teacher model (`f_prev`) provides soft targets to the student model (`f_i`). The distillation loss is calculated by comparing the softmax outputs of both models. This allows the student model to learn from the teacher model's predictions, improving its generalization.

2.7 Distillation Loss

The distillation loss is computed as the Kullback-Leibler divergence between the softmax outputs of the teacher and student models. The formula for distillation loss is:

$$\mathcal{L}\text{distill} = -\sum i = 1^{N} p_{\text{teacher}}(i) \log p_{\text{student}}(i)$$

where $p_{\text{teacher}}(i)$ and $p_{\text{student}}(i)$ are the softmax probabilities of the teacher and student models, respectively.

2.8 Regularization

L2 regularization is applied to the model's parameters to prevent overfitting. The regularization loss is calculated as:

$$\mathcal{L}\text{reg} = \lambda \sum i = 1^{N} \|\theta_i - \theta_i^{\text{prev}}\|^2$$

where θ_i are the parameters of the student model, and θ_i^{prev} are the parameters of the teacher model.

2.9 Training and Evaluation

The models are trained iteratively on datasets D_1 to D_{20}. The first model, f_1, is trained on D_1 using the extracted features. For subsequent datasets, the student model is trained using pseudo-labels generated by applying the teacher model to unlabeled data.

2.10 Training Pipeline

The training pipeline includes:

- Training the first model, f_1, on the labeled dataset D_1.
- For each subsequent dataset D_i, the teacher model generates pseudo-labels, which are then used to train the student model, f_i.
- Regularization and distillation losses are combined during the training process. The overall process for model updating from f_1 to f_2 and f_2 to f_3 can be broken down into the following steps:
- **Step 1:** Train f_1 on dataset D_1.
- **Step 2:** Use f_1 to generate pseudo-labels for the next dataset. Here we have used LwP model to generate the labels. D_2.
- **Step 3:** Train f_2 on dataset D_2 with the generated pseudo-labels, while applying knowledge distillation from f_1 (soft targets) and regularizing the model using L2 regularization to prevent forgetting.
- **Step 4:** Update the model f_2, and repeat the process for f_3 using the updated f_2 as the new teacher model for generating pseudo-labels and distilling knowledge.

2.11 Evaluation

After training each model, the accuracy is evaluated on the validation datasets, $D_{\hat{1}}$ to $D_{\hat{i}}$, using the softmax outputs from the student model.

3 Results

The accuracy matrix is generated by evaluating each model on the corresponding validation datasets. This allows for a comparison of the models' performance across different datasets.

	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10
Model f1	89.48	-	-	-	-	-	-	-	-	-
Model f2	89.60	89.56	-	-	-	-	-	-	-	-
Model f3	89.68	89.68	89.80	-	-	-	-	-	-	-
Model f4	89.56	89.48	89.72	89.12	-	-	-	-	-	-
Model f5	89.60	89.40	89.84	89.24	88.52	-	-	-	-	-
Model f6	89.44	89.56	89.80	89.20	88.40	89.16	-	-	-	-
Model f7	89.44	89.68	89.76	89.32	88.52	89.20	88.96	-	-	-
Model f8	89.28	90.00	89.96	89.56	88.72	89.04	89.04	88.76	-	-
Model f9	89.20	90.00	89.76	89.44	88.36	88.64	89.20	88.52	88.76	-
Model f10	89.28	89.96	89.68	89.32	88.36	88.72	89.08	88.76	89.00	89.52

Table 1: Accuracy for Models f1 to f10 (Task 1)

	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10
Model f11	88.92	89.48	89.60	89.24	88.28	88.56	88.76	88.64	88.80	89.08
Model f12	88.52	89.16	88.64	88.76	88.20	88.32	87.92	88.44	88.20	88.56
Model f13	88.24	89.44	88.36	88.44	87.96	88.00	87.60	88.20	88.00	88.64
Model f14	88.20	89.12	88.24	88.52	87.80	87.96	87.60	88.16	87.92	88.16
Model f15	88.32	89.00	88.28	88.24	87.64	88.04	87.60	87.72	87.44	88.40
Model f16	87.88	88.60	88.12	88.16	87.32	87.96	87.44	87.84	87.56	88.04
Model f17	87.84	88.60	87.88	88.20	87.28	87.76	87.36	87.64	87.48	87.76
Model f18	87.88	89.00	87.76	88.16	87.20	87.84	87.28	87.64	87.24	87.60
Model f19	87.68	88.88	87.52	87.92	87.04	87.76	87.20	87.48	87.08	87.44
Model f20	87.40	88.60	86.92	87.64	86.92	87.64	86.72	87.40	87.00	87.44

Table 2: Accuracy for Models f11 to f20 (d1 to d10) (Task 2)

	d11	d12	d13	d14	d15	d16	d17	d18	d19	d20
Model f11	71.24	-	-	-	-	-	-	-	-	-
Model f12	70.92	57.80	-	-	-	-	-	-	-	-
Model f13	71.00	58.00	77.00	-	-	-	-	-	-	-
Model f14	71.48	57.92	76.72	75.52	-	-	-	-	-	-
Model f15	71.44	58.36	76.80	75.52	87.24	-	-	-	-	-
Model f16	70.64	58.28	76.28	75.28	87.04	72.84	-	-	-	-
Model f17	70.76	58.68	76.80	75.20	87.24	72.72	71.08	-	-	-
Model f18	70.84	59.00	76.68	75.52	87.12	72.84	71.36	73.64	-	-
Model f19	70.64	58.68	76.16	75.48	87.20	72.36	70.80	73.12	67.24	-
Model f20	70.24	58.76	76.44	74.92	86.88	72.76	70.56	72.96	66.96	83.80

Table 3: Accuracy for Models f11 to f20 (d11 to d20) (Task 2)

Link to youtube video