Machine Learning Mini Project - 2

Ansh Makwe
Roll No. - 241110010

Divyansh Chaurasia
Roll No. - 241110022

Kunal Kumar Anand
Roll No. - 241110040

Parjanya Aditya Shukla
Roll No. - 241110046

1 Task 1.1

To tackle this task we took the following steps:

1. We took a pretrained ResNet18[**?**] model (trained on the CINIC-10 Dataset) to do feature extraction from the images.
2. To get the feature vectors, following steps were done:
   (a) A hook function was defined to capture the output of a specified layer (head_block). This function was registered using register_forward_hook.
   (b) The input image was resized to 224x224 pixels and normalized to match the pretrained model's input requirements.
   (c) The preprocessed image was passed through the model. During this pass, the hook captures the output of the targeted layer.
   (d) The extracted features are reshaped (to 1x512), now this works as the feature vector in our approach.
3. For each dataset we feed the images to the pipeline mentioned above and get the feature vectors.
4. Specifically, for dataset 1 we get the mean of the feature vectors for each of the 10 classes. Now, we use these means as prototypes in the following Euclidean Distance formula to predict the label of a test instance.

$$\arg \min_{k \in \{0,\ldots,9\}} \sqrt{(\mathbf{x}-\mu_k)^T(\mathbf{x}-\mu_k)}$$

   We test our initial model on the evaluation sets by assigning a test instance the label of the closest class prototype based on the above Euclidean Distance formula.
5. Now, using our model we predict the labels of the next train dataset (by using the embeddings of this). Using these labels and embeddings, we get another set of means. We combine the two sets of model parameters in a weighted sense, a factor of 0.9 is multiplied to the initial ones and 0.1 to the new ones, we did this because we had the actual labels in Dataset 1 so, the parameters obtained from it must be more important. Now, these combined parameters become our initial parameters for the next iteration. This process is used iteratively for all of the remaining datasets along with computing accuracies on the evaluation sets. The mean update formula is:

$$\mu_k = 0.9 \cdot \mu_{k-1} + 0.1 \cdot \mu_k^{\text{new}} \tag{1}$$

   Here:
   - Here, param_k represents parameters obtained after training on the k_th dataset.

- $$Param\_k^{new}$$
  represents the parameters of the classes when just the k_th training data and their respective labels according to the model of the previous iteration is considered.

1.1 Results and Observations

Dataset	Model 1	Model 2	Model 3	Model 4	Model 5	Model 6	Model 7	Model 8	Model 9	Model 10
Dataset 1	0.9620	0.9620	0.9616	0.9616	0.9620	0.9616	0.9612	0.9612	0.9608	0.9604
Dataset 2	—	0.9524	0.9512	0.9512	0.9516	0.9516	0.9516	0.9520	0.9516	0.9516
Dataset 3	—	—	0.9536	0.9536	0.9536	0.9536	0.9532	0.9536	0.9528	0.9524
Dataset 4	—	—	—	0.9500	0.9500	0.9496	0.9492	0.9492	0.9488	0.9488
Dataset 5	—	—	—	—	0.9528	0.9524	0.9528	0.9528	0.9528	0.9524
Dataset 6	—	—	—	—	—	0.9500	0.9496	0.9496	0.9488	0.9488
Dataset 7	—	—	—	—	—	—	0.9500	0.9496	0.9496	0.9492
Dataset 8	—	—	—	—	—	—	—	0.9512	0.9516	0.9520
Dataset 9	—	—	—	—	—	—	—	—	0.9508	0.9508
Dataset 10	—	—	—	—	—	—	—	—	—	0.9572

Table 1: Accuracies of Models across Datasets

The accuracy of all models - Model 1 to Model 10 were quite similar on different datasets. The different models have good generalized performance with accuracies in the range of 94% to 96%.

2 Task 1.2

The approach for the Task 1.2 is similar as the Task 1.1, but with some changes, here we introduce basic preprocessing of our images. Note that this preprocessing is applied before, extracting the features. The preprocessing was applied only on some of the datasets from the second part datasets, after seeing on which datasets the final model of Task 1.1 is giving bad accuracies.

2.1 Reason for Pre-processing

The reason for pre-processing the datasets like this is that the images in the dataset part 2 are slight variations with some being blurred, some having additive noises of different types, some having mosaics, or jitters, etc. So, to negate these degradations applied to images we applied these processing steps. Here is a brief description of the pre-processing steps applied.

Given datasets from different distributions, preprocessing aims to align their distributions to a shared representation, ensuring better consistency for downstream evaluation tasks.

2.2 Part 2 Dataset 1

1. **Enhance details using unsharp masking for sharpening.**
   Sharpening enhances high-frequency components in the image by modifying pixel intensities as:
   $$I_{\text{sharpened}} = I_{\text{original}} + \alpha(I_{\text{original}} - I_{\text{blurred}}), \tag{2}$$
   where α controls the enhancement strength. By redistributing pixel values, this emphasizes edges and fine details, aligning the image intensity distribution $P(I)$ closer to that of sharper datasets.

2.3 Part 2 Dataset 2, 8, 9

1. **Gaussian smoothing and bilateral filtering to reduce mosaic noise.**
   Noise introduces randomness in intensity distributions, increasing variance. Smoothing transforms the image distribution $P(I)$ by suppressing high-frequency noise:
   $$I_{\text{smoothed}}(x, y) = \sum_{i,j} I(x + i, y + j) \cdot G(i, j, \sigma), \tag{3}$$
   where $G(i, j, \sigma)$ is a Gaussian kernel. Bilateral filtering further refines this by preserving edge regions while reducing noise.

2. **Enhance details using unsharp masking.**

   Post-smoothing, unsharp masking mildly sharpens the image, restoring some high-frequency details while maintaining noise reduction.

3. **Adjust image contrast using `ImageEnhance.Contrast`.**

   Contrast adjustment maps the pixel intensity distribution to span a wider range:

$$I_{\text{contrast-adjusted}} = \text{ContrastFactor} \cdot (I_{\text{original}} - \mu) + \mu, \tag{4}$$

   where μ is the mean intensity. This increases the dynamic range, aligning intensity distributions $P(I)$ across datasets.

2.4 Other Reamining Datasets of Part-2

They were used as it is without any preprocessing.

2.5 Results and Observations

Dataset	Model 1	Model 2	Model 3	Model 4	Model 5	Model 6	Model 7	Model 8	Model 9	Model 10
D1-S1	0.9564	0.9564	0.9576	0.9576	0.9568	0.9572	0.9572	0.9576	0.9572	0.9556
D2-S1	0.9492	0.9492	0.9496	0.9492	0.9476	0.9488	0.9488	0.9480	0.9480	0.9480
D3-S1	0.9508	0.9512	0.9516	0.9508	0.9504	0.9500	0.9500	0.9508	0.9492	0.9492
D4-S1	0.9464	0.9480	0.9472	0.9468	0.9468	0.9472	0.9464	0.9472	0.9464	0.9472
D5-S1	0.9548	0.9536	0.9524	0.9524	0.9524	0.9520	0.9524	0.9516	0.9512	0.9512
D6-S1	0.9460	0.9460	0.9452	0.9456	0.9448	0.9448	0.9448	0.9460	0.9456	0.9448
D7-S1	0.9468	0.9468	0.9464	0.9468	0.9468	0.9472	0.9472	0.9464	0.9456	0.9456
D8-S1	0.9496	0.9484	0.9488	0.9484	0.9484	0.9484	0.9476	0.9484	0.9480	0.9480
D9-S1	0.9476	0.9472	0.9464	0.9460	0.9460	0.9448	0.9444	0.9448	0.9444	0.9440
D10-S1	0.9536	0.9540	0.9536	0.9536	0.9532	0.9544	0.9536	0.9528	0.9524	0.9524
D1-S2	0.8268	0.8292	0.8264	0.8260	0.8244	0.8236	0.8244	0.8232	0.8244	0.8252
D2-S2	—	0.6384	0.6360	0.6340	0.6316	0.6268	0.6264	0.6276	0.6296	0.6300
D3-S2	—	—	0.8036	0.8032	0.8016	0.8024	0.8016	0.8016	0.8004	0.8004
D4-S2	—	—	—	0.8756	0.8752	0.8756	0.8752	0.8760	0.8760	0.8756
D5-S2	—	—	—	—	0.9168	0.9172	0.9168	0.9164	0.9168	0.9168
D6-S2	—	—	—	—	—	0.7652	0.7660	0.7656	0.7640	0.7644
D7-S2	—	—	—	—	—	—	0.8860	0.8852	0.8852	0.8856
D8-S2	—	—	—	—	—	—	—	0.7428	0.7428	0.7412
D9-S2	—	—	—	—	—	—	—	—	0.7688	0.7684
D10-S2	—	—	—	—	—	—	—	—	—	0.8668

Table 2: Accuracies of Models across Datasets (Set 1 and Set 2)

The models performed extremely well on the datasets 1-10 from Set 1 and achieved decent accuracy of 63% - 91% on datasets 1-10 from Set 2. The accuracy of each model on some tough datasets like D2 from set 2 is around 63% as its distribution may be a lot more different than the rest. The accuracy of each model on some similar dataset like D5 from set 2 is around 91% as its distribution may have high similarity with the ones in Set 1.

3 NOTE

We have uploaded the feature vectors and other required files in our drive, they can be accessed from the following link- Drive Link

These can be downloaded and put in the folder of the code to reproduce the results (paths in the code should verified, more on this in the readme).

4 Task 2

The link of the YouTube video for the task 2 is - YouTube Link

5 References

1. **He, K., Zhang, X., Ren, S., & Sun, J.** (2015). *Deep Residual Learning for Image Recognition*. arXiv:1512.03385.
2. **PyTorch for Deep Learning & Machine Learning – Full Course** Learn Pytorch.
3. **Feature Extraction Methods for the classification of images** Learn Feature Extraction.
4. **Learning With Prototypes** LwP Model.