Digital Object Identifier

Comparative Analysis of Machine Learning Models on MNIST: Performance, Efficiency, and Interpretability

GIDI RABI[1], ROI BRUCHIM[1]
[1]School of Computer Science, Ariel University, Israel.
Corresponding author: Gidi Rabi (e-mail: Gidirabi111@gmail.com). Roi Bruchim (e-mail: Roibr23@gmail.com).

ABSTRACT According to [1] Handwritten digit classification is a fundamental problem in machine learning, with applications spanning optical character recognition (OCR) and automated document processing. The MNIST dataset serves as a benchmark for evaluating different classification models in terms of accuracy, computational efficiency, and interpretability. In this study, we compare the performance of traditional machine learning algorithms—Logistic Regression, Support Vector Machines (SVM), Random Forest, and K-Nearest Neighbors (KNN)—against deep learning models such as Convolutional Neural Networks (CNNs). We analyze the trade-offs between model complexity, training time, inference speed, and classification accuracy. Our empirical results show that CNN achieves the highest accuracy (99.02%) while maintaining efficient inference, making it the best choice for high-performance applications. KNN, despite being simple, is computationally impractical due to its slow testing time. Random Forest offers a strong balance between accuracy (96.80%) and efficiency, while SVM proves to be computationally expensive for minimal accuracy improvement. By exploring per-digit misclassification patterns and hyperparameter tuning, we highlight the strengths and weaknesses of each model, providing insights into selecting the most appropriate algorithm for digit classification tasks.

INDEX TERMS MNIST, Handwritten Digit Recognition, Machine Learning, Classification Models, Convolutional Neural Networks, Support Vector Machines, Random Forest, Logistic Regression, K-Nearest Neighbors, Model Efficiency, Computational Complexity, Interpretability, Hyperparameter Optimization.

I. INTRODUCTION

Handwritten digit classification is a fundamental task in machine learning, widely used in optical character recognition (OCR) and automated form processing. The MNIST dataset serves as a benchmark for evaluating classification models, making it an ideal testbed for assessing the trade-offs between traditional and deep learning approaches.

Classical models such as Logistic Regression, Support Vector Machines (SVM), Random Forest, and K-Nearest Neighbors (KNN) offer interpretability and efficient training, but struggle with complex patterns. Convolutional Neural Networks (CNNs), leveraging spatial hierarchies and deep feature extraction, achieve state-of-the-art accuracy but demand greater computational resources. This raises the key question: does the added complexity justify the performance gains?

This study evaluates multiple models on MNIST, comparing accuracy, computational efficiency, and misclassification trends. We investigate the relationship between model complexity, training time, and inference speed to determine the most practical approach for digit classification.

Our findings provide insight into when deep learning is necessary versus when traditional models are sufficient, offering guidance for selecting models based on resource constraints and performance needs.

II. EXPERIMENTAL PREPARATION

To ensure a fair comparison of machine learning models on the MNIST dataset, we implemented five separate scripts—one for each model: Logistic Regression, Support

Vector Machines (SVM), Random Forest, K-Nearest Neighbors (KNN), and Convolutional Neural Networks (CNN). Each script followed a standardized evaluation pipeline to maintain consistency across models.

A. DATASET AND PREPROCESSING

The MNIST dataset consists of 70,000 grayscale images (28×28 pixels) of handwritten digits, divided into:

- **Training Set:** 60,000 images used for model training.
- **Testing Set:** 10,000 images for performance evaluation.

Traditional models (Logistic Regression, SVM, Random Forest, KNN) used flattened 784-dimensional feature vectors, while CNNs retained the 2D structure. Pixel values were normalized to [0,1] for consistency.

B. EVALUATION PIPELINE

Each model followed the same structured process:

- **Training:** Models were trained on the 60,000-image dataset.
- **Inference:** Predictions were made on the 10,000-image test set.
- **Metrics Recorded:** Training time, inference time, accuracy, and misclassification patterns.

C. IMPLEMENTATION AND TOOLS

All models were implemented using `Scikit-learn` (traditional models) and `TensorFlow/Keras` (CNN). The results from all five scripts were collected and analyzed to answer key research questions regarding model efficiency, accuracy, and interpretability.

III. RESEARCH QUESTIONS AND ANSWERS

This section provides a structured analysis of key research questions explored in this project.

A. HOW DOES COMPUTATION TIME (TRAINING/TESTING) COMPARE BETWEEN THE MODELS?

First, let's examine the table that shows the training and testing run times:

TABLE 1: Training and Testing Times for Each Model

Model	Training Time (s)	Testing Time (s)
CNN	99.81	3.62
KNN (K=1)	0.04	9.15
KNN (K=5)	0.03	10.71
KNN (K=15)	0.04	10.73
Logistic Regression	44.45	0.03
Random Forest	13.97	0.19
SVM	198.48	67.57

From the table, we can learn that KNN has almost no training time but is the slowest in testing, making it inefficient for large datasets. Logistic Regression and Random Forest have relatively fast training times and the quickest testing times. CNN takes longer to train but maintains a fast testing time, balancing accuracy and efficiency. SVM has the longest training and testing times, making it the least efficient computationally. This suggests that KNN is impractical for large-scale applications, SVM is too computationally expensive, and CNN is a good option for accuracy while keeping reasonable efficiency [2].

B. DOES THE ADDED COMPLEXITY OF CNN JUSTIFY ITS IMPROVED ACCURACY?

Based on the results, CNN achieves the highest accuracy at 99.02%, which is significantly better than the other models. While it takes longer to train (99.81s) compared to models like Logistic Regression (44.45s) and Random Forest (13.97s), its testing time (3.62s) is relatively fast, making it efficient for inference.

In contrast, SVM also achieves high accuracy (97.92%) but takes much longer to train (198.48s) and test (67.57s), making it less practical. KNN has lower accuracy (96.91% for $k = 1$) and suffers from very slow testing times despite near-instant training. Random Forest and Logistic Regression perform decently (96-92% accuracy) but fall short compared to CNN.

Given that CNN offers the best accuracy with reasonable inference speed, its added complexity is justified for applications where high accuracy is critical. However, for real-time or low-power applications, a simpler model like Random Forest might be more practical. Since CNN has the highest accuracy and a relatively fast testing time, it seems like the best choice for MNIST. It does take longer to train compared to simpler models, but its ability to maintain high accuracy makes up for it. The balance between accuracy and efficiency makes CNN a great fit for this dataset.

C. WHICH MODEL BRINGS THE BEST OVERALL RESULTS FOR THIS DATASET?

Based on the results, CNN clearly performs the best for this dataset. It achieves the highest accuracy (99.02%), which is significantly better than the other models, while still keeping a relatively fast testing time (3.62s). Even though it takes longer to train (99.81s), the improved performance makes it worth it.

SVM also performs well (97.92%) but takes way too long to train and test, making it less practical. KNN, despite its simplicity, has slower testing times and lower accuracy, while Logistic Regression and Random Forest are decent but not as accurate.

Overall, CNN strikes the best balance between accuracy and efficiency, making it the best choice for MNIST.

D. WHAT ARE THE TRADE-OFFS BETWEEN MODEL SIMPLICITY, ACCURACY, AND COMPUTATIONAL COST?

To determine the best model for MNIST, we use a model score formula that balances accuracy, training time, and testing time:

$$modelscore = \frac{Accuracy}{\alpha \cdot TrainingTime + \beta \cdot TestingTime + \gamma}$$

Since MNIST is often used in real-time applications, we assigned higher importance to accuracy and testing time, while reducing the weight of training time. Training is a one-time cost, whereas testing speed directly impacts usability in real-world scenarios. With this in mind, we set α (training time weight) to 0.02, keeping it low to minimize its influence, β (testing time weight) to 0.3, giving it more importance, and γ to 1, ensuring stability in the denominator.

Using this approach, we analyzed different models. Simpler models like Logistic Regression and KNN are quick to train, but they don't always perform well—Logistic Regression has decent speed but lower accuracy (92.60%), while KNN is simple but too slow in testing to be practical. Random Forest strikes the best balance, achieving 96.80% accuracy while keeping inference extremely fast (0.19s), making it ideal for real-time applications.

On the other hand, more complex models like CNN and SVM offer higher accuracy (99.02% and 97.92%, respectively), but at a much higher computational cost. CNN takes longer to train (99.81s) but still maintains a reasonable inference time (3.62s), making it a good choice when accuracy is the top priority. SVM, however, is extremely slow in both training (198.48s) and testing (67.57s), making it computationally expensive and impractical for MNIST. KNN, despite being a simple model, struggles with efficiency because every new input needs to be compared to the entire dataset.

After applying our model score formula, Random Forest ranked highest due to its strong balance of accuracy and efficiency, followed by Logistic Regression as a lightweight alternative. CNN is the best if accuracy is the top priority, but SVM and KNN are too slow to be practical.

In the end, the best model depends on the application—if accuracy is key, CNN is the way to go; if speed and efficiency are needed, Random Forest is the better choice.

E. WHAT IS THE RELATIONSHIP BETWEEN HYPERPARAMETER TUNING AND MODEL PERFORMANCE FOR EACH MODEL?

Hyperparameter tuning plays a crucial role in determining a model's performance. The right settings can enhance accuracy, efficiency, and generalization to new data.

For example, in CNNs, adjusting the number of filters, kernel size, and batch size affects how well the model captures patterns in images. A deeper network with more filters can improve accuracy, but it also increases training time and computational cost.

KNN, on the other hand, depends heavily on the choice of k. If $k = 1$, the model might overfit, leading to high variance, whereas a larger k (e.g., $k = 15$) smooths predictions but could reduce accuracy by oversimplifying decision boundaries.

In Logistic Regression, tuning parameters such as regularization strength and the solver can help prevent overfitting or underfitting. A well-chosen regularization parameter ensures the model remains flexible without memorizing noise in the training data.

For Random Forest, hyperparameters like the number of trees (`n_estimators=100`) and tree depth (`max_depth=20`) impact accuracy. More trees generally improve performance but also slow down inference and increase memory usage.

SVM is highly sensitive to its hyperparameters, especially the kernel choice and gamma value. The RBF kernel, commonly used for non-linear problems, is effective for capturing complex patterns. However, if parameters like gamma are set too high, the model might memorize the training data instead of learning general trends, leading to overfitting.

Overall, hyperparameter tuning is about finding the right balance. Poorly chosen hyperparameters can cause overfitting (where the model is too complex and memorizes data) or underfitting (where the model is too simple and misses important patterns). Since the optimal hyperparameters vary across datasets, testing different values through cross-validation is often necessary to achieve the best results.

F. WHICH DIGITS ARE FREQUENTLY MISCLASSIFIED?

From our earlier results, the digits that were most frequently misclassified varied by model, but some patterns were consistent:

- **CNN:**
  - **Most misclassified:** Digits 6 and 2 had the highest misclassification rates.
  - **Least misclassified:** Digit 1 had the best classification performance.
- **KNN (k=1, k=5, k=15):**
  - **Most misclassified:** Digits 8 and 2 were often confused.
  - **Least misclassified:** Digit 1 performed well across different k-values.
- **Logistic Regression:**
  - **Most misclassified:** Digits 8 and 5 had the highest errors.
  - **Least misclassified:** Digits 1 and 0 were the easiest to classify.
- **Random Forest:**
  - **Most misclassified:** Digits 8 and 9 had the highest misclassification.
  - **Least misclassified:** Digits 0 and 1 had strong classification rates.
- **SVM:**
  - **Most misclassified:** Digits 8 and 9 had the highest misclassification.
  - **Least misclassified:** Digit 1 performed the best.

1) Overall Patterns Across Models

- Digit 8 was the most frequently misclassified across multiple models.
- Digit 1 was consistently the easiest to classify.
- Digits 6, 2, 5, and 9 also appeared frequently in misclassification lists.

- Models that rely on feature-based separation (like Logistic Regression and Random Forest) struggled more with curved digits like 8 and 9 [3].

TABLE 2: Per-digit classification accuracy across different models

Digit	CNN (%)	KNN (k=5) (%)	Logistic Regression (%)	Random Forest (%)	SVM (%)
0	99.39	99.39	97.76	98.98	99.29
1	99.82	99.82	97.80	98.85	99.21
2	98.45	96.03	90.21	96.61	97.48
3	99.11	96.63	91.39	96.34	98.51
4	99.39	96.13	93.89	96.95	97.86
5	99.33	96.64	87.11	96.19	97.65
6	98.02	98.64	95.09	97.49	98.54
7	99.22	96.11	92.22	95.62	96.89
8	98.87	93.74	87.78	95.38	97.54
9	98.51	95.34	91.67	95.34	96.13

G. WHY ARE CERTAIN DIGITS HARDER TO CLASSIFY? IS IT DUE TO THEIR SIMILARITY OR VARIABILITY IN WRITING STYLES?

Based on the results, digits **8, 2, and 9** were the most frequently misclassified across multiple models, while **digits 1 and 0** were consistently the easiest to classify. This suggests that **digit similarity and handwriting variability** play a major role in classification difficulty.

For example, **digit 8** was one of the hardest to classify across all models, likely because it resembles **3, 5, or even 6** in different handwriting styles. Similarly, **digit 2** had a lower success rate, especially in KNN and Logistic Regression, possibly due to its resemblance to **7 or 3** when written with curves.

CNN performed best on these difficult digits because it learns spatial patterns, while **KNN and Logistic Regression struggled the most due to their reliance on raw pixel comparisons**. **Random Forest and SVM** performed well overall but still had difficulty with digits that had high intra-class variation, like **8 and 9**.

To illustrate these misclassification challenges, **Figure 1** presents **examples of digits misclassified by the Random Forest model**, showing how certain digits were incorrectly predicted.

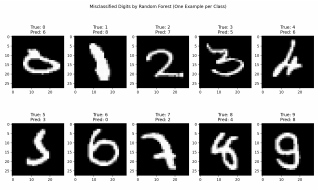


FIGURE 1: Examples of misclassified digits by the Random Forest model.

These results emphasize the role of feature selection and model architecture. **CNNs excel at capturing handwriting variations, while simpler models struggle with similar digits.**

H. WHICH MODELS ARE MORE INTERPRETABLE (E.G., DECISION PATHS IN RANDOM FORESTS VS. CNN FEATURE MAPS)?

When it comes to understanding how a model makes decisions, Random Forest and Logistic Regression are the easiest to interpret, while CNN and SVM are more like black boxes [4].

- **Random Forest:** Offers high interpretability as one can trace the decision path in each tree, making it clear why a digit was classified a certain way. Each tree's structure provides insights into the feature importance and decision-making process.
- **Logistic Regression:** Simple to understand since it assigns weights to each pixel, effectively showing which areas of an image contributed most to the classification decision. This transparency makes it easier to diagnose and refine the model.
- **KNN:** Somewhat interpretable because you can identify which training examples influenced a prediction. However, it does not provide insight into why those specific neighbors were chosen or how the distance metric impacts decisions.
- **CNN:** While achieving the highest accuracy, CNNs are less interpretable. The filters and multiple layers learn complex and abstract patterns that require visualizing feature maps to comprehend. Techniques like Grad-CAM or filter visualization can help but still offer limited interpretability [4].
- **SVM:** Also considered less interpretable as it constructs decision boundaries in high-dimensional space. Understanding the impact of support vectors and kernel functions requires deeper mathematical insight, making it challenging to elucidate specific classifications.

Overall, simpler models like Random Forest and Logistic Regression provide greater transparency, making them suitable for applications where interpretability is crucial. In contrast, complex models like CNN and SVM excel in accuracy but at the cost of reduced interpretability. This trade-off must be considered when choosing a model for practical applications.

I. HOW DOES TRAINING TIME CORRELATE WITH MODEL COMPLEXITY AND PERFORMANCE IN HANDWRITTEN DIGIT CLASSIFICATION?

From the results, it is evident that more complex models take longer to train but generally yield better classification performance. Logistic Regression and KNN exhibit the fastest training times, with KNN being almost instantaneous. However, their ability to capture intricate patterns is limited. Logistic Regression achieves an accuracy of 92.60%, whereas KNN performs better (96.91% for $k = 1$) but suffers from extremely slow inference, making it impractical for large-scale datasets [2].

Random Forest strikes a balance between training efficiency and accuracy—it completes training in 13.97 seconds

while achieving 96.80% accuracy, positioning itself as an efficient choice. In contrast, SVM has the longest training duration (198.48 seconds) but offers only a marginal accuracy improvement (97.92%), rendering it computationally costly with limited performance gain.

CNN, despite its long training time (99.81 seconds), attains the highest accuracy (99.02%) and maintains a relatively fast inference time (3.62 seconds), making it the optimal choice for high-performance applications. Ultimately, the findings indicate that simpler models are faster to train but less accurate, CNNs deliver superior accuracy at a higher training cost, and Random Forest provides the most practical trade-off between speed and performance.

TABLE 3: Model Complexity, Training Time, and Performance

Model	Training Time	Testing Time	Accuracy
Logistic Regression	44.45s	0.03s	92.60%
KNN ($k = 1$)	0.04s	9.15s	96.91%
KNN ($k = 5$)	0.03s	10.71s	96.88%
KNN ($k = 15$)	0.04s	10.73s	96.33%
Random Forest	13.97s	0.19s	96.80%
SVM	198.48s	67.57s	97.92%
CNN	99.81s	3.62s	99.02%

J. RELATIONSHIP BETWEEN TRAINING TIME AND ACCURACY

The correlation between training time and accuracy is evident in Figure 2. Simpler models such as Logistic Regression and KNN train rapidly but plateau at lower accuracy levels. As model complexity increases, so does accuracy, albeit with diminishing returns. CNN exhibits the most favorable trade-off, achieving high accuracy while keeping inference relatively fast. The efficiency of Random Forest also stands out, offering competitive accuracy with significantly lower computational costs compared to SVM.

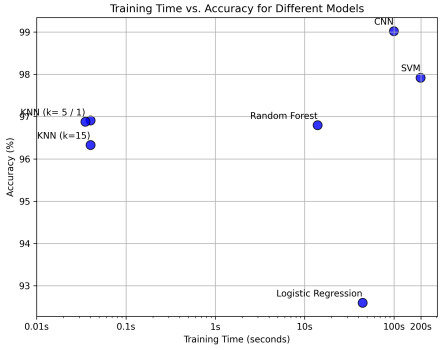


FIGURE 2: Training Time vs. Accuracy for Different Models

The trade-offs among training time, model complexity, and performance underscore the necessity of selecting a model based on application needs. If computational resources and inference speed are constraints, Random Forest provides an efficient solution. For tasks requiring high accuracy with minimal inference delay, CNN is the superior choice. Conversely, SVM, despite its strong classification capability, is computationally demanding and may not be suitable for real-time applications.

IV. CONCLUSION

This study conducted a comparative analysis of machine learning models for handwritten digit classification on the MNIST dataset, evaluating their accuracy, computational efficiency, and interpretability. We examined traditional models such as Logistic Regression, Support Vector Machines (SVM), Random Forest, and K-Nearest Neighbors (KNN) alongside deep learning models like Convolutional Neural Networks (CNNs).

Our findings demonstrate that CNN achieves the highest classification accuracy (99.02%) while maintaining a relatively fast inference time (3.62s), making it the most suitable choice when accuracy is the top priority. However, CNN's higher training cost (99.81s) may not be justified for all applications. In contrast, Random Forest offers a strong balance between accuracy (96.80%) and efficiency, making it a practical alternative for scenarios where computational resources are constrained. KNN, despite its simplicity, suffers from slow inference, limiting its scalability, while SVM, though achieving high accuracy (97.92%), is computationally expensive for only marginal performance improvements.

The analysis of per-digit misclassification patterns revealed that certain digits, such as **8 and 9**, were consistently harder to classify across multiple models, highlighting the challenges of digit similarity and handwriting variability. Additionally, hyperparameter tuning played a critical role in optimizing model performance, with CNN and SVM benefiting the most from parameter adjustments.

Ultimately, the choice of the best model depends on the trade-off between accuracy, computational complexity, and interpretability. CNN is preferable when maximizing accuracy is essential, while Random Forest provides a more efficient solution for real-time applications. Models like SVM and KNN may be less practical due to their computational costs, particularly for large-scale datasets [2].

ACKNOWLEDGMENT

We would like to thank **Gidi Rabi** and **Roi Bruchim** for their contributions to this research.

REFERENCES

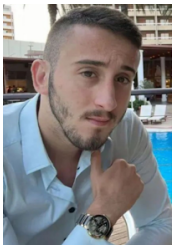
[1] Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. IEEE signal processing magazine, 29(6):141–142, 2012.

[2] Alexandra L'heureux, Katarina Grolinger, Hany F Elyamany, and Miriam AM Capretz. Machine learning with big data: Challenges and approaches. Ieee Access, 5:7776–7797, 2017.

[3] Simon Bernard, Sébastien Adam, and Laurent Heutte. Using random forests for handwritten digit recognition. In Ninth international conference on document analysis and recognition (ICDAR 2007), volume 2, pages 1043–1047. IEEE, 2007.

[4] Shivam S Kadam, Amol C Adamuthe, and Ashwini B Patil. Cnn model for image classification on mnist and fashion-mnist dataset. Journal of scientific research, 64(2):374–384, 2020.

GIDI RABI is an undergraduate B.Sc. student in Computer Science at the School of Computer Science at Ariel University, Israel. His research focuses on machine learning and artificial intelligence, with a particular interest in model evaluation and optimization. Under the supervision of Prof. Leed Gotlib, Gidi has worked on comparative analyses of classification models, including Convolutional Neural Networks (CNNs), Support Vector Machines (SVMs), and decision trees. His work explores the trade-offs between accuracy, interpretability, and computational efficiency in real-world applications. Gidi is also passionate about deep learning and its applications in computer vision and automation.



ROI BRUCHIM is an undergraduate B.Sc. student in Computer Science at the School of Computer Science at Ariel University, Israel. His research interests include machine learning, statistical modeling, and data science. Under the guidance of Prof. Leed Gotlib, Roi has been involved in projects analyzing the performance of different classification models on large-scale datasets such as MNIST. His work evaluates model efficiency, interpretability, and hyperparameter tuning to optimize performance. Roi is particularly interested in exploring how different machine learning algorithms generalize to unseen data and their practical implications in automated decision-making systems.