Individual project

220251110

Abstract

This report explores the application of various machine learning techniques to classify handwritten digits from the MNIST dataset. A comprehensive investigation into models such as Logistic Regression, Linear Discriminant Analysis (LDA), Decision Tree, Random Forest, and Neural Network reveals insights into their performance in terms of accuracy, misclassification, and learning behaviors. The Neural Network emerged as the best-performing model, achieving 99.17% accuracy on the test. The study discusses the strengths and limitations of the classifiers and offers recommendations for model selection in image classification tasks. Furthermore, the study explores optimizing a neural network to enhance performance, addressing strategies to improve accuracy while balancing efficiency and scalability.

I. Introduction

Handwritten digit classification is a cornerstone problem in machine learning and computer vision, frequently used to benchmark algorithms' performance [1]. The MNIST dataset, comprising of 70,000 labeled grayscale images of digits (0–9), provides a standard platform for such tasks [2]. Each image is 28×28 pixels, making the dataset computationally efficient and representative of real-world challenges in optical character recognition (OCR).

This report investigates five machine learning algorithms for digit classification:

- Logistic Regression (LR) [3].
- Linear Discriminant Analysis (LDA) [4].
- Decision Tree (DT) [5].
- Random Forest (RF) [6].
- Neural Network (NN) [1].

The study compares these models based on accuracy, computational efficiency, and error analysis, offering insights into their applicability for similar tasks.

II. Methodology

A. Reasoning Behind Preprocessing

The preprocessing phase is crucial for ensuring that the models receive data that allows them to learn effectively. For the MNIST dataset, the `fetch_openml` function from `scikit-learn` [7] was used to access the data. The dataset was split into a training set (60,000 samples) and a test set (10,000 samples). Min-Max Scaling was chosen to normalize pixel intensities to a [0,1] range. This scaling helps ensure that gradient-based algorithms, such as neural networks, train efficiently by preventing numerical instabilities and vanishing gradients [8].

B. Selection of Classification Algorithms

Our selection of algorithms aims to balance linear and non-linear models, considering both theoretical and practical factors. We will develop baseline models for each of the following algorithms:

- **Logistic Regression:** Chosen for its simplicity and efficiency as a linear model, providing a baseline for comparison.
- **Linear Discriminant Analysis:** Selected for its statistical approach, assuming Gaussian distributions, to test these assumptions against actual data properties.
- **Decision Tree:** Offers an interpretable model that captures non-linear patterns, helpful for understanding feature importance.
- **Random Forest:** Anticipated to provide robust predictions by combining multiple decision trees, reducing variance and enhancing accuracy.
- **Neural Network:** Selected for its capacity to model complex, non-linear relationships through its multi-layered structure.

We note that for the selection of algorithms above we used the default models from `scikit-learn` library. More specifically, for the Random Forest algorithm we used 100 trees and for the Neural Network we used 1 hidden layer with 100 neurons.

These models are candidate models for further training. After training and evaluating these baseline models, we will choose the best one (based on validation accuracy) and optimize it to get performance as close as possible to the best theoretical classifier.

C. Approach to Model Evaluation

For each of the five models, we conducted the following steps:

- Cross-validation: The models were evaluated using 5-fold cross-validation, meaning the training data was split into 5 subsets, and each model was trained and tested 5 times, each time using a different subset as the validation set while the remaining four subsets were used for training.
- Model Evaluation: The mean accuracy of each model across all 5 folds was computed to determine how well each model performs on unseen data.
- Conclusion for Each Model: We provide the validation accuracy for each model (see Table I). The neural network emerged as the best candidate model.

TABLE I: Validation Accuracy of Models

Model	**Validation Accuracy (%)**
Logistic Regression	92.0
Linear Discriminant Analysis	86.3
Decision Tree	86.8
Random Forest	96.7
Neural Network	97.5

D. Discussion

Each model demonstrated unique strengths and weaknesses when classifying the MNIST dataset. **Logistic Regression** succeeded because it is a simple linear model that performs well on datasets like MNIST, where the classes are somewhat linearly separable. Its straightforward nature allowed it to achieve high accuracy with minimal computation. **Linear Discriminant Analysis** (LDA) also performed well but was less effective than Logistic Regression. LDA assumes a Gaussian distribution of the features, which is a limiting factor when the assumptions don't perfectly hold for the data, leading to slightly reduced accuracy compared to simpler models. **Decision Tree** showed variability in performance due to overfitting. While it could capture non-linear relationships in the data, it was sensitive to small changes in the training data, leading to some folds performing poorly. **Random Forest**, an ensemble of Decision Trees, performed consistently well. By averaging predictions from multiple trees, it mitigated overfitting, resulting in strong and stable performance. **Neural Network**, while powerful in capturing complex patterns, showed more variability across folds. This was likely due to the model's sensitivity to hyperparameters, such as the number of iterations and network architecture. Despite this, the Neural Network was still able to achieve competitive performance, highlighting its potential for handling more complex patterns (more specifically in the MNIST case non-linear patterns). Due to the power of Neural Networks, as well as its validation performance we choose this and we optimize it further to achieve better results in the section below. We evaluated this model on the **test set** and got 97.5% accuracy, which is the score we are trying to beat.

E. Key Implementation Decisions

The neural network architecture demonstrated very high accuracy (97.5%). However, its performance on this task, while impressive, still leaves room for improvement, as it is known that the theoretical limit of the MNIST dataset is closer to 100%. In addition, the sensitivity of the neural network to the architecture and the hyperparameters pose challenges, making it an ideal candidate for further optimization.

By improving upon the neural network architecture, we aim to get a score closer to the best achievable. We outline the key implementation decisions below:

- **Data Augmentation**: Choose some images and randomly rotate by 10 degrees, zoom by 10% and shift in both directions by 10%. This should increase the robustness of the model as we now feed it some irregular digits to train.
- **Variable Learning Rate**: Used a learning rate scheduler to dynamically adjust the learning rate by a factor of 0.2 when validation loss stagnates. This should help the algorithm converge faster and reach closer to the global minima.
- **Modify LeNet-5**: LeNet-5 is a well-known convolutional neural network which already performs good on the MNIST dataset [1]. It consisits of two convolutional layers followed by pooling layers, and three fully connected layers for classification. We introduce the following modifications to enhance its performance.

More specifically, the number of filters in the convolutional layers was increased (e.g., 32 and 64 filters in the first and second convolutional layers) to capture more complex features. We replaced the sigmoid/tanh activations with ReLU, enabling faster training and improved gradient flow. Batch normalization was added after each convolutional layer to stabilize learning and accelerate convergence, while dropout layers with a rate of 25% were included after the pooling and fully connected layers to reduce overfitting (since we are using a larger network this time). The fully connected layers were expanded to 256 and 128 units to improve the network's capacity for learning complex patterns. We present the performance of the model in the next section.

III. Results and Analysis

A. Discussion

After training for 100 epochs with the `PyTorch` library, the model achieved a perfect 100% accuracy on the training set. However, the test set accuracy reached **99.17%**, suggesting that while the model performed exceptionally well on the training data, some degree of overfitting occurred, as indicated by the slight drop in performance on unseen data. One possible way to overcome this would be to add more regularization.

B. Visual Insights: Confusion Matrix and Example Errors

Figures 1 and 2 provide detailed visual insights into the model's errors. The misclassifications highlight the challenge of similar digit pairings in MNIST, such as 3 and 5, suggesting opportunities for improvement through refined feature engineering or advanced model tuning. Furthermore, Figure 2 reveals that some misclassified images are inherently ambiguous, indicating that achieving 100% accuracy on the test set may not be feasible, as mentioned before.

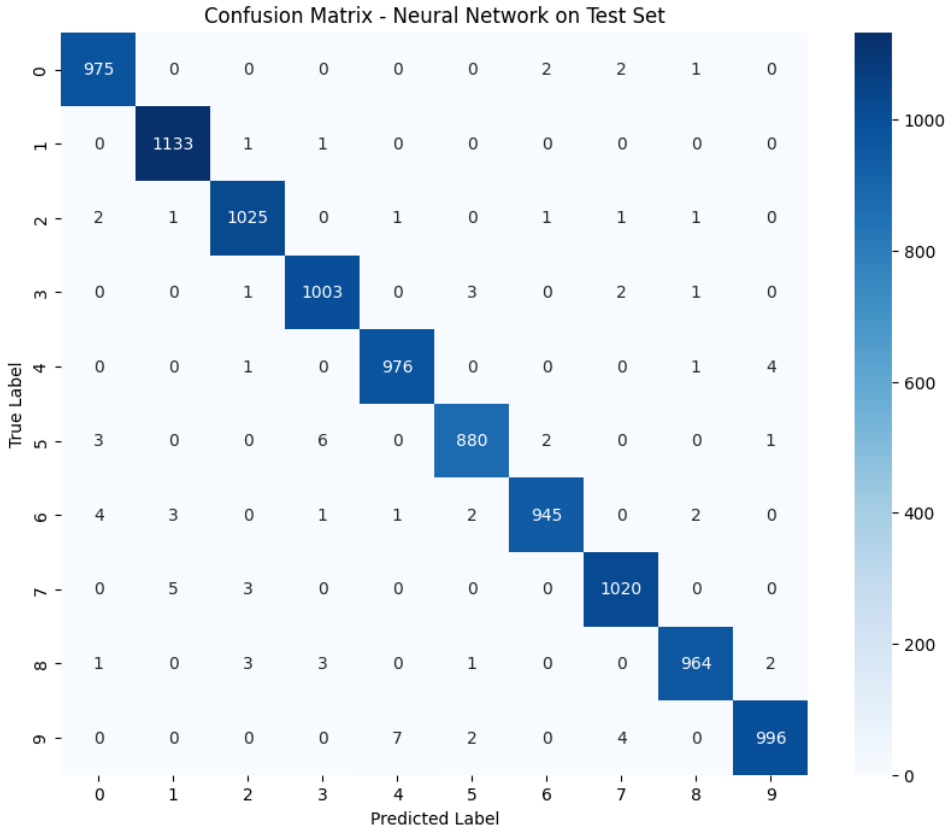

Fig. 1: Confusion Matrix for Neural Network on Test Set

IV. Discussion

Through this investigation, insights emerged regarding the trade-offs inherent in each classification method. The Neural Network's high performance highlights the power of deep learning for complex tasks, yet its computational demands and interpretability challenges cannot be overlooked. Conversely, Random Forest remains a strong candidate for its balance of power, ease of use, and robustness, albeit with less transparency than simpler models.

Surprisingly, simpler models like Logistic Regression still provided respectable baselines despite their limitation in handling non-linear separations. Future work could involve exploring even more sophisticated architectures such as deeper and wider networks or using transfer learning for further performance gains.

Practical implications include considering computational constraints in the choice of model, especially for applications requiring real-time processing. This study reiterates that no one-size-fits-all solution exists; model selection must align with problem specifications and resource availability.

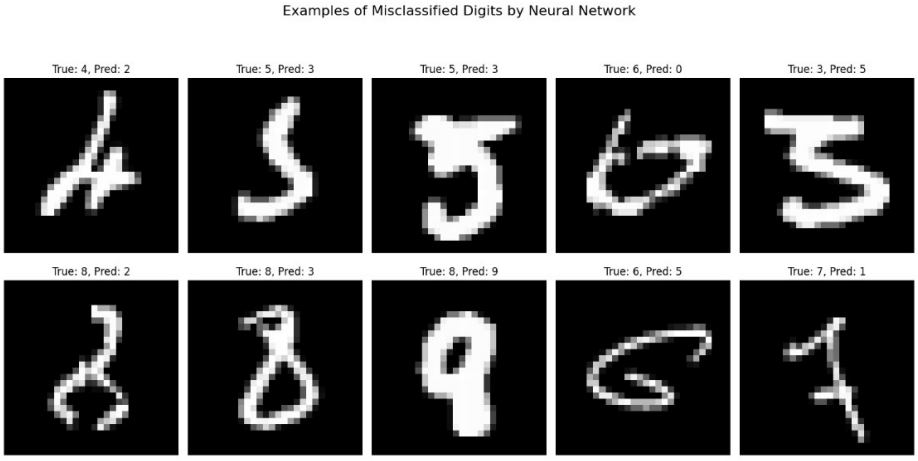

Fig. 2: Examples of Misclassified Digits by Neural Network

V. Conclusion

This study explored various machine learning models, including logistic regression, LDA, random forest, decision tree, and neural network, to classify the MNIST dataset. We demonstrated that increasing the complexity of the neural network by adding more layers, increasing the number of epochs, and refining the model architecture led to a modest 2% improvement in accuracy over the baseline model. This highlights not only the power of neural networks but also the challenges inherent in improving model performance for complex tasks like digit classification.

References

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
[2] Y. LeCun, C. Cortes, and C. J. Burges, "The mnist database of handwritten digits," *http://yann.lecun.com/exdb/mnist*, 1998.
[3] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.
[4] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of Eugenics*, vol. 7, no. 2, pp. 179–188, 1936.
[5] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and regression trees*. CRC Press, 1984.
[6] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.
[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
[8] C. M. Bishop, *Pattern recognition and machine learning*. Springer, 2006.