A Comprehensive Analysis of CNN for Handwritten Digit Classification with Reference to SVM Using the MNIST Dataset

Authors: Nadav Vaknin

Keywords: MNIST, Support Vector Machine (SVM), Convolutional Neural Network (CNN), Image Classification, Machine Learning, Deep Learning, Hyperparameter Tuning, Data Distribution

Abstract: Handwritten digit classification is a fundamental task in machine learning and computer vision. The MNIST dataset has been widely used to benchmark the performance of various classification models. This study primarily investigates the performance of Convolutional Neural Networks (CNNs), examining the impact of hyperparameter selection, data distribution, and network architecture on classification accuracy. As a reference, we compare CNN with a traditional machine learning model, Support Vector Machine (SVM), to highlight the advantages of deep learning. The results demonstrate that CNN significantly outperforms SVM in accuracy and generalization, emphasizing the importance of deep learning techniques in image classification.

1. Introduction

Handwritten digit classification plays a crucial role in the fields of optical character recognition (OCR), automated form processing, and digital banking. The ability to accurately classify handwritten digits is essential for various real-world applications, such as postal code recognition in mail sorting and automatic check processing. The MNIST dataset, introduced by LeCun et al., is one of the most widely used benchmarks for evaluating the performance of machine learning algorithms in this domain. It consists of grayscale images of handwritten digits (0-9), standardized to a 28x28 pixel format.

Machine learning techniques have evolved significantly over the years, with early methods relying on hand-crafted features such as pixel intensity histograms and edge detection. Traditional classifiers, such as k-Nearest Neighbors (k-NN), Decision Trees, and Support Vector Machines (SVM), have shown reasonable success in digit classification. However, with the rise of deep learning, neural network-based approaches, particularly Convolutional Neural Networks (CNNs), have set new performance benchmarks by leveraging their ability to automatically learn hierarchical feature representations from raw image data.

This study focuses on investigating the properties of CNNs and their role in digit classification. The main objectives include:

1. Evaluating the effect of hyperparameter tuning, data distribution, and network architecture on CNN performance.
2. Comparing CNN with SVM as a reference machine learning model to understand their strengths and limitations.
3. Assessing the generalization ability of CNNs across different data splits and training configurations.

The remainder of this paper is structured as follows: Section 2 reviews related literature on CNN and SVM for digit classification. Section 3 outlines the methodology, including data preprocessing, model design, and training procedures. Section 4 presents the experimental results, followed by a discussion and conclusion in Section 5. Future research directions are explored in Section 6, and references are provided in Section 7.

2. Literature Review

The classification of handwritten digits has been a widely studied topic in machine learning and pattern recognition. Early approaches used feature extraction techniques such as Histogram of Oriented Gradients (HOG), Principal Component Analysis (PCA), and handcrafted filters to extract meaningful features from the MNIST dataset. Classical machine learning models like Support Vector Machines (SVM), k-Nearest Neighbors (k-NN), and Decision Trees demonstrated reasonable performance, with SVM often outperforming others due to its ability to handle high-dimensional data and find optimal decision boundaries.

With the advent of deep learning, researchers began to explore Convolutional Neural Networks (CNNs) as a more effective method for image classification. LeCun et al. (1998) pioneered CNN-based approaches for digit classification, showing that CNNs could automatically extract relevant features without requiring manual feature engineering. Studies have consistently demonstrated that CNNs outperform traditional models, often achieving accuracy rates above 99% on the MNIST dataset.

Comparative studies between traditional machine learning models and deep learning approaches have highlighted the trade-offs between accuracy, computational cost, and training time. While SVMs remain effective for small and moderate-sized datasets, CNNs excel in large-scale classification tasks due to their hierarchical feature extraction capabilities. Recent advancements in deep learning, such as hyperparameter optimization, transfer learning, and data augmentation, have further improved CNN performance, making them the preferred choice for image recognition tasks.

3. Methodology

Dataset

The MNIST dataset consists of 60,000 training images and 10,000 test images of handwritten digits (0-9). Each image is a 28x28 grayscale pixel representation of a digit. For convenience we chose to use only the training set which in turn allows us to determine the partition of the samples to train, validation and test sets in different ways.

Since the MNIST dataset is well known, we perform a quick EDA and data visualization to be more familiar with the data and the distribution of its properties (trough the different class and in each sample).

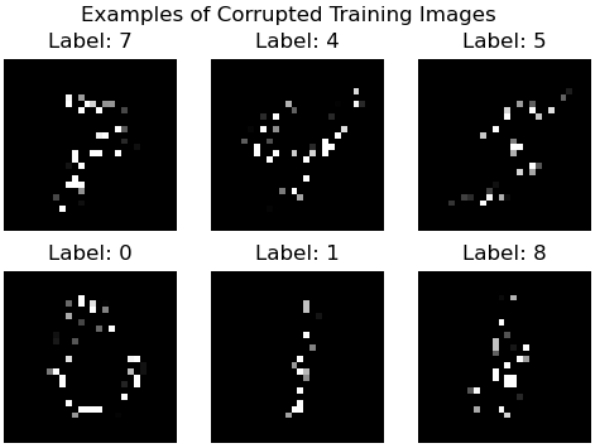
SVM Model

For the SVM model, the images were flattened into 784-dimensional feature vectors. We applied a radial basis function (RBF) kernel to classify the digits. The SVM model was trained using 3-fold cross-validation to find the optimal hyperparameters, including the regularization parameter (C) and kernel coefficient (γ). The model was optimized using grid search over a range of values for C and γ to ensure the best classification performance. Once the optimal parameters were identified, the final SVM model was trained on the full training set and evaluated on the test set.

CNN Model

First, we use a CNN with default hyperparameters which was designed as follows: two convolutional layers, each followed by max-pooling layers. The network was then flattened and passed through a fully connected layer with ReLU activation, followed by a softmax output layer. After documenting the basic default net performance, we vary the hyperparameters, including learning rate, batch size, and the number of epochs, to examine their impact. The model was trained using the Adam optimizer with categorical cross-entropy loss.

In the next step we alter the number of images in the train set to find a threshold of the minimal set size for which the performance of the net is dramatically decreased. Furthermore, we test the impact of deleting some of the data by converting a random number of pixels in each image to black.



Conversely, we tried to examine a few approaches that may encourage the performance of the model to increase - we started with training the model in its basic form and documenting all the misclassified samples the model predicted. Then we applied to the training set those samples multiple times and shuffled the data randomly, we trained the model again on this new data set and evaluated the accuracy. Also, we plot a histogram for each of the true labels (0-9 numbers) presenting the distribution of the predicted labels - as attempts to see if the model tends to predict some number as another (i.e. switching two similar numbers like 4 and 9 or 2 and 7).

In addition, we enlarged the training set 10 times bigger than the original one, applying the bootstrap approach, examine the performance of a model that trained on a much larger data. Another check we did is to alter the distribution of the training set and make it imbalanced - by centering the distribution of the numbers around one of them that picked randomly, in three different levels – train the model each time on the new bias data set and evaluate the accuracy of the test.

Finally, we examine a slightly different architecture of the CNN from the one we used through this all study, in the attempt to improve its performance, meanly by adding a batch normalization after each convolution layer and making the dense net a bit more deep whit different values of dropout after each layer. In addition, due to the fact that the convolutional layers are actually preforming a reduction in the dimensionality of the observations (i.e. creating a feature map in smaller dimensions than the input) we replaced those layers in PCA approach and trained a fully connected net on the principal component without any Conv layers.

4. **Results**

The performance of both the Support Vector Machine (SVM) and Convolutional Neural Network (CNN) models was evaluated using classification accuracy on the test dataset. The SVM model achieved a test accuracy of 97.88%, with hyperparameters set to C = 10 and γ = 0.01. However, the training time for the SVM model was significantly longer due to the high-dimensional feature space, making it computationally expensive. In contrast, CNNs efficiently leveraged GPU acceleration, significantly reducing training time while achieving higher accuracy.

The CNN model, using its default hyperparameter values, consistently achieved an accuracy of 99% on average (Fig. 1). The initial hyperparameter settings included a learning rate of 0.001, 10 epochs, and a batch size of 32 samples. However, further experimentation and fine-tuning of these parameters demonstrated that hyperparameter optimization had a noticeable impact on the CNN's performance. In particular, increasing the number of epochs to 20, while keeping the learning rate at 0.001 and maintaining a batch size of 32, resulted in the highest observed accuracy (Fig. 2a - 2c). This improvement suggests that a longer training duration allows the CNN model to learn more complex patterns, ultimately leading to better generalization and performance.

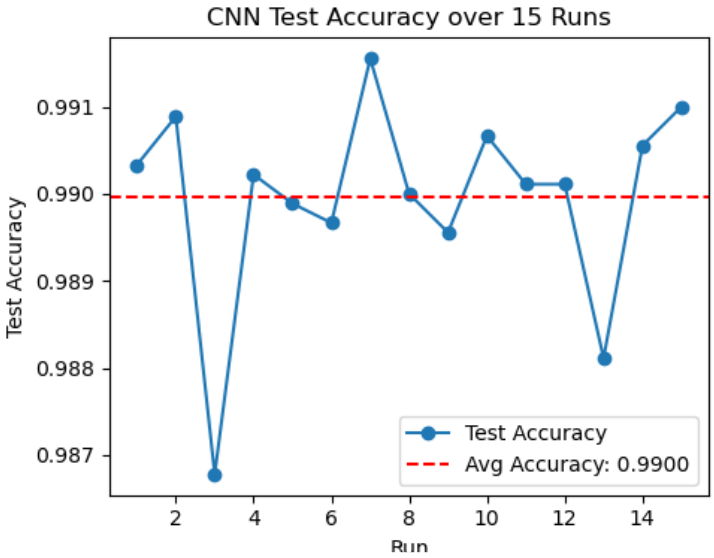


Fig1. The test accuracy on each iteration of the model training (in blue) and the average test accuracy (red).

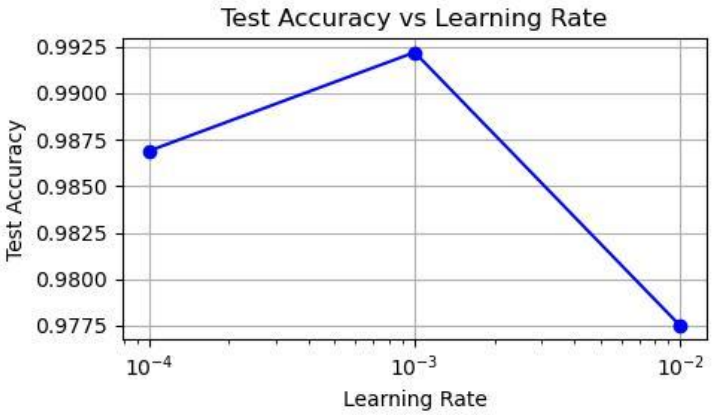


Fig 2a. Test accuracy in the different learning rates

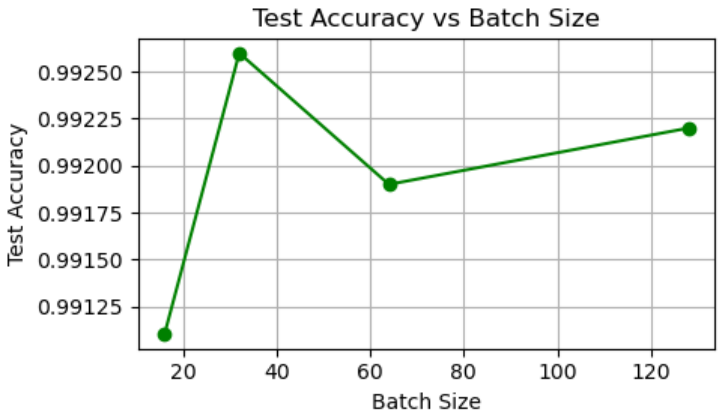


Fig 2b. Test accuracy in four different batch size

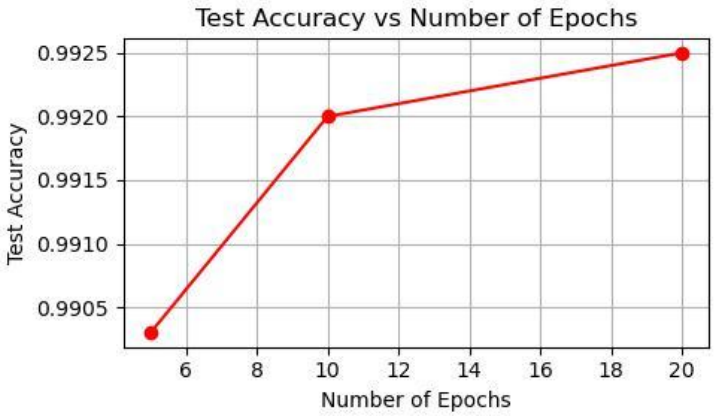


Fig 2c. Test accuracy in different number of epochs

In the second part of this work, we tried to examine the effect of the data available to the model during training. It turns out that, for this specific architecture, a huge portion of the data needs to be unavailable or corrupted to cause a dramatic change in the model's performance. Even when we used less than 1% of the data, the model still converged and achieved slightly over 80% accuracy.

Surprisingly, when we changed 80% of the pixels in each image to black - making it impossible for a human to recognize the original number - the CNN still managed to converge and reached more than 80% accuracy on the test set.

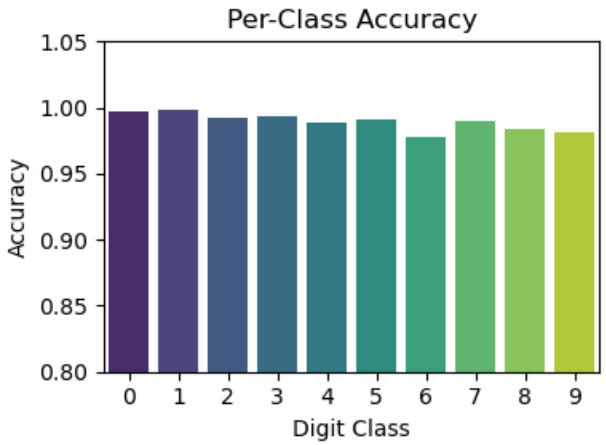


Fig. 3b Per-class accuracy test

On the other hand, using the bootstrap approach to create a much larger training set (specifically 10 times larger) led to only a slight increase in test accuracy.

Although the model appears to switch between specific numbers (Fig. 3a), the per-class accuracy test and the ROC curve for each number indicate that the model does not favor predicting one number over the others (Fig. 3b, Fig. 3c).

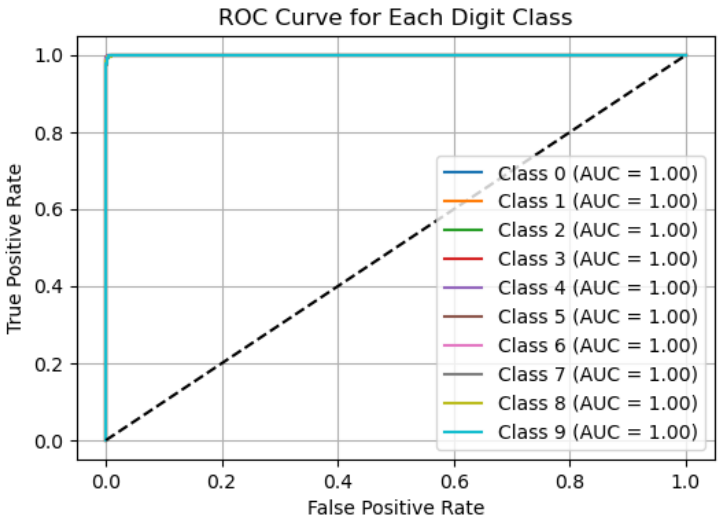


Fig. 3c The ROC curve per each number

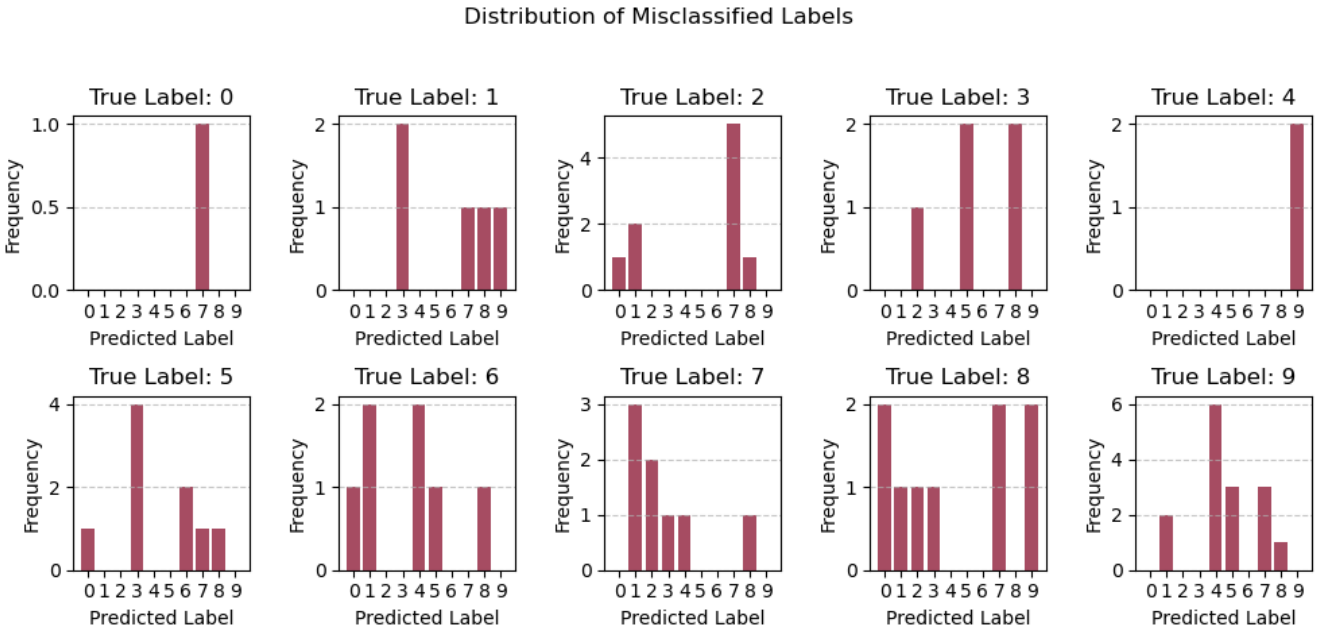


Fig. 3a The distribution of misclassified labels for each of the digits. The histogram shows that the model tends to misclassify two specific digits

The changes we introduced in the distribution of the training set - centering the samples around the number 5 at three different levels of bias - did not significantly impact the model's test accuracy. Across all three levels, accuracy remained between 96% and 98%, compared to 99% on the unbiased dataset. (Fig.4)

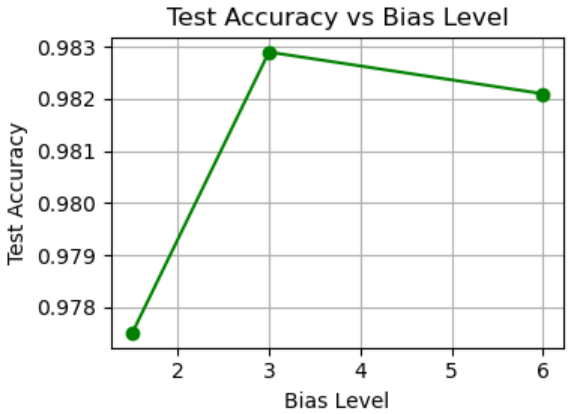


Fig.4 Test accuracy in three bias levels

Attempts to improve the network's performance through architectural modifications also did not yield significant gains. Adding Batch Normalization, Global Average Pooling, Learning Rate Scheduler, LeakyReLU, or an additional dense layer did not lead to any noticeable improvement in test accuracy.

Finally, the attempt to use a dimension reduction method on a CNN can be misleading since the convolutional layer essentially dose exactly that – crating a feature map in a lower-dimensional space. Thus, the accuracy rate on such approach – using PCA whit CNN was significantly lower. Conversely, when we use PCA and three dense layers, skipping any performance of convolutional, the accuracy almost reached to those of the CNN, whit 96.99%.

5. Discussion and Conclusions

The comparative study illustrates that while SVM provides decent performance for digit classification, CNN significantly outperforms it in terms of accuracy and especially in computational efficiency. The key advantage of CNN lies in its ability to learn hierarchical spatial features directly from the data, eliminating the need for manual feature engineering. This unique property not only explains CNN's overall superiority over traditional methods but also its remarkable ability to learn from highly corrupted data, even when recognition seems impossible to a human observer. Additionally, CNN performance is highly dependent on hyperparameter selection, relative to the specific network architecture, where careful optimization essential for achieving the best results.

6. Future Work

Future research on much larger datasets could be explored, as some studies showed this approach may improve performance. Additionally, testing these models on more complex handwritten datasets, such as EMNIST or real-world digit recognition tasks, could provide further insights. A more extensive comparison between PCA with dense layers and CNN-based approaches could be conducted to determine whether it is possible to achieve comparable

accuracy without convolutional layers. Investigating more advanced CNN architectures, such as ResNet and Vision Transformers, may also provide further improvements in accuracy and generalization ability.

7. References

[1] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-Based Learning Applied to Document Recognition. Proceedings of the IEEE. [2] Cortes, C., & Vapnik, V. (1995). Support-Vector Networks. Machine Learning. [3] Krizhevsky, A., Sutskever, I., & Hinton, G. (2012). ImageNet Classification with Deep Convolutional Neural Networks. Advances in Neural Information Processing Systems (NIPS).

Appendix

From the left corner above: another 15 iteration of the basic CNN model and its accuracy in each run. The three levels of bias distribution of the data, centered around the number five