CMSC828C

Statistical Pattern Recognition

Project 2

Digit Recognition and Monkey Species Classification

ADITYA VARADARAJ

(UID: 117054859)

(SECTION: 0101)

M.Eng. Robotics (PMRO),

University of Maryland - College Park, College Park, MD - 20742, USA.

Date of Submission: 14th December, 2022

List of Figures

List of Tables

CONTENTS

I. Hand-Written Digit Recognition

A. Dataset

Dataset used was MNIST dataset which can be loaded from torchvision library in Python or downloaded from official site of MNIST.

B. Preprocessing

We perform PCA or MDA before feeding data into SVM and Logistic Regression classifiers.

1) PCA: Principal Component Analysis (PCA) projects data into a lower dimension to represent major aspects of data with minimum no. of features (Direction of Maximum Variance). We use the eigenvector found for projecting the training data to project the testing data as well.

2) MDA: MDA projects data into a lower dimension to get more seperable data (Direction of maximum discriminability). We use the eigenvector found for projecting the training data to project the testing data as well

C. Kernel SVM

Uses Kernels and RKHS to perform SVM linear classification in higher dimension. Uses the idea that nonlinear classification in lower dimension can be represented as a linear classification in higher dimension.

1) Linear: Linear Kernel is given by $K(x, y) = x^T y$

2) Polynomial: Polynomial Kernel is given by $K(x, y) = (\gamma x^T y + r)^d$, where, $\gamma,\ r$ and degree d are hyperparameters

3) RBF: RBF Kernel is given by $\left(K(x, y) = e^{-\gamma||x-y||^2}\right)$ with γ as hyperparameter.

D. Logistic Regression

Here, the class posteriors are given as:

$$P(\omega_m|x_n) \ = \ \phi_{nm} \ = \ \frac{\exp \theta_m^T x_n}{\sum_{j=1}^M \exp \theta_j^T x_n} \tag{1}$$

We need to encode y such that y is an $N \times M$ matrix with $y_{nm} = 1$ if label of n^{th} sample is m. We try to minimize the cross-entrop loss function

$$\begin{gathered} L(\theta_1, ..., \theta_m) = \ -\sum_{n=1}^N \sum_{m=1}^M y_{nm} \log \phi_{nm} \\ \nabla_{\theta_j} L = \sum_{n=1}^N (\phi_{nj} - y_{nj}) x_n \end{gathered} \tag{2}$$

Then we use gradient descent update

$$\theta_j^{(i)} = \theta_j^{(i-1)} - \alpha * \nabla_{\theta_j^{(i-1)}} L, \ j = 1, ..., M \tag{3}$$

, where, α is learning rate

E. Deep Learning

The **LeNet-5** model [1] was implemented in PyTorch. Loss used was Cross-Entropy loss and optimizer used was Adam.

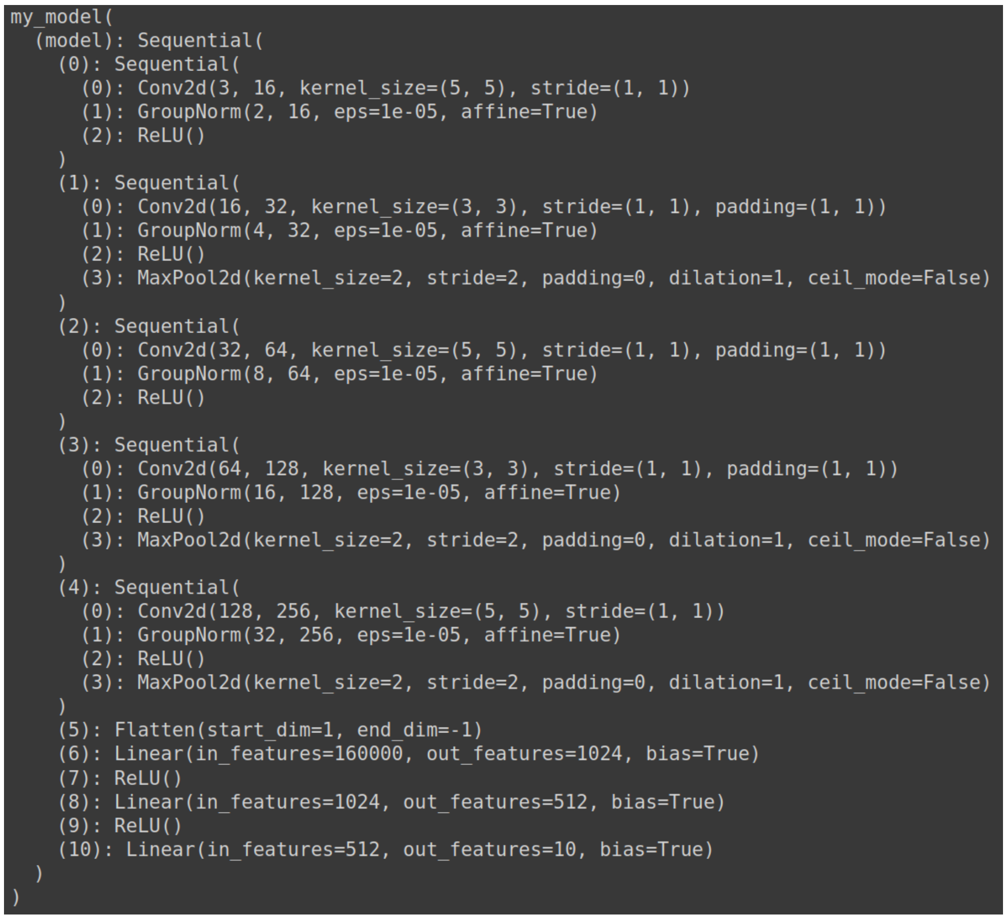
II. Transfer Learning

A. Dataset

The dataset used was the **Kaggle Monkey Species dataset**

B. Simple 3-5 Convolutional Network

A network having 5 Convolutional layers was trained on Monkey Species dataset and training and validation accuracies were checked. The model architecture is shown in Fig. 1. Each of the 5 convolutional layers were followed by a Spatial Group Normalization and ReLU activation. 2^{nd}, 4^{th} and 5^{th} layers were followed by a Max Pooling layer.



```
my_model(
  (model): Sequential(
    (0): Sequential(
      (0): Conv2d(3, 16, kernel_size=(5, 5), stride=(1, 1))
      (1): GroupNorm(2, 16, eps=1e-05, affine=True)
      (2): ReLU()
    )
    (1): Sequential(
      (0): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): GroupNorm(4, 32, eps=1e-05, affine=True)
      (2): ReLU()
      (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    )
    (2): Sequential(
      (0): Conv2d(32, 64, kernel_size=(5, 5), stride=(1, 1), padding=(1, 1))
      (1): GroupNorm(8, 64, eps=1e-05, affine=True)
      (2): ReLU()
    )
    (3): Sequential(
      (0): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): GroupNorm(16, 128, eps=1e-05, affine=True)
      (2): ReLU()
      (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    )
    (4): Sequential(
      (0): Conv2d(128, 256, kernel_size=(5, 5), stride=(1, 1))
      (1): GroupNorm(32, 256, eps=1e-05, affine=True)
      (2): ReLU()
      (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    )
    (5): Flatten(start_dim=1, end_dim=-1)
    (6): Linear(in_features=160000, out_features=1024, bias=True)
    (7): ReLU()
    (8): Linear(in_features=1024, out_features=512, bias=True)
    (9): ReLU()
    (10): Linear(in_features=512, out_features=10, bias=True)
  )
)
```

Fig. 1. 5-layer CNN Model Architecture

C. Pre-Trained Model as Feature Extractor

A pre-trained **ResNet-34** model (Fig. 2) was loaded. The convolutional layers were all frozen and the fully connected layers were replaced by untrained fully connected classifier layers and were updated by training on the data.

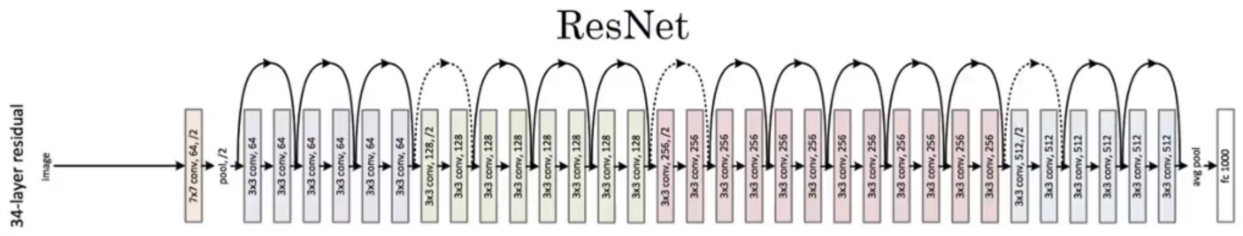


Fig. 2. ResNet-34 Model Architecture

D. *Fine-Tuning*

We unfreezed the convolutional layers of a **ResNet-50** model (Fig. 3)to fine-tune the network and trained the weights for 2 more epochs.

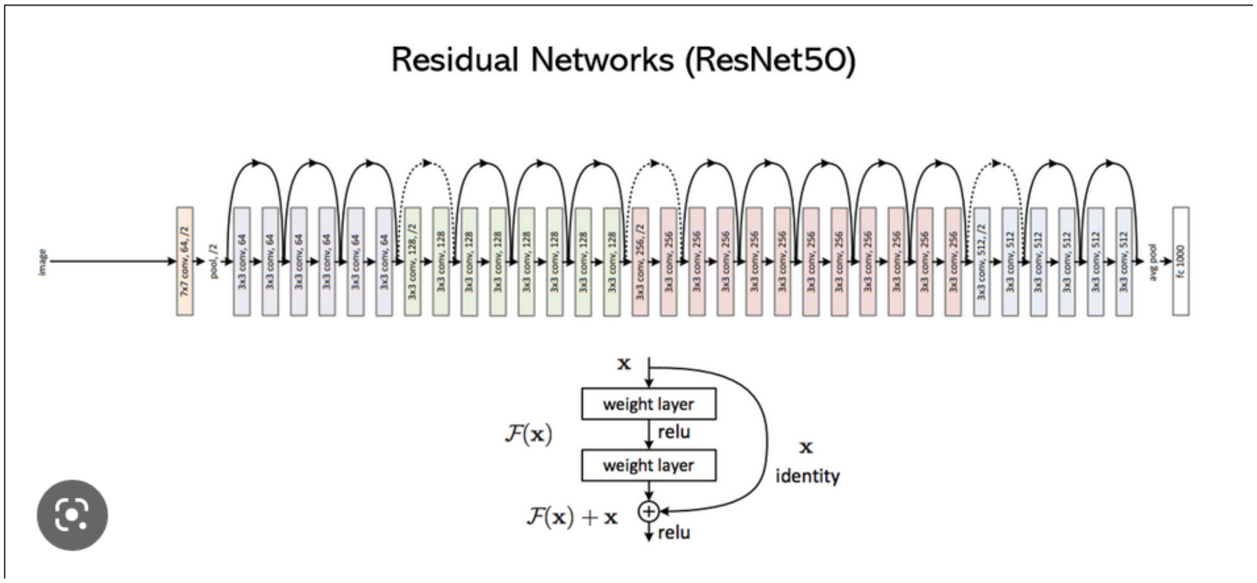


Fig. 3. ResNet-50 Model Architecture

III. Results

The results observed for **Handwritten Digit Recognition** are summarized in Table I.

TABLE I
Results for Task 1: Handwritten Digit Recognition

Classifier	Dimensionality Reduction	Kernel	Best hyperparameter values	Best Training Accuracy (%)	Best Test Accuracy (%)
SVM	PCA	Linear			66.22
SVM	PCA	Polynomial	$d = 3$		71.28
SVM	PCA	RBF	$\gamma = 1000$		11.35
SVM	MDA	Linear			11.39
SVM	MDA	Polynomial	$d = 3$		11.35
SVM	MDA	RBF	$\gamma = 1000$		87.83
Logistic Regression	PCA		$\alpha =$ 1e-8	89.78	75.4
Logistic Regression	MDA		$\alpha =$ 1e-8	78.84	78.95
LeNet-5	None		$lr =$1e-3, 15 epochs	Validation accuracy: 98.47	Test Accuracy: 98.43

A. Kernel SVM

1) Linear: From the table I, we see that it gives 66.22% accuracy for SVM with PCA, while, it gives 11.39% accuracy for SVM with MDA

2) Polynomial: From the table I, we see that it gives 71.28% accuracy for SVM with PCA, while, it gives 11.35% accuracy for SVM with MDA. We tried various degrees of polynomial and found 3rd degree polynomial to work best as shown in Fig. 4

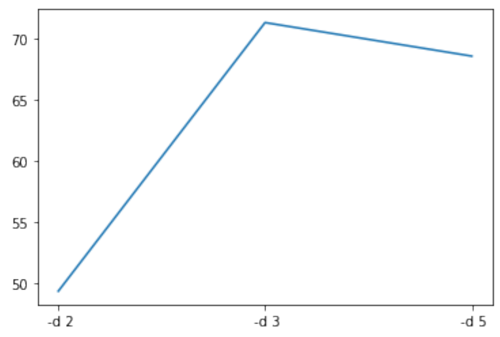


Fig. 4. Testing accuracy of Polynomial Kernel SVM with PCA for different degrees

3) RBF: From the table I, we see that it gives 11.35% accuracy for SVM with PCA, while, it gives 87.83% accuracy for SVM with MDA We tried various values of γ and found $\gamma = 1000$ to work best.

B. Logistic Regression

From the table I, we see that, with learning rate of 1e-8, it gives 89.78% training accuracy and 75.4% testing accuracy for Logistic Regression with PCA, while, it it gives 78.84% training accuracy and 78.95% testing accuracy for Logistic Regression with MDA. The training losses were plotted as shown in Fig. 5 and Fig. 6.

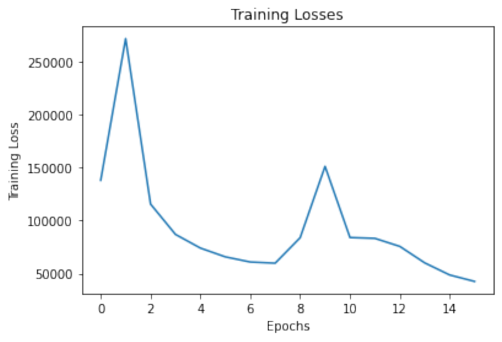


Fig. 5. Training Losses of Logistic Regression with PCA over 15 epochs

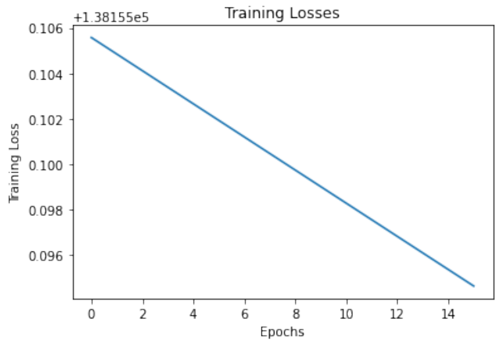


Fig. 6. Training Losses of Logistic Regression with MDA over 15 epochs

C. *Deep Learning*

From table I, we observe that we get Validation accuracy of 98.47% and Test accuracy of 98.43% by training and testing on MNIST dataset directly for learning rate of 1e-3 over 15 epochs. The plot for Training Losses and Validation accuracies over 15 epochs is shown in Fig. 7

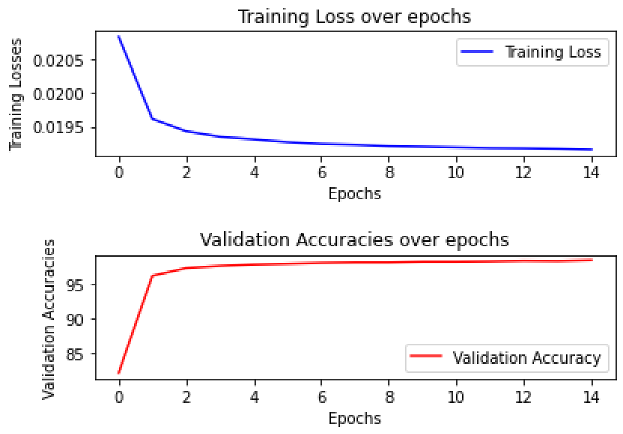


Fig. 7. Training Losses and Validation Accuracies of LeNet-5 over 15 epochs

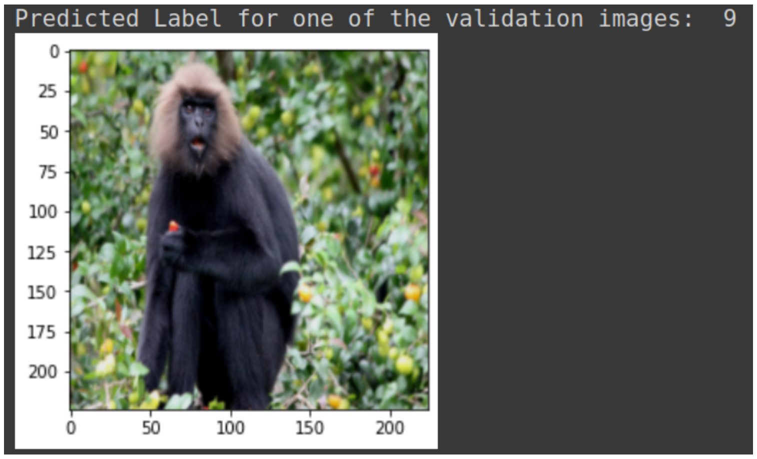
D. Transfer Learning

We get the results shown in Table II with learning rate 0.001, momentum 0.9, SGD + Momentum, Cross-Entropy Loss, Exponential LR decay.

TABLE II
RESULTS FOR TASK 2: MONKEY SPECIES CLASSIFICATION

Classifier	No. of Epochs	Training Loss	Validation Accuracy (%)
5-layer CNN	5	1.3691	33.46
Pretrained ResNet-34	5	0.0969	98.16
Fine-Tuned ResNet-50	2	0.1323	99.63

One of the predictions for validation set is shown in Fig. 8 which can be checked by checking for the image in folder n9 of validation set.



```
Predicted Label for one of the validation images:  9
```

Fig. 8. Transfer Learning Final Prediction for a validation sample

IV. CONCLUSION

From the observed results, we can conclude that:

- Degree 3 Polynomial Kernel works best for SVM on MNIST data projected using PCA
- RBF Kernel with $\gamma = 1000$ works best for SVM on MNIST data projected using MDA.
- Logistic regression gives better training accuracy with PCA but better testing accuracy with MDA.
- LeNet-5 is able to correctly classify MNIST data in 5 epochs with learning rate of 1e-3 with very good validation and test accuracies.
- Since data is less in the Monkey dataset, the 5-layer network gives only 33.46% validation accuracy on monkey dataset which is very small.
- Using transfer learning by using pre-trained ResNet-34 as a feature extractor increases the validation accuracy to 98.16% which is pretty good.
- Further fine-tuning using ResNet-50 gives extremely good 99.63% accuracy.

References

[1] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, ”Gradient-based learning applied to document recognition”, Proceedings of the IEEE, Nov 1998.