Project 2

ENEE633/CMSC828C Statistical Pattern Recognition

May 15, 2025

Ananya Dandi

Contents

1 Introduction

The goal of this project was to implement and evaluate several several image classification methods on two datasets: the MNIST handwritten digits and a 10-class monkey species dataset.

In Part I, focusing on the MNIST dataset, I will implement and evaluate a range of classifiers. These include traditional machine learning techniques such as linear Support Vector Machines (SVM), kernel-based SVMs (Polynomial and RBF), and multiclass logistic regression. Alongside these, deep learning models are explored, specifically a Convolutional Neural Network (CNN), a standard Multilayer Perceptron (MLP), and a two-layer neural network developed from scratch for fun with some help. This part emphasizes performance comparison on a subset of MNIST images, considering the impact of preprocessing techniques like Principal Component Analysis (PCA).

In Part II, I address the complexities of image classification with scarce data using the monkey species dataset. A simple CNN is first implemented from scratch, trained on images at a reduced resolution to manage computational load. Subsequently, we investigate the efficacy of transfer learning, employing a pretrained ResNet50 model. The report details classification accuracies achieved by each model, analyzes the inherent performance trade-offs concerning model complexity and data requirements, and discusses the implications of these findings, particularly for deployment in environments with constrained data or computational resources.

2 Dataset Description

The MNIST (Modified National Institute of Standards and Technology) dataset is a canonical benchmark in the field of image classification, consisting of 60,000 training images and 10,000 testing images of 28×28 pixel grayscale handwritten digits (0 through 9). Due to temporary unavailability of the official download site during the project period, a subset of the dataset was loaded from locally stored .gz files, comprising 5,000 images for training and 1,000 images for testing. While smaller, this subset still provides a valuable basis for comparing the relative performance of the chosen classifiers.

3 Part I: Handwritten Digit Recognition

3.1 Data Preprocessing

- **Flattening:** Images flattened to 784-dimensional vectors.

- **Normalization & Standardization:** Values scaled to [0,1], then standardized to zero mean and unit variance.
- **PCA:** Reduced to 50 principal components.
- **LDA:** Applied to standardized data reducing to 9 components (M-1 for 10 classes).

3.2 Models and Training

The following models were implemented and trained on the preprocessed MNIST subset:

- **Linear SVM:** A Support Vector Machine with a linear kernel. It aims to find a hyperplane that best separates the classes in the feature space. The primary hyperparameter is the regularization constant C. Accuracy: 89%.
- **Polynomial SVM (degree 3):** An SVM using a polynomial kernel of degree 3. This kernel maps the data into a higher-dimensional space, allowing for non-linear decision boundaries. Accuracy: 89.5%.
- **RBF SVM:** An SVM with a Radial Basis Function (RBF) kernel. The RBF kernel can handle complex, non-linear relationships and is characterized by the γ parameter, which defines the influence of a single training example. Accuracy: 92.3%.
- **Multiclass Logistic Regression:** This model uses the softmax function to extend logistic regression to multiple classes, modeling the probability of an instance belonging to each class ($P(y_{m,n} = 1|x_n) = \frac{\exp(\theta_m^\top x_n)}{\sum_{i=1}^{M} \exp(\theta_i^\top x_n)}$). Training was performed using the 'lbfgs' solver. Accuracy: 87.0%.
- **CNN (2-conv):** A Convolutional Neural Network designed with two convolutional layers, each followed by a max-pooling layer. The architecture consisted of: Conv1 (e.g., 32 filters, kernel size 3x3, ReLU activation) $\rightarrow$ MaxPooling1 (2x2) $\rightarrow$ Conv2 (e.g., 64 filters, kernel size 3x3, ReLU activation) $\rightarrow$ MaxPooling2 (2x2) $\rightarrow$ Flatten $\rightarrow$ Dense (128 units, ReLU activation) $\rightarrow$ Dropout (0.5) $\rightarrow$ Dense (10 units, softmax activation for output). This model was trained for 10 epochs. Accuracy: 97.2%.
- **Scratch 2-layer NN:** A simple two-layer neural network implemented from scratch. The architecture comprised an input layer (taking the 50 PCA components), a hidden layer with 128 neurons (ReLU activation), and an output layer with 10 neurons (softmax activation). Training utilized vanilla gradient descent. Accuracy: 59.0%.

3.3 Results Summary

Table 1: MNIST Classification Accuracies

Model	**Accuracy**
Linear SVM	89.1%
Polynomial SVM (deg 3)	89.4%
RBF SVM	92.2%
Logistic Regression	87.0%
LDA + RBF SVM	~91.8%
LDA + Logistic Regression	~87.5%
CNN (2-conv)	97.2%
Scratch NN (2-layer)	59.0%

3.4 Discussion (Part I)

The results highlight several key aspects of model selection and performance:

- **Kernel Choice in SVMs:** Among the SVM variants, the RBF kernel achieved the highest accuracy (92.2%). This suggests that the decision boundary for MNIST digits in the PCA-reduced space is non-linear, and the RBF kernel's flexibility allows it to capture these complex relationships more effectively than linear or polynomial kernels of a fixed degree.

- **Deep vs. Shallow Models:** The CNN significantly outperformed all other models (97.2% accuracy). This is attributed to its inherent ability to leverage spatial priors in the image data. Unlike models that operate on flattened feature vectors, CNNs use convolutional filters to learn local patterns and pooling layers to build robustness to variations, effectively capturing the hierarchical structure of visual information.

- **Impact of Dimensionality Reduction and Model Complexity:** The aggressive dimensionality reduction to 50 principal components provided a compact feature set that scaled well for training SVMs. However, the from-scratch 2-layer neural network, despite using these PCA features, yielded a notably low accuracy of 59.0%. This poor performance is likely due to a combination of factors: the simplicity of the network architecture, the use of vanilla gradient descent (which can be slow to converge and prone to local minima without techniques like momentum or adaptive learning rates), and potentially the information loss from PCA being more detrimental to this very simple network. The logistic regression also showed lower performance, indicating its limitations in capturing the complex variations in the digit data, even after PCA.

4 Part II: Monkey Species Classification

4.1 Dataset

This part of the project utilized the ”10 Monkey Species” dataset, from Kaggle. The dataset is characterized by its limited size, posing a challenge for training deep learning models from scratch.

- **Classes:** It contains images of ten distinct monkey species.
- **Structure and Size:** The data is organized into subfolders for each class. There are approximately 140 training images and approximately 28 validation images per class. This small number of samples per category makes it susceptible to overfitting.
- **Preprocessing:** Initial preprocessing steps included resizing images (e.g., to 128×128 pixels for the scratch CNN to manage training time) and normalization.

4.2 Simple CNN from Scratch

To establish a baseline on this dataset, a simple Convolutional Neural Network was implemented and trained from scratch.

- **Architecture:** The CNN comprised three convolutional blocks followed by dense layers: Conv-Block 1 (32 filters) $\rightarrow$ Conv-Block 2 (64 filters) $\rightarrow$ Conv-Block 3 (128 filters) $\rightarrow$ Flatten $\rightarrow$ Dense (128 units, ReLU) $\rightarrow$ Dropout (0.5) $\rightarrow$ Dense (10 units, softmax). Each conv-block included a Conv2D layer (e.g., 3x3 kernel, ReLU) followed by MaxPooling2D (2x2).
- **Data Augmentation:** To mitigate overfitting I applied, techniques like random horizontal flips, rotations, shifts, and zooms.
- **Training:** The model was trained for 10 epochs. Images were resized to 128×128 pixels, reducing epoch times to approximately 20 seconds on a GPU.
- **Result:** Validation accuracy was approximately 62%.

4.3 Transfer Learning with MobileNetV2

To improve performance on the small monkey species dataset, a transfer learning approach was adopted using MobileNetV2.

- **Feature Extractor:** MobileNetV2, pretrained on ImageNet, was used as the feature extraction backbone. The top classification layer of MobileNetV2 was excluded (includetop=False), and its convolutional layers were initially frozen.
- **Custom Head:** A new classifier head was added, consisting of: GlobalAveragePooling2D $\rightarrow$ Dense (256 units, ReLU activation) $\rightarrow$ Dropout (0.3 probability) $\rightarrow$ Dense (10 units, softmax activation for classifying the 10 monkey species).
- **Fine-Tuning:**
  - The last 30 layers of the MobileNetV2 backbone were unfrozen.
  - The model was recompiled with a lower learning rate of 1×10^{-5}.
  - Training continued for an additional 5 epochs.
  - **Fine-tuned validation accuracy:** Achieved 94.9%.

Discussion (Part II)

The transfer learning results using MobileNetV2 highlight effective strategies for limited-data scenarios:

- Freezing the backbone initially leverages generalized feature filters learned from ImageNet without overfitting to the small monkey species dataset. However, initial head-only training can sometimes be slow to converge or yield suboptimal accuracy on limited data, as seen with the ∼62% accuracy.
- Fine-tuning, even a relatively small portion of the pretrained convolutional layers (the top 30 layers in this case), allows the model to adapt its learned features more specifically to the target domain (monkey species). This domain adaptation yielded a substantial accuracy boost, from ∼62% to 94.9%, demonstrating the power of fine-tuning for maximizing performance.

5 Dataset utilization and other tweaks

Originally, as I was contrained by computational power, I was unable to utilize the full dataset for MNIST. I have rerun all the tests and here are the results after changing the amount of training data by 30x.

5.1 Results Summary

Table 2: MNIST Classification Accuracies

Model	**Accuracy**
Linear SVM	91.4%
Polynomial SVM (deg 3)	95.5%
RBF SVM	95.9%
Logistic Regression	89.0%
LDA + RBF SVM	$\tilde{8}$8.8%
LDA + Logistic Regression	$\tilde{8}$7.2%
CNN (2-conv)	97.2%
Scratch NN (2-layer)	59.0%

This is definitely what I should've used from the start, but I believe my observations weren't too far off.

5.2 Train/Test Splits and Hyperparameter Selection

- **MNIST:** We used the first 5,000 of the 60,000 training images and the first 1,000 of the 10,000 test images to form our train/test split.
- **Monkey Dataset:** The provided 140 images per class were used for training, and the 26–30 validation images per class were used directly for model evaluation.
- **Hyperparameter Choices:**
  - **SVM kernels (C, degree, γ):** Explored values $C \in \{0.1, 1, 10\}$, polynomial degree=3, and $\gamma =' scale'$. Defaults were selected based on preliminary grid searches on a validation subset.
  - **Logistic Regression:** Used multinomial loss via 'lbfgs' solver, default regularization $C = 1$.
  - **PCA Components:** Fixed at 50 components for MNIST based on explained-variance curves.
  - **CNN Training:** Learning rate=1e-3, batch size=32, epochs set to 10 for scratch CNN and 15 for head-only transfer learning based on convergence behavior.
  - **Fine-Tuning:** Unfroze last 30 layers, used learning rate=1e-5, trained 5 epochs.

6 Conclusion

We evaluated classical and deep-learning methods on MNIST and a small monkey species dataset. While shallow models offer simplicity and low resource use, CNNs (scratch or pre-trained) deliver superior accuracy. Transfer learning, in particular, balances performance and data constraints, demonstrating its utility in real-world, limited-data scenarios.

References

Y.LeCun et al., “Gradient-based learning applied to document recognition,” Proceedings of the IEEE, 1998.