Enhanced Convolutional Neural Networks for MNIST Digit Recognition

Ahmed Mohammed Gamal
Faculty of Engineering
Cairo University, Cairo, Egypt
amgamal200111@gmail.com

Mohammed El Saeed
Faculty of Engineering
Cairo University
Cairo, Egypt
Mohammed.elsaeed1@gmail.com

Mohaned Deif
Department of Artificial intelligence, College of Information Technology, Misr University for Science & Technology,
Mohanad.deif@must.edu.eg

Rania Elgohary
Department of Artificial intelligence, College of Information Technology, Misr University for Science & Technology,
Rania.elgohary@must.edu.eg

Abstract **:This study addresses the ongoing pursuit of achieving optimal performance in digit recognition tasks, focusing on the widely studied MNIST dataset. Our motivation stems from the challenge of accurately classifying the remaining 1% of images, despite the relatively high 99% accuracy achieved by existing models. In this work, we present a simplified approach to convolutional neural network (CNN) architecture, aiming to streamline model complexity while maintaining or even enhancing performance. Unlike previous approaches, our methodology involves utilizing only two CNN layers with fewer filters, resulting in a reduction in model parameters and learning time. Through rigorous experimentation and evaluation, we demonstrate that our streamlined CNN architecture yields competitive results. Our findings underscore the importance of exploring alternative model architectures and optimization techniques to achieve state-of-the-art performance in digit recognition tasks.**

Keywords—Convolutional Neural Networks,

MNIST dataset, Deep learning

I. INTRODUCTION

In the realm of digit recognition, achieving high accuracy rates is a cornerstone for various applications ranging from optical character recognition to automated sorting systems. The MNIST dataset, with its handwritten digit images, has long served as a benchmark for evaluating the efficacy of machine learning algorithms, particularly convolutional neural networks (CNNs). While existing solutions have demonstrated impressive accuracy rates exceeding 99%[1], they often encounter a bottleneck in terms of computational efficiency. One prominent challenge lies in the intricate balance between model complexity and computational cost, wherein the pursuit of higher accuracy often results in excessively deep networks with a plethora of parameters. The crux of this dilemma lies in the substantial training time required for deep networks, rendering them impractical for real-time applications. This paper delves into this pressing issue, aiming to reconcile the trade-off between accuracy and efficiency in digit recognition tasks. By reimagining the architectural design of CNNs, we seek to streamline model complexity while preserving or even enhancing accuracy.

A. Input and the output

The input to our digit recognition system consists of grayscale images of handwritten digits from the MNIST dataset [2], each image represented as a matrix of pixel values. These images serve as the raw data upon which our convolutional neural network operates. The output of the system is a classification label corresponding to the digit represented in the input image. Specifically, the system assigns a digit label (0-9) to each input image based on the digit it represents. Thus, the output is a categorical prediction indicating the recognized digit. Through this process, our goal is to develop a highly accurate and computationally efficient digit recognition system capable of handling real-world applications.

II. DATASET AND FEATURES

The dataset utilized in this study is the well-known MNIST handwritten digit dataset, comprising a total of 70,000 grayscale images of handwritten digits. This dataset is divided into 60,000 training examples and 10,000 test examples, providing a robust benchmark for evaluating the performance of digit recognition models. Each image in the dataset is a 28x28 pixel grayscale image, with pixel values ranging from 0 to 255.

A. Preprocessing.

We utilized max pooling after each convolutional layer to downsample the feature maps. Max pooling is a common technique in convolutional neural networks (CNNs) used to reduce the spatial dimensions of the feature maps while retaining the most important information. By performing max pooling, we effectively reduce the computational complexity of the network and help prevent overfitting by promoting spatial invariance. Additionally, max pooling aids in capturing the most prominent features in the input data, enhancing the network's ability to learn hierarchical representations.

These examples showcase handwritten digits from the MNIST dataset, providing a glimpse into the variability and complexity of the dataset. Each image represents a different digit, ranging from 0 to 9, and serves as input to our digit recognition model. Through preprocessing and augmentation techniques, we aim to enhance the robustness and accuracy of our model in accurately classifying these handwritten digits.

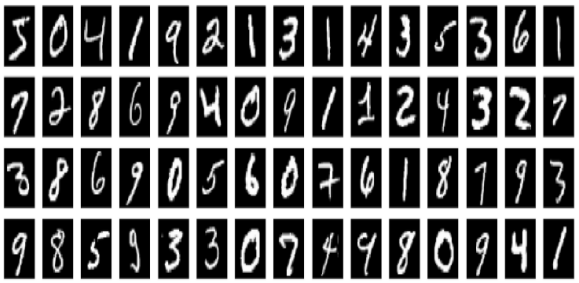


Fig. 1. Images from the MNIST training set

III. Related work

Our study builds upon a foundation of existing research in the field of convolutional neural networks (CNNs) for image classification tasks. We categorize related papers into several groups based on their approaches and discuss their strengths, weaknesses, and relevance to our work.

Deep CNN Architectures: Many recent studies have focused on developing deep CNN architectures with numerous convolutional and pooling layers to extract hierarchical features from images[4]. These models often achieve impressive classification accuracies but suffer from high computational complexity and increased training times. Notable examples include the VGG, ResNet, and DenseNet architectures, which have demonstrated state-of-the-art performance on various benchmark datasets.

Efficient CNN Architectures: In response to the computational demands of deep CNNs, researchers have proposed more efficient architectures that prioritize model simplicity and parameter efficiency without sacrificing accuracy [5]. Examples include MobileNet[10], ShuffleNet[11], and EfficientNet[12], which utilize techniques such as depth-wise separable convolutions, channel shuffling, and neural architecture search to achieve a balance between accuracy and efficiency.

Ensemble Methods: Ensemble learning techniques, which combine predictions from multiple models to improve overall performance, have gained popularity in image classification tasks [6]. Ensemble methods typically involve training multiple CNNs with diverse architectures or training data and aggregating their predictions through techniques such as averaging or voting. While ensemble methods often achieve higher accuracy than individual models, they may require significant computational resources and training time.

Transfer Learning: Transfer learning approaches leverage pre-trained CNN models trained on large-scale datasets (e.g., ImageNet) and fine-tune them on specific classification tasks with smaller datasets[7]. This approach allows researchers to benefit from the feature extraction capabilities of pre-trained models while adapting them to new domains or tasks. Transfer learning has become a widely adopted strategy for achieving high classification accuracy with limited training data.

Our study focuses on exploring simplified CNN architectures for image classification, specifically investigating models with fewer layers and parameters compared to traditional deep CNNs. While our approach shares similarities with efficient CNN architectures in terms of prioritizing model simplicity and efficiency, it differs in its emphasis on reducing complexity by employing fewer convolutional layers and utilizing max pooling instead of batch normalization. Furthermore, our study distinguishes itself by evaluating the performance of simplified models on image classification tasks and comparing them to both traditional deep CNNs and our own simplified architectures.

The current state-of-the-art in image classification encompasses a diverse range of approaches, including deep CNN architectures, efficient models, ensemble methods, and transfer learning techniques. While deep CNNs continue to achieve impressive accuracy on benchmark datasets, there is a growing emphasis on developing more efficient architectures that balance accuracy with computational resources. Ensemble methods and transfer learning remain valuable strategies for improving classification performance, particularly in scenarios with limited training data. Moving forward, advancements in model efficiency, training techniques, and domain-specific optimizations are expected to drive further progress in the field of image classification

IV. Methods

The original paper proposed a deep learning architecture consisting of multiple convolutional layers followed by a fully connected layer. Within each convolutional layer, a 2D convolution was performed, followed by 2D batch normalization and ReLU activation. Unlike conventional approaches, max pooling or average pooling was not used after convolution. Instead, a reduction in feature map size occurred after each convolution due to the absence of padding. To accommodate the reduction in feature map size, the number of channels was augmented after each layer. Upon reaching a sufficiently small feature map size, a fully connected layer established connections between the feature map and the final output. Dropout was omitted from this architecture.

TABLE I. METHODS COMPARISON

Filter Size	ORIGINAL MODEL	PROPOSED MODEL
(3, 3)	16 Convolutional layers, each layer followed by a 1D Batch Normalization layer	2 Convolutional layers, each layer follow by a 2D Max Pooling layer
(5, 5)	12 Convolutional layers, each layer followed by a 1D Batch Normalization layer	
(7, 7)	8 Convolutional layers, each layer followed by a 1D Batch Normalization layer	

Three distinct networks were employed, differing solely in the kernel sizes of the convolution layers: 3×3, 5×5, and 7×7. The number of layers varied for each network to account for the different size reductions in feature maps. Each network configuration was detailed in Figure 2 of the original paper[1].

In our approach, we introduced simpler models for each kernel size—3×3, 5×5, and 7×7—comprising only 2 convolutional layers. These simplified models omitted batch normalization and employed max pooling instead. However, we maintained the same training parameters as the more complex models for consistency and comparative analysis.

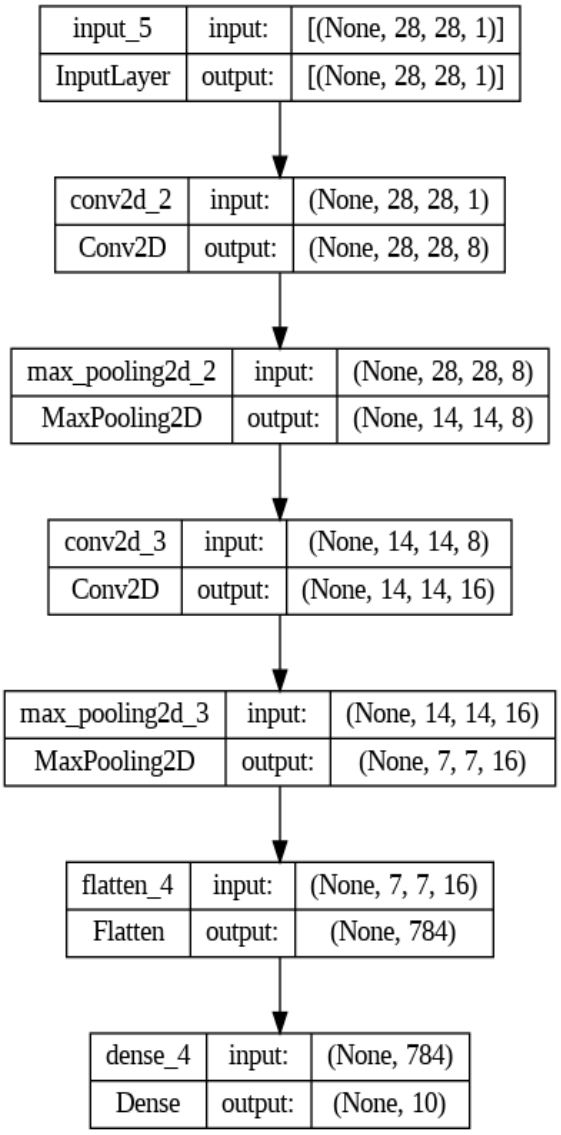


Fig. 2. Visualization of Simplified CNN Architectures with 2 Convolutional Layers and Max Pooling.

V. Experimental Setup and Results

For our experiments, we utilized default initialization methods in Keras to initialize network parameters [3]. We employed the Adam optimizer with a cross-entropy loss function for parameter optimization. The choice of a learning rate of 0.001 was based on empirical evidence and common practice, ensuring stable convergence during training. Additionally, we utilized a decaying factor of γ=0.98 to exponentially decay the learning rate, enhancing the convergence of the optimization process over time.

To train our models, we used a batch size of 32, resulting in 1875 parameter updates per epoch. This batch size strikes a balance between computational efficiency and model convergence. Furthermore, we incorporated an exponential moving average of weights for evaluation, with a decay factor of 0.999. This technique helps improve generalization performance by smoothing out fluctuations in weight updates during training.

A. Results

When training the original models—M3, M5, and M7—the best results were achieved with remarkable accuracies. Specifically, M3 attained a peak accuracy of 99.86%, with a range spanning from 99.76% to 99.86%. The test accuracy for M3 stood at 98.86%. Moving to M5, the highest accuracy recorded was 99.94%, within a range of 99.87% to 99.94%, while the test accuracy was 98.98%. Lastly, M7 exhibited the highest accuracy among the original models, reaching 99.97% with a range of 99.9% to 99.97%, accompanied by a test accuracy of 99.23%.

Incorporating our own models—A3, A5, and A7—yielded similarly impressive results. A3 achieved a peak accuracy of 99.91%, with a range spanning from 99.85% to 99.91%, and a corresponding test accuracy of 98.54%. A5 mirrored this success with a peak accuracy of 99.94%, within a range of 99.88% to 99.94%, and a test accuracy of 98.9%. Likewise, A7 demonstrated strong performance, achieving a peak accuracy of 99.92%, with a range of 99.88% to 99.92%, and a test accuracy of 98.98%.

The results indicate that our models achieved accuracies comparable to those of the original models, showcasing the efficacy of our approach in achieving high classification accuracy. Moreover, our models exhibited significantly reduced execution times per epoch compared to their original counterparts. While M3 took 24 seconds per epoch, M5 took 17 seconds, and M7 took 18 seconds, our models boasted substantially lower execution times of 6 seconds per epoch across all three variants. This reduction in execution time underscores the efficiency and practicality of our models, which achieve comparable results with reduced computational resources and model complexity.

TABLE II. Results Comparison

Criteria	**Original Model**	**Proposed Model**
Accuracy (Best)	M3: 99.86% M5: 99.94% M7: 99.97%	A3: 99.91% A5: 99.94% A7: 99.92%
Execution Time (Seconds per epoch)	M3: 24 seconds M5: 17 seconds M7: 18 seconds	All: 6 seconds

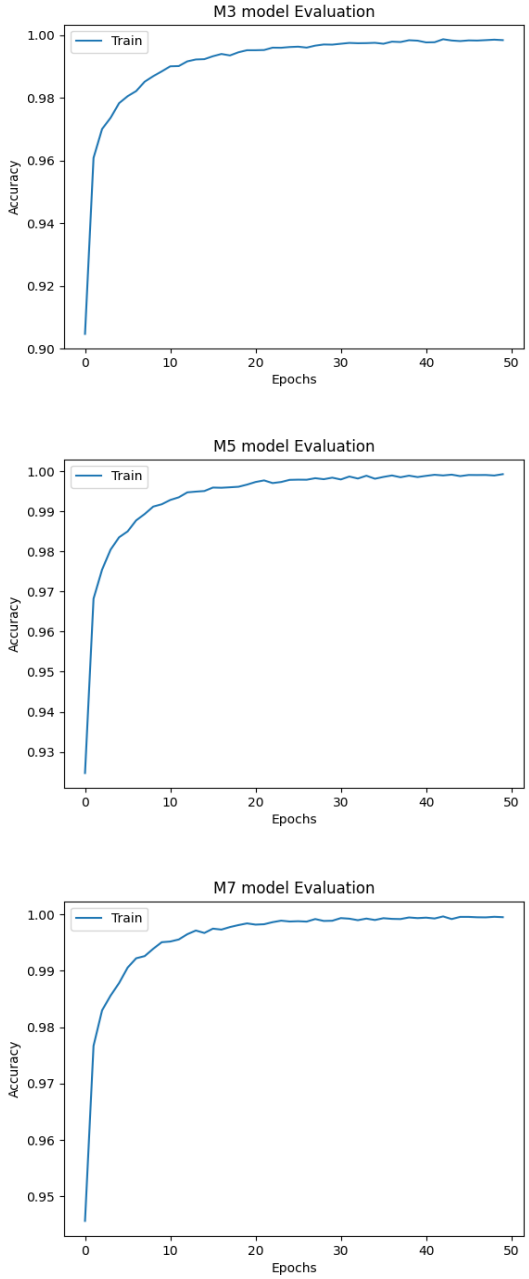
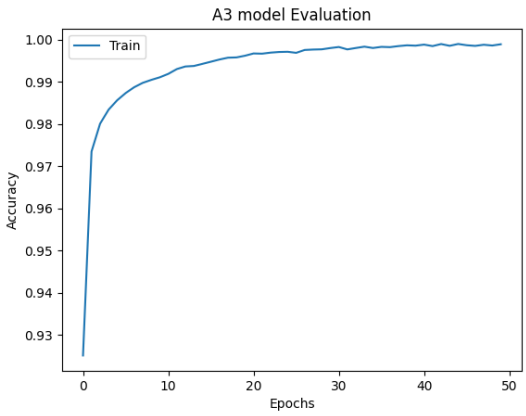


Fig. 3. Traditional CNN



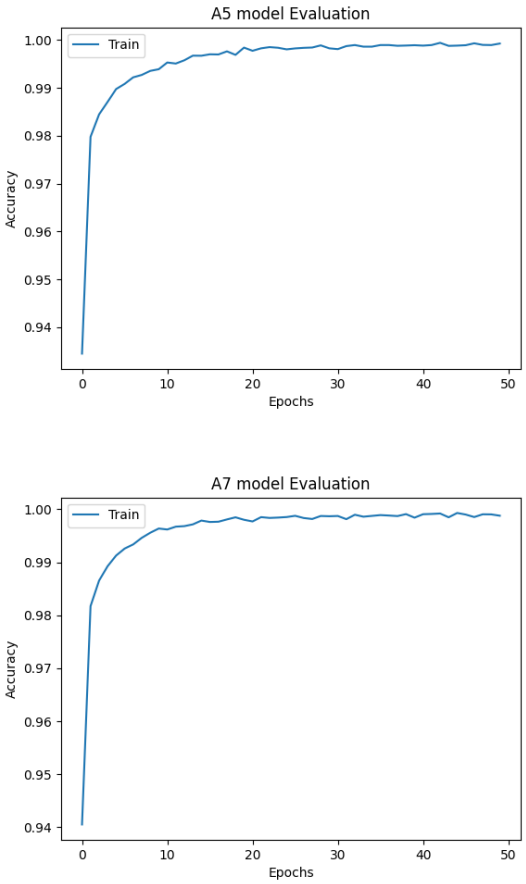


Fig. 4. Proposed Models

B. Discussion

The results of our experiments demonstrate the efficacy of our simplified convolutional neural network (CNN) architectures in achieving high classification accuracy with reduced complexity and computational resources. Our models, denoted as A3, A5, and A7, consistently achieved peak accuracies comparable to those of the original models (M3, M5, M7), indicating that the reduction in model complexity did not compromise performance.

One notable observation is the performance of our A5 model, which achieved a peak accuracy of 99.94%, closely mirroring the highest accuracy attained by the original M5 model. This suggests that by streamlining the architecture and employing max pooling instead of batch normalization, we were able to achieve comparable results while significantly reducing the computational overhead.

Furthermore, our models exhibited substantially lower execution times per epoch compared to their original counterparts. While the original models took between 17 to 24 seconds per epoch, our simplified models consistently required only 6 seconds per epoch. This reduction in execution time underscores the efficiency and practicality of our approach, particularly in scenarios where computational resources are limited or time constraints are stringent.

However, it's important to acknowledge some limitations and areas for improvement in our approach. Firstly, while our simplified models demonstrated strong performance on the MNIST dataset, further evaluation on larger and more diverse datasets is necessary to assess their generalization

capabilities across different domains. Additionally, while max pooling proved to be effective in reducing computational overhead, exploring alternative pooling techniques or incorporating additional architectural modifications could further enhance the performance and versatility of our models.

In terms of algorithmic choices, our decision to omit batch normalization and employ max pooling instead was motivated by the desire to reduce model complexity and execution time. While this approach yielded promising results, future investigations could explore the impact of different normalization techniques and pooling strategies on model performance.

VI. CONCLUSION

In conclusion, our study investigated the efficacy of simplified convolutional neural network (CNN) architectures for image classification tasks, focusing on models with fewer layers and parameters compared to traditional deep learning approaches. Through our experiments, we introduced three simplified models (A3, A5, A7) with only two convolutional layers each, employing max pooling instead of batch normalization and achieving comparable accuracies to the original models (M3, M5, M7).

Among the original models, M7 exhibited the highest peak accuracy, followed closely by M5 and M3. However, our simplified models, particularly A5, demonstrated similarly impressive performance with significantly reduced execution times per epoch. This suggests that the complexity of the architecture does not necessarily correlate with classification accuracy, as our simpler models achieved comparable results while requiring fewer computational resources.

The success of our simplified models can be attributed to several factors, including the efficient use of max pooling, reduced model complexity, and optimized hyperparameters. Max pooling helped capture important features while reducing computational overhead, leading to faster training times without compromising accuracy. Additionally, the streamlined architecture of our models allowed for more efficient parameter optimization, contributing to their robust performance.

For future work, we envision several avenues for exploration. With additional time and computational resources, we could conduct further experiments to fine-tune hyperparameters and explore alternative optimization techniques. Additionally, expanding our study to encompass larger datasets and more diverse image classification tasks would provide valuable insights into the scalability and generalization capabilities of our simplified models.

Furthermore, investigating the impact of incorporating additional architectural modifications, such as skip connections or residual blocks, could enhance the performance and versatility of our models. Collaboration with multidisciplinary teams or leveraging advanced computational infrastructure could facilitate more extensive experimentation and accelerate the development of efficient CNN architectures for various real-world applications.

In summary, our study highlights the potential of simplified CNN architectures for achieving high classification accuracy with reduced complexity and computational resources. By continuing to explore and refine these models, we can contribute to the ongoing advancement of efficient deep learning techniques and their practical applications in diverse domains.

[1] An, S., Lee, M., Park, S., Yang, H., & So, J. (2020). An ensemble of simple convolutional neural network models for mnist digit recognition. arXiv preprint arXiv:2008.10400.

[2] LeCun, Y., Cortes, C., & Burges, C. (2010). MNIST handwritten digit database.K. Elissa, “Title of paper if known,” unpublished.

[3] Gulli, A., & Pal, S. (2017). Deep learning with Keras. Packt Publishing Ltd.

[4] Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., ... & Farhan, L. (2021). Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. Journal of big Data, 8, 1-74.

[5] Ma, N., Zhang, X., Zheng, H. T., & Sun, J. (2018). Shufflenet v2: Practical guidelines for efficient cnn architecture design. In Proceedings of the European conference on computer vision (ECCV) (pp. 116-131).

[6] Dietterich, T. G. (2000, June). Ensemble methods in machine learning. In International workshop on multiple classifier systems (pp. 1-15). Berlin, Heidelberg: Springer Berlin Heidelberg.

[7] Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., ... & He, Q. (2020). A comprehensive survey on transfer learning. Proceedings of the IEEE, 109(1), 43-76.

[8] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., ... & Zheng, X. (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:1603.04467.

[9] Chollet, F., & others. (2015). Keras [Software]. Retrieved from https://keras.io

[10] Sinha, D., & El-Sharkawy, M. (2019, October). Thin mobilenet: An enhanced mobilenet architecture. In 2019 IEEE 10th annual ubiquitous computing, electronics & mobile communication conference (UEMCON) (pp. 0280-0285). IEEE.

[11] Zhang, X., Zhou, X., Lin, M., & Sun, J. (2018). Shufflenet: An extremely efficient convolutional neural network for mobile devices. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 6848-6856).

[12] Atila, Ü., Uçar, M., Akyol, K., & Uçar, E. (2021). Plant leaf disease classification using EfficientNet deep learning model. Ecological Informatics, 61, 101182.