CS-771 MINI-PROJECT-1 GROUP-49

Manan Kumar	Ayan Gupta	Anitej Jain	Kushal Bansal	Pranay Chaudhari
Roll No. 220607	Roll No. 220258	Roll No. 220153	Roll No. 220575	Roll No. 220794

October, 2024

1 PROBLEM - 1 TASK - 1

1.1 Insights

- It was found out that working with raw images without any feature extraction wasn't fruitful as we got accuracies near around 27 percent.
- We set up a Resnet34 feature extractor which is pretrained on the ImageNet dataset.
- We first tried with normal LwP as our main model i.e. non probabilistic but it didn't end of achieving accuracies near around 40 percent.
- We switched to probabilistic version of LwP that helped achieve accuracies near around 58 percent.
- We also worked with a primitive model which processed image with resnet50 and used squeezing in carrying this out. We got an accuracy of around 80 but the model was taking too long to run so to continue working we decided to work with a time efficient model with lesser accuracy.

1.2 Method Description

- Of-the shelf Resnet model with hit and tried mean and variance values of R, G, and B fields were used.
- Normalization were done on the data with random horizontal flips, random jitter in color and random affine transformations.

1.3 Implementation Overview

UDASequentialTrainer class

- Model: Resnet34 feature extractor from [Pyt] with normalization mean set to (0.485, 0.456, 0.406) and standard deviation set to (0.229, 0.224, 0.225).
- Data pre-processing:
  - Scaled down values if greater than 1 by dividing by 255 and transformed the data.
  - Extracted the features in a batch of 64 from the Resnet34.
- Evaluation Metric:
  - Accuracy on Evaluation datasets.

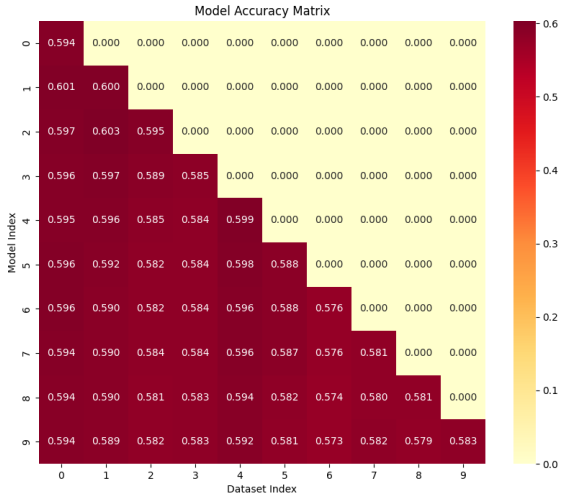

Figure 1: Emoticon

ProbabilisticLwPClassifier class

- Model: This is a probabilistic machine learning classifier using a Learned with Prototypes (LwP) approach and it supports handling uncertainty and incremental learning.
- Data pre-processing:
  - Dimensionality of input feature vectors (default 512).
  - uncertainty_threshold: Controls sensitivity to classification uncertainty.
- It does the following things incremental learning, handling uncertainty, adaptive classification, robust probability estimation.

1.4 Results

- Output: Accuracy matrix with model model_i tested on on all the datasets from 1_train_data.tar.pth to i_train_data.tar.pth with i ranging from 1 to 10.
- The lowest accuracy that was noted on any pair of model and dataset evaluation was around 58 percent.

2 PROBLEM - 1 TASK - 2

2.1 Insights

- Maximum Mean Discrepancy [Kag] seems reasonable to mitigate domain shift.

2.2 Method Description

- Filters out low-confidence predictions
- Adjusts class prototypes based on domain discrepancy
- mmd_weight allows fine-tuning domain adaptation strength

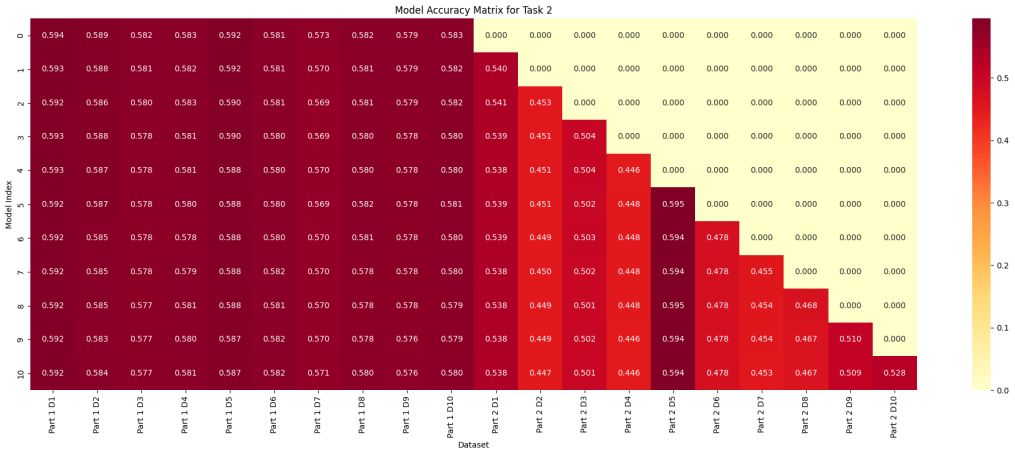

Figure 2: Emoticon

2.3 Implementation Overview

Adversarial Domain Adaption class

- Model: Calculates kernel matrices for source and target features with themselves and also in between them.
- Computation:
  - Computes the mean of these kernel matrices.
  - The loss is calculated as: xx_mean + yy_mean - 2 * xy_mean.
- Objective : Reducing domain shift in LwP model.

2.4 Results

- Output: Accuracy matrix with model model_(i+10) tested on on all the datasets from 1_train_data.tar.pth to i+10_train_data.tar.pth with i ranging from 1 to 10.
- The lowest accuracy that was noted on any pair of model and dataset evaluation was around 45 percent.

3 PROBLEM - 2

This is the video explanation is available at this channel: Youtube

Acknowledgments

Thanks to *Piyush Rai* for being in touch on *Piazza*.

References

[Kag] Kaggle. *mmd*. URL: `https://www.kaggle.com/code/onurtunali/maximum-mean-discrepancy`. (accessed: 24.11.2024).

[Pyt] Pytorch. *Resnet34*. URL: `https://pytorch.org/vision/main/models/generated/torchvision.models.resnet34.html`. (accessed: 24.11.2024).