Handwritten Digit Classification: A Performance Comparison of Decision Trees, Random Forests, and Neural Networks

Brandon Park, Jungjae Park, Kyunghwan Min, Minjae Lee, Minseo Jang
Stony Brook University
AMS 325
Harry Markakis

Abstract and the Project Objectives

In this project we consider the classical handwritten digit classification problem, where the goal is to assign each 28×28 grayscale image of a digit (0–9) to the correct class. In class we have recently studied single–layer and multi–layer neural networks, and we have seen that they provide a natural framework for solving this type of image classification task.

Our main objective is to implement such neural network models for the digit dataset and evaluate their performance. In addition, we would like to understand how these neural networks compare to more traditional machine-learning methods that were not covered in detail in class, such as decision trees and random forests. Using Python (scikit-learn and a simple neural-network library), we will train all three types of models on the same data and compare them in terms of classification accuracy and computational cost (training and prediction time). This will allow us to see the trade-offs between model complexity, speed, and performance on a concrete example.

Techniques and Tools

To build a program capable of distinguishing handwritten digits from the MNIST feature dataset, we implemented several machine learning models in Python. Our pipeline uses NumPy, Pandas, scikit-learn, XGBoost, and TensorFlow/Keras, with extensive hyperparameter tuning for each model type. The following sections summarize how each method was constructed.

Single Layer Neural Networks

We began with a single-layer neural network as a baseline for comparison. Each MNIST image is represented as a high-dimensional feature vector, and the network maps these features to 10 output logits followed by a softmax activation. The model is trained using cross-entropy loss and evaluated on a held-out validation set. This provides a simple benchmark against which deeper neural architectures and tree-based methods can be compared.

Multilayer Neural Networks

To capture more complex nonlinear patterns in the MNIST features, we implemented a multilayer neural network using TensorFlow/Keras.

Our code performs an extensive grid search over:

- Number of hidden units
- Dropout rates
- Batch sizes

Early stopping is used to prevent overfitting. After identifying the best-performing architecture based on validation error, we retrain the model on the full training set and evaluate it on the test set. This model represents the most flexible and expressive approach in our study.

Random Forest

We implemented Random Forests as a strong classical machine learning baseline. Before training, we remove monochrome (low-variance) pixel columns to reduce noise and dimensionality.

Key elements of our Random Forest implementation:

- Two candidate mtry values are tested: sqrt(p) and 0.5*p
- Each forest uses N = 5,000 trees
- Out-of-Bag (OOB) error is used to select the better mtry
- Models are tested on the full 0–9 multiclass task
- A variable-importance plot highlights the features most useful for distinguishing digits

This method serves as a reliable high-performance nonparametric classifier.

Boosted Trees

We also implemented boosted decision trees using XGBoost, which builds an ensemble of weak learners where each tree fits the residual errors of the previous model.

Our code performs a grid search over:

- Number of boosting rounds (10,000–30,000)
- Maximum tree depth
- Fixed learning rate

The best configuration is chosen based on validation error and then retrained on the full dataset. Predictions on the test set allow direct comparison with Random Forests and neural networks in terms of accuracy and computational cost.

Observations and Conclusions

Random Forest

Random Forests provided a strong and reliable baseline for handwritten digit classification. By averaging a large number of decision trees trained on bootstrap samples and random subsets of features, the Random Forest effectively reduced variance while maintaining high predictive accuracy. In our experiments, selecting mtry = sqrt{p} consistently outperformed larger feature subsets, confirming the theoretical principle that decorrelating trees improves ensemble performance.

The use of Out-of-Bag error as an internal validation measure allowed efficient hyperparameter selection without requiring a separate validation set. Overall, Random Forests achieved competitive classification accuracy on the full 0–9 digit task with relatively modest tuning effort. They also exhibited stable performance and robustness to overfitting, making them an effective classical alternative to neural networks for high-dimensional image data.

Boosted Tree

Boosted decision trees achieved the highest classification accuracy than random forest. By sequentially fitting trees to the residual errors of previous models, boosting was able to capture subtle structure in the data that single trees and Random Forests could not. To better study model capacity and overfitting behavior, we simplified the task to a binary classification problem (digits 3 vs. 8). In this setting, increasing the number of boosting rounds and tree depth initially led to improved validation performance, but gains eventually plateaued. This behavior is consistent with boosting approaching its capacity limit in a simpler learning task. The use of a very small learning rate and moderate tree depths acted as strong regularization, preventing a clear increase in validation error due to overfitting within the tested parameter range. Although boosted trees required substantially more computational time than Random Forests, their superior accuracy highlights the effectiveness of sequential error correction when sufficient computational resources are available.

Single-Layer Neural Network

The single-layer neural network served as a simple baseline for evaluating the effectiveness of neural models on the handwritten digit classification task. Despite its limited depth, the model achieved reasonable classification accuracy, demonstrating that much of the MNIST structure can already be captured through a single nonlinear transformation of the input features. However, achieving good performance required a relatively large number of hidden units, since the single hidden layer was responsible for both feature extraction and classification simultaneously.

Hyperparameter tuning showed that mild regularization through dropout was sufficient, while stronger dropout values led to underfitting and degraded validation performance. Larger batch sizes consistently produced more stable training and lower validation error, particularly when combined with the Adam optimizer and a small learning rate. Overall, the single-layer neural network provided a strong and computationally efficient baseline, but its representational capacity was inherently limited compared to deeper architectures.

Multilayer Neural Network

The multilayer neural network achieved improved performance over the single-layer model by distributing representational capacity across multiple hidden layers. The best-performing architecture used a larger first hidden layer followed by a smaller second hidden layer, reflecting a hierarchical learning process in which early layers extract coarse features and later layers refine them. This structure allowed the model to reduce bias more effectively without requiring a dramatic increase in total parameters.

As with the single-layer network, low dropout rates provided sufficient regularization, while higher dropout values consistently led to underfitting. Larger batch sizes again yielded more stable gradient updates and improved validation performance, indicating that batch size primarily influenced optimization stability rather than model capacity. Although the multilayer network required greater computational cost during training, it achieved lower validation and test errors, confirming the benefit of depth for learning more expressive representations of handwritten digits. The observed gains were moderate, suggesting diminishing returns from increasing depth on a well-structured dataset such as MNIST.

Conclusion

In this project, we compared neural network architectures with tree-based methods for handwritten digit classification on the MNIST dataset. Random Forests provided a strong and robust classical baseline with minimal tuning, while boosted trees achieved the highest accuracy by sequentially correcting errors at the cost of increased computational time. Among neural approaches, the single-layer neural network offered a fast and effective baseline, but its limited depth constrained representational power. The multilayer neural network consistently outperformed the single-layer model by learning hierarchical feature representations, though the improvement was moderate due to the structured nature of the MNIST dataset. Overall, our results highlight the trade-offs between model complexity, accuracy, and computational cost, demonstrating that deeper or more sophisticated models yield diminishing returns on well-studied problems when simpler methods already perform strongly.

Additional Notes and Limitations

Due to differences in computational environments, the training time comparisons across models were not fully consistent. The Random Forest and Boosted Tree models were trained on one server, while the single-layer and multilayer neural networks were trained on a different server equipped with GPU acceleration. As a result, direct comparisons of training time across all model classes are not strictly meaningful, since the latter benefited from GPU-based computation.

Furthermore, because of time and computational constraints, we were unable to exhaustively explore all possible behaviors of the models. In particular, we did not observe clear overfitting effects in some settings, nor did we see dramatic performance improvements from increased network depth beyond two hidden layers. This suggests that, on a well-structured dataset such as MNIST, additional model complexity may yield diminishing returns, especially when practical constraints limit extensive experimentation.

References

Bernard, S., Adam, S., & Heutte, L. (2007, September). Using random forests for handwritten digit recognition. In Ninth International Conference on Document Analysis and Recognition (ICDAR 2007) (Vol. 2, pp. 1043–1047). IEEE. https://doi.org/10.1109/ICDAR.2007.4377074

Voloshchenko, O., & Plechawska-Wójcik, M. (2021). Comparison of classical machine learning algorithms in the task of handwritten digits classification. Journal of Computer Sciences Institute, 21, 279–286. https://doi.org/10.35784/jcsi.2723

Chimminiyan, S. (2025). Neural Network from Scratch - MNIST. Kaggle Notebook. Retrieved December 3, 2025, from https://www.kaggle.com/code/samithsachidanandan/neural-network-from-scratch-mnist

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). *An introduction to statistical learning: With applications in R* (2nd ed.). Springer.

LeCun, Y., Cortes, C., & Burges, C. J. C. (1998). **The MNIST database of handwritten digits**. http://yann.lecun.com/exdb/mnist/

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., … Duchesnay, É. (2011). *Scikit-learn: Machine learning in Python*. *Journal of Machine Learning Research, 12*, 2825–2830.

Chen, T., & Guestrin, C. (2016). *XGBoost: A scalable tree boosting system*. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785–794). ACM.

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., … Zheng, X. (2016). *TensorFlow: Large-scale machine learning on heterogeneous systems*. https://www.tensorflow.org/

Chollet, F., et al. (2015). *Keras*. https://keras.io