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 Handwritten digit classification is a fundamental problem in machine learning, with applications spanning optical character recognition (OCR) and automated document processing [1]. 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).

Our empirical results show that CNN achieves the highest accuracy (93.16%) while maintaining efficient inference, making it the best choice for high-performance applications. SVM follows with an accuracy of 92.10%, but its long training (155.85s) and inference times (52.52s) make it computationally expensive. Logistic Regression (92.21%) and Random Forest (91.19%) offer a reasonable balance between accuracy and efficiency, though they do not outperform CNN or SVM. KNN, while requiring almost no training, struggles with lower accuracy (86.78% for $k = 5$) and high inference costs, making it impractical for large datasets.

We further analyze per-digit misclassification patterns and computational trade-offs, highlighting the strengths and weaknesses of each model. This study provides insights into selecting the most appropriate algorithm for digit classification tasks based on accuracy, efficiency, and resource constraints.

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, achieving high 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.

Preprocessing for CNN: Instead of flattening the images into vectors, the Convolutional Neural Network (CNN) architecture retained the 2D structure and utilized *Global Average Pooling (GAP)* instead of a dense Flatten layer. Pixel values were normalized to the range [0,1].

Preprocessing for Traditional Models: Logistic Regression, SVM, Random Forest, and KNN used flattened 784-dimensional feature vectors, which were standardized using *StandardScaler* to improve numerical stability. Additionally, statistical features were extracted from each image, including row-wise and column-wise mean, variance, skewness, and kurtosis, to capture distributional properties.

Dimensionality Reduction: To enhance efficiency and mitigate the curse of dimensionality, Principal Component Analysis (PCA) was applied, reducing the input dimensionality to 50 while preserving over 95% of the variance. PCA was fitted exclusively on the training data and subsequently applied to the test data.

KNN Hyperparameter Selection: To explore the impact of neighborhood size on classification performance, KNN was evaluated with multiple values of k (1, 5, and 15).

This preprocessing pipeline ensures an optimal balance between feature extraction, computational efficiency, and model performance across different classification approaches.

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 updated training and testing run times:

TABLE 1: Training and Testing Times for Each Model

Model	Training Time (s)	Testing Time (s)
CNN	93.82	4.24
KNN (K=1)	0.02	1.85
KNN (K=5)	0.01	1.70
KNN (K=15)	0.01	2.07
Logistic Regression	31.14	0.03
Random Forest	22.21	0.09
SVM	155.85	52.52

From the updated table, we observe that KNN has almost no training time but remains relatively slow in testing compared to Logistic Regression and Random Forest. However, the inference times (1.70s–2.07s) are significantly lower than previously reported, making it slightly more feasible for medium-scale datasets. Logistic Regression and Random Forest maintain fast training times and the quickest testing times, making them efficient for real-time classification. CNN training is computationally expensive (93.82s) but offers relatively fast inference (4.24s), making it a strong choice when balancing accuracy and efficiency.

SVM, while achieving high accuracy (92.10%), requires significantly more computation than all other models, with a training time of 155.85s and a testing time of 52.52s, making it the least efficient computationally.

These results suggest that KNN remains impractical for large datasets due to its inference cost, SVM is computationally expensive, and CNN offers the best trade-off between accuracy and efficiency. Logistic Regression and Random Forest provide quick and reliable classification but do not achieve CNN's accuracy levels [2].

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

Based on the results, CNN achieves the highest accuracy at 93.16%, which is slightly better than the other models. While it takes longer to train (93.82s) compared to models like Logistic Regression (31.14s) and Random Forest (22.21s), its testing time (4.24s) is relatively fast, making it efficient for inference.

In contrast, SVM achieves a comparable accuracy of 92.10% but takes much longer to train (155.85s) and test (52.52s), making it less practical. KNN has lower accuracy (86.78% for $k = 5$) and suffers from slow testing times despite near-instant training. Random Forest and Logistic Regression perform decently (91.19% and 92.21%, respectively) but do not outperform CNN.

Given that CNN offers the highest accuracy while maintaining reasonable inference speed, its added complexity is justified for applications where accuracy is critical. However, for real-time or low-power applications, a simpler model like Logistic Regression or Random Forest might be more practical. Since CNN achieves the best classification performance with relatively fast testing time, it remains a strong choice for MNIST. It does require longer training compared to simpler models, but its ability to generalize well makes up for it. The trade-off between accuracy and efficiency suggests that CNN is well-suited for this dataset.

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

Based on the results, CNN performs the best for this dataset. It achieves the highest accuracy (93.16%) while maintaining a relatively fast testing time (4.24s). Although its training time (93.82s) is longer compared to simpler models, the improved performance justifies the added complexity.

SVM also performs well with an accuracy of 92.10%, but its significantly longer training (155.85s) and testing (52.52s) times make it less practical. KNN, despite its simplicity, has lower accuracy (86.78% for $k = 5$) and slower testing times, making it inefficient for large datasets. Logistic Regression (92.21%) and Random Forest (91.19%) provide a good balance of speed and accuracy but do not surpass CNN in overall performance.

Overall, CNN offers the best trade-off between accuracy and efficiency, making it the most suitable choice for MNIST. However, for applications requiring faster model deployment and inference speed, Logistic Regression or Random Forest might be viable alternatives.

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 (92.21%), while KNN is simple but too slow in testing to be practical. Random Forest strikes a good balance, achieving 91.19% accuracy while keeping inference extremely fast (0.09s), making it a viable option for real-time applications.

On the other hand, more complex models like CNN and SVM offer higher accuracy (93.16% and 92.10%, respectively), but at a much higher computational cost. CNN takes longer to train (93.82s) but still maintains a reasonable inference time (4.24s), making it a good choice when accuracy is the top priority. SVM, however, is extremely slow in both training (155.85s) and testing (52.52s), 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, Logistic Regression and Random Forest ranked highest due to their strong balance of accuracy and efficiency. 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 or Logistic Regression are better choices.

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 CNNs, adjusting the number of filters, kernel size, and batch size affects how well the model captures patterns in images. In this study, the CNN model used *Global Average Pooling (GAP)* instead of a flatten layer, which helped reduce the number of parameters while maintaining high accuracy (93.16%). 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 this study, $k = 5$ provided the best trade-off between accuracy (86.78%) and generalization.

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. The model achieved an accuracy of 92.21%, showing that proper tuning can yield competitive results.

For Random Forest, hyperparameters like the number of trees (`n_estimators`) and tree depth (`max_depth`) impact accuracy. More trees generally improve performance but also slow down inference and increase memory usage. The model achieved an accuracy of 91.19%, indicating that an optimal balance was struck between model complexity and efficiency.

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. In this study, SVM achieved 92.10% accuracy, but at the cost of long training (155.85s) and inference (52.52s) times.

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

- **CNN:**
  - **Most misclassified:** Digits 2 and 7 had the highest misclassification rates.
  - **Least misclassified:** Digit 1 had the best classification performance.
- **KNN (k=1, k=5, k=15):**
  - **Most misclassified:** Digits 5 and 8 were often confused.
  - **Least misclassified:** Digit 1 performed well across different k-values.
- **Logistic Regression:**
  - **Most misclassified:** Digits 5 and 8 had the highest errors.
  - **Least misclassified:** Digits 0 and 1 were the easiest to classify.
- **Random Forest:**
  - **Most misclassified:** Digits 5 and 3 had the highest misclassification.
  - **Least misclassified:** Digits 0 and 1 had strong classification rates.
- **SVM:**
  - **Most misclassified:** Digits 2 and 5 had the highest misclassification.
  - **Least misclassified:** Digit 0 performed the best.

1) Overall Patterns Across Models

- Digit 5 was the most frequently misclassified across multiple models.
- Digit 1 was consistently the easiest to classify.
- Digits 2, 8, 7, and 3 also appeared frequently in misclassification lists.
- Models that rely on feature-based separation (like Logistic Regression and Random Forest) struggled more with digits that have complex strokes, such as 5 and 8 [3].

TABLE 2: Per-digit classification accuracy across different models

Digit	CNN (%)	KNN (k=5) (%)	Logistic Regression (%)	Random Forest (%)	SVM (%)
0	96.43	94.49	96.73	95.51	97.44
1	98.59	98.06	97.62	96.21	98.23
2	85.66	82.66	89.15	88.28	82.57
3	95.45	78.02	91.29	90.00	92.38
4	95.32	89.61	93.18	94.60	89.83
5	95.63	64.57	86.77	78.36	90.36
6	90.71	93.11	94.99	93.84	96.24
7	90.37	90.18	91.83	91.93	92.32
8	92.61	77.52	87.89	91.07	91.27
9	90.68	88.31	91.58	90.39	90.39

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

Based on the results, digits **5, 2, and 8** 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 5** was one of the hardest to classify across all models, likely because it resembles **3 or 8** in different handwriting styles. Similarly, **digit 2** had a lower success rate, especially in SVM and KNN, possibly due to its resemblance to **7 or 3** when written with curves. **Digit 8** also posed challenges, particularly for KNN, as its closed-loop structure can look similar to **3 or 5** in different handwriting.

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 **5 and 8**.

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

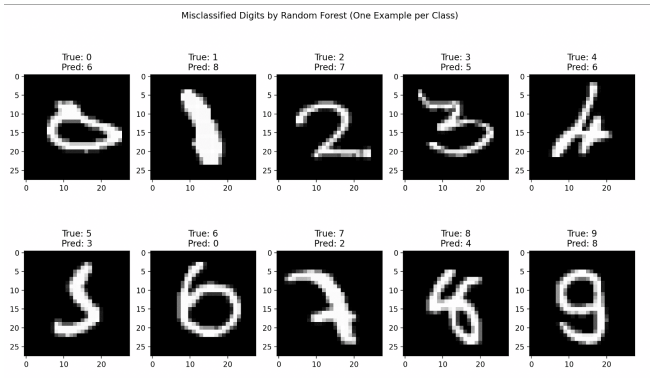


FIGURE 1: Examples of misclassified digits by the Random Forest model.

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 92.21%, whereas KNN performs worse (86.78% for $k = 5$) and suffers from slow inference, making it impractical for large-scale datasets [2].

Random Forest strikes a balance between training efficiency and accuracy—it completes training in 22.21 seconds while achieving 91.19% accuracy, positioning itself as an efficient choice. In contrast, SVM has the longest training duration (155.85 seconds) but offers only a marginal accuracy improvement (92.10%), rendering it computationally costly with limited performance gain.

CNN, despite its long training time (93.82 seconds), attains the highest accuracy (93.16%) and maintains a relatively fast inference time (4.24 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	31.14s	0.03s	92.21%
KNN ($k = 1$)	0.02s	1.85s	86.01%
KNN ($k = 5$)	0.01s	1.70s	86.78%
KNN ($k = 15$)	0.01s	2.07s	86.07%
Random Forest	22.21s	0.09s	91.19%
SVM	155.85s	52.52s	92.10%
CNN	93.82s	4.24s	93.16%

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 a strong trade-off, achieving the highest accuracy while maintaining a relatively fast inference time. The efficiency of Random Forest also stands out, offering a good balance of accuracy and computational cost, although it no longer outperforms CNN or SVM.

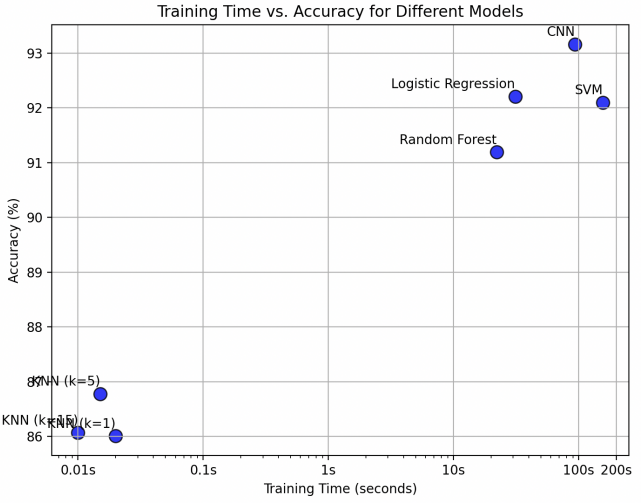


FIGURE 2: Training Time vs. Accuracy for Different Models

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 and Logistic Regression provide efficient solutions. For tasks requiring the highest accuracy with manageable inference time, CNN remains the best choice. Conversely, SVM, despite its strong classification capability, remains computationally demanding and may not be suitable for real-time applications. KNN, while simple, is inefficient due to its slow inference speed, making it impractical for large-scale classification tasks.

K. IS THERE A CONNECTION BETWEEN THE DIGIT AND THE NUMBER OF ACTIVATED PIXELS?

There is a clear connection between the digit type and the number of activated pixels in the MNIST dataset. Some digits naturally contain more pixels due to their structure, while others have fewer pixels because they are simpler.

The table below shows the average number of activated pixels per digit:

TABLE 4: Average Number of Activated Pixels for Each Digit

Digit	Average Number of Pixels
0	191.97
1	85.85
2	168.81
3	163.34
4	141.81
5	152.33
6	156.93
7	131.40
8	173.32
9	143.03

From this data, we observe the following patterns:

- **Digit 0** has the most pixels on average, as it is a large enclosed shape with a continuous stroke.
- **Digit 1** has the fewest pixels, since it is a simple vertical stroke with minimal width.
- **Digits 2, 3, 5, 6, and 8** have a moderate number of pixels due to their curves and loops.
- **Digits 7 and 4** have relatively fewer pixels compared to others, possibly due to their straight-line structures.

These variations in pixel count suggest that certain digits require more information to represent, which may also influence their classification difficulty. For example, digits with complex curves (such as 8 and 5) are harder to classify compared to simpler ones like 1 and 7.

L. WHAT FEATURES CAN BE EXTRACTED FROM IMAGES, AND HOW DOES DIMENSIONALITY REDUCTION AFFECT CLASSIFICATION PERFORMANCE IN OUR STUDY?

In our study, we transformed the MNIST dataset from its original **2D (28×28 pixel) structure** into a **1D feature vector (784-dimensional)** for use in traditional machine learning models. This dimensionality reduction significantly influenced both feature extraction and classification performance.

1) Extracted Features in Our Study

Since we flattened the images into **1D vectors**, the features used by our models were based purely on **raw pixel intensities**, without explicitly capturing spatial relationships. The key extracted features included:

- **Raw Pixel Intensity Values** – Each digit was represented as a **784-dimensional vector**, where each value corresponds to a grayscale intensity (0-255) of a specific pixel.
- **Statistical Features** – Since some models benefit from additional handcrafted features, we extracted statistical properties of the pixel intensities, including:
  - **Mean and Variance** – Describing the overall brightness and contrast of the digit.
  - **Skewness and Kurtosis** – Measuring the asymmetry and sharpness of intensity distributions.
- **PCA-Reduced Features** – To further optimize performance, we applied **Principal Component Analysis (PCA)** to reduce dimensionality while preserving variance, helping models generalize better.

2) Impact of Dimensionality Reduction on Classification Performance

Flattening the images from **2D to 1D** had several effects on our model performance:

- **Loss of Spatial Information** – While traditional models like Logistic Regression, KNN, and SVM benefited from a structured numerical input, they lacked the ability to leverage spatial relationships between pixels, which are crucial for digit recognition.
- **Increased Computational Efficiency** – Flattening the images significantly **reduced the computational complexity** of distance-based models like KNN and tree-based models like Random Forest.
- **Effect on Accuracy** – Models that rely solely on raw pixel intensity (e.g., KNN and Logistic Regression) performed worse compared to CNN, which retained the 2D structure. However, PCA helped mitigate this by reducing redundancy in the input features.

IV. CONCLUSION

This study compared multiple machine learning models for handwritten digit recognition on MNIST, evaluating classification accuracy, computational efficiency, and interpretability. Our findings show that **CNN achieved the highest accuracy (93.16%)**, leveraging spatial hierarchies for superior performance. Despite its longer training time (93.82s), its inference speed (4.24s) makes it highly practical for real-world applications. **SVM performed well (92.10%) but remains computationally expensive**, requiring 155.85s for training and 52.52s for inference, making it impractical for large-scale use. Logistic Regression (92.21%) and Random

Forest (91.19%) provided a strong balance of accuracy and efficiency, with fast inference times (0.03s and 0.09s, respectively), making them ideal for real-time applications. **KNN was the least efficient**, struggling with low accuracy (86.78%) and high inference times, making it impractical for large datasets.

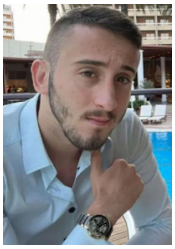
Misclassification patterns revealed that **digits 2, 5, and 8 were the hardest to classify**, while **digit 1 was the easiest** across all models. Feature extraction and dimensionality reduction significantly impacted performance: PCA improved efficiency for traditional models, but CNN benefited from retaining the 2D image structure.

Overall, CNN is the most suitable for this task, but simpler models like Logistic Regression and Random Forest remain competitive when computational efficiency is the priority. The choice of model should be based on the trade-offs between accuracy, efficiency, and interpretability. [2].

ACKNOWLEDGMENT

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

GIDI RABI is an undergraduate B.Sc. student in Computer Science at Ariel University, Israel. He developed and implemented machine learning scripts, conducted performance analyses, and contributed to model evaluation. His work included scripting model comparisons, optimizing computational efficiency, and researching relevant literature to support findings.



ROI BRUCHIM is an undergraduate B.Sc. student in Computer Science at Ariel University, Israel. He contributed to implementing classification models, optimizing hyperparameters, and analyzing misclassification patterns. Roi also documented findings, formulated research insights, and sourced academic references to contextualize results.

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.