An Efficient Deep Learning Model for Bangla Handwritten Digit Recognition

1st A K M Nihalul Kabir
Department of Computer Science and Engineering
BRAC University
Dhaka, Bangladesh
akm.nihalul.kabir@g.bracu.ac.bd

2nd MD Shahadat Hossain Shamim
Department of Computer Science and Engineering
BRAC University
Dhaka, Bangladesh
shahadat.hossain.shamim@g.bracu.ac.bd

3rd Md. Rumman Shahriar
Department of Computer Science and Engineering
BRAC University
Dhaka, Bangladesh
md.rumman.shahriar@g.bracu.ac.bd

4th Md. Asif Hasan Biplob
Department of Computer Science and Engineering
BRAC University
Dhaka, Bangladesh
asif.hasan.biplob@g.bracu.ac.bd

***Abstract*—This paper presents an efficient, high-performance deep learning model for the recognition of handwritten Bangla digits. Recognizing handwritten digits in Bengali, the world's fifth most spoken language, is a critical task for digitization and automation, yet it poses significant challenges due to high calligraphic variability. Current solutions often require a trade-off between recognition accuracy and computational efficiency, a major hurdle for practical deployment. We address this challenge by proposing a custom-designed Convolutional Neural Network (CNN), termed EfficientCNN, specifically optimized for the 32x32 pixel images of the BHaND dataset. A key novelty of our architecture is the use of a Global Average Pooling layer instead of a traditional Flatten layer, which significantly reduces the model's parameter count without compromising its feature-learning capacity. The proposed model achieves a test accuracy of 99.78%. A comprehensive benchmark analysis demonstrates that our EfficientCNN outperforms established architectures, including AlexNet, LeNet-5, and MobileNetV2-0.5, offering a superior balance of accuracy, model size, and inference speed. This combination of state-of-the-art accuracy and computational efficiency underscores the model's suitability for deployment in real-world, resource-aware applications.**

***Index Terms*—Bangla, handwritten digit recognition, convolutional neural network, deep learning, BHAND dataset, computational efficiency.**

I. Introduction

The automatic recognition of handwritten text is a cornerstone of modern Optical Character Recognition (OCR) systems, enabling the seamless integration of physical documents into digital workflows. This technology is indispensable for automating data entry, digitizing historical archives, and streamlining financial services. For the Bengali language, which is spoken by over 265 million people worldwide, the development of accurate and efficient OCR is of paramount importance. However, the intrinsic characteristics of the handwritten Bangla script, including its complex digits, diverse writing styles, cursive connections, and structural similarities between certain numerals, present formidable challenges to recognition systems.

Early machine learning approaches often fell short in this domain, struggling to generalize across the vast range of human handwriting. The advent of deep learning, particularly Convolutional Neural Networks (CNNs), has led to significant breakthroughs. While these models have pushed the boundaries of accuracy, many state-of-the-art architectures are computationally intensive, limiting their practical application in resource-constrained environments like mobile devices or real-time processing systems.

This research aims to bridge this gap by developing a deep learning model that is both highly accurate and computationally efficient. The primary objective is to design, implement, and validate a novel CNN architecture tailored specifically for the task of Bangla handwritten digit recognition on the standardized BHaND dataset.

The main contributions of this paper are fourfold:

- A Novel and Efficient Architecture: We propose EfficientCNN, a custom CNN architecture featuring parameter-efficient design choices like Global Average Pooling, specifically optimized for the 32x32 BHaND dataset images.
- State-of-the-Art Accuracy: Our model achieves a test accuracy of 99.78%, demonstrating top-tier performance on a challenging and widely used benchmark.
- Comprehensive Benchmarking: We provide a rigorous comparative analysis of EfficientCNN against seminal and efficient architectures—AlexNet, LeNet-5, and MobileNetV2—across multiple performance metrics including accuracy, parameter count, inference latency, and model size.
- Demonstrated Robustness: We quantitatively prove the model's excellent generalization and lack of overfitting using the percentage-point gap (Δpp) metric, confirming its reliability.
- This paper is structured as follows: Section II provides a review of the relevant literature. Section III details

the dataset and our proposed methodology, including the model architecture and training protocol. Section IV describes the experimental setup and implementation. Section V presents and discusses the experimental results. Finally, Section VI concludes the paper and suggests directions for future work.

II. Literature Review

The quest for accurate and efficient handwritten-digits recognition is a long-standing challenge in pattern recognition and computer vision. For the Bengali language, progress has been rapid, moving from classical machine-learning methodologies to the sophisticated deep-learning paradigms that now dominate. This review situates our work within that trajectory by examining (i) the shift from traditional methods to deep learning, (ii) the evolution of CNN architectures, (iii) the foundational role of standardized datasets, and (iv) recent advanced techniques and practical considerations.

A. From Traditional Methods to Deep Learning

Early Bangla-digit recognition systems relied on traditional machine-learning pipelines [3], [11]. These frameworks required extensive pre-processing and hand-crafted feature extraction—e.g., Histogram of Oriented Gradients (HOG) or Local Binary Patterns (LBP) [12]—followed by classical classifiers such as SVMs or k-NN [17]. Such approaches were limited by their dependence on manually engineered features, which struggled to capture the high variability and non-linear patterns inherent in handwritten Bangla numerals [2], [11]. Diverse writing styles, cursive connections, and structural ambiguities proved especially difficult for rigid feature-based systems [3]. Convolutional Neural Networks (CNNs) revolutionized the field [15], [18]. By learning hierarchical representations directly from raw pixels, CNNs eliminated manual feature design: early layers detect simple edges and curves, while deeper layers capture complex digit structures [12]. This end-to-end learning consistently outperformed traditional methods across datasets and metrics [1], [5], [6], establishing deep learning as the de facto standard for character recognition.

B. Evolution of CNN Architectures for Bangla Digits

Architecture design quickly became a research focus. Bayanno-Net (2019) [4] demonstrated that even a comparatively simple CNN with three convolutional and three max-pooling layers, plus dropout regularization [23], could achieve 96.5% accuracy. Alom et al. [5] advanced performance by augmenting CNNs with Gaussian and Gabor filters; the Gabor-enhanced variant reached 98.78% accuracy on CMATERdb 3.1.1. Gabor filters capture texture and edge information at multiple scales and orientations, complementing CNN feature learning. More recently, Sikder et al. [7] proposed an optimized, task-specific CNN that delivered 99.44% validation accuracy on the BHaND dataset. Their bespoke architecture outperformed generic models such as AlexNet [15] and Inception V3, underscoring a key theme of our work: carefully tailored architectures can surpass off-the-shelf networks for specialized tasks [1], [7].

C. The Foundational Role of Standardized Datasets

Rapid progress has depended on large, standardized datasets, which (i) supply the data volume deep networks require and (ii) enable fair benchmarking [2], [11], [17]. Major resources include:

CMATERdb – the foundational benchmark for early CNN studies [5], [25].

NumtaDB – a large-scale collection regarded as the Bangla analogue of MNIST [13], [21]; essential for training generalizable models [24].

ISI dataset – frequently used in cross-dataset comparisons to test robustness.

BHaND – the dataset for our work [13]; its scale and quality have enabled recent state-of-the-art results, including Sikder et al.'s 99.44% accuracy [7].

The availability of multiple datasets has encouraged cross-dataset validation, where models trained on one corpus are tested on others [11]. Such protocols better demonstrate generalization to real-world data drawn from varied sources.

D. Advanced Techniques and Practical Considerations

Beyond architecture design, researchers have embraced advanced techniques. Sikder et al.'s semi-supervised GAN (SGAN) generates realistic synthetic digits to address data imbalance and scarcity [7].

Concurrently, practical deployment demands computational efficiency. Recent studies focus on "lightning-fast" models for mobile and embedded devices [8]. This trend balances accuracy with constraints on parameter count, memory footprint [19], and inference speed. Designing parameter-efficient networks and applying optimization techniques ensure that high-accuracy models can serve real-world applications for the 265 million Bengali speakers [2]. Our research positions itself squarely at this intersection of performance and efficiency.

III. METHODOLOGY

A. BHaND Dataset

The BHaND (Bengali Handwritten Numerals Dataset) is the cornerstone of this study. It was explicitly created to be analogous to the widely-used MNIST dataset for English digits.

- Composition: The dataset comprises a total of 70,000 image samples. These are partitioned into a training set of 50,000 samples, a validation set of 10,000 samples, and a test set of 10,000 samples.
- Image Properties: Every image is a 32x32 pixel grayscale image. The pixel values are normalized to an intensity range of [0,1]. A key preprocessing step applied during the dataset's creation was the inversion of pixel intensities, so that black pixels (the digit strokes) have a value of 1 and white pixels (the background) have a value of 0.
- Data Format: In its raw form, each sample is a row containing two components: a 1024-dimensional flattened image vector (since 32x32=1024) and a corresponding

integer label from 0 to 9. For our CNN, these vectors were reshaped to a 32x32x1 tensor format. The entire dataset is stored in a single pickle file compressed with GZIP for efficient transmission and loading.

B. Proposed EfficientCNN Architecture

The EfficientCNN is a sequential CNN model custom-designed to maximize accuracy while maintaining computational efficiency for the specific input dimensions of the BHaND dataset. The architecture is detailed below.

- Input Layer: The model accepts tensors of shape (32, 32, 1).
- Convolutional Blocks: The core of the network consists of four sequential convolutional blocks. Each block is designed to learn progressively more complex features.
  - **Block 1 and 2:** Use filters of size 32 and 64, respectively. Each block contains two Conv2D layers (3x3 kernel, 'relu' activation, 'same' padding), with a BatchNormalization layer following each convolution. The block concludes with a MaxPooling2D (2x2) layer and a Dropout layer with a rate of 0.25.
  - **Block 3 and 4:** Increase the filter depth to 128 and 256, respectively, following the same pattern of Conv2D, BatchNormalization, MaxPooling, and Dropout. This hierarchical increase in filter depth allows the model to learn simple features like edges and curves in the early layers and more abstract shapes and digit components in the deeper layers.
- Global Average Pooling Layer: After the final convolutional block, a `GlobalAveragePooling2D()` layer is applied. Instead of flattening the entire multi-dimensional feature map, this layer computes the average of each feature map, outputting a single value per map. This reduces the number of parameters, mitigates overfitting, and increases robustness to spatial translations.
- Fully Connected Classifier Head: The output of the pooling layer is fed into two dense layers.
  - A Dense layer with 512 units and 'relu' activation, followed by BatchNormalization and a Dropout rate of 0.5.
  - A second Dense layer with 256 units and 'relu' activation, also followed by BatchNormalization and Dropout(0.5). The aggressive dropout rate in these dense layers provides strong regularization.
- Output Layer: A final Dense layer with 10 units (one for each digit class) and a softmax activation function to produce a probability distribution for classification. The total number of trainable parameters is 1,439,658, with a total parameter count of 1,442,154.

C. Training Protocol and Augmentation

- Optimizer and Loss Function: The model was compiled using the Adam optimizer with an initial learning rate of 0.001 and sparse_categorical_crossentropy as the loss function, which is suitable for integer-based, multi-class classification problems.
- Data Augmentation: To expose the model to a wider variety of handwriting styles and prevent overfitting on the training set, we used Keras's ImageDataGenerator for real-time data augmentation. The specific transformations included:
  - rotation_range=10
  - width_shift_range=0.1
  - height_shift_range=0.1
  - shear_range=0.1
  - zoom_range=0.1
- Training Callbacks: The training loop was governed by a list of callbacks designed to optimize the training process and save the best results:
  - ModelCheckpoint: This callback monitored the validation accuracy (val_accuracy) and saved the entire model only when an improvement was detected, ensuring that the final saved model corresponds to the best-performing epoch.
  - ReduceLROnPlateau: This monitored the validation loss (val_loss) and reduced the learning rate by a factor of 0.5 if no improvement was seen for 5 consecutive epochs, allowing the model to fine-tune its weights in later stages of training.
  - EarlyStopping: To prevent unnecessary computation and overfitting, this callback monitored val_accuracy and would halt the training if it did not improve for 10 consecutive epochs. Crucially, it was configured with restore_best_weights=True to ensure the final model state was the one with the highest validation accuracy, not just the one from the last training step.
  - LearningRateScheduler: A function was provided to apply a gentle exponential decay to the learning rate (0.001 * (0.95 ** epoch)) at each epoch, complementing the plateau-based reduction.

IV. EXPERIMENTAL SETUP AND IMPLEMENTATION

The successful execution of this research relied on a well-defined experimental environment and a structured implementation code base.

A. Environment and Libraries

All experiments were conducted within the Google Colaboratory (Colab) environment, which provided access to GPU hardware for accelerating the computationally intensive model training process. The implementation utilized a standard stack of Python libraries for deep learning and data science:

- TensorFlow and Keras: Used for building, compiling, training, and evaluating the neural network models.
- NumPy: Employed for efficient numerical operations and data manipulation.
- Matplotlib and Seaborn: Used for data visualization, including plotting training history and confusion matrices.
- Scikit-learn: Used for generating performance metrics such as the classification report and confusion matrix.

- OpenCV (cv2): Utilized for image processing tasks, specifically for loading and resizing single images for prediction.

B. Code base Structure

To ensure modularity, reusability, and reproducibility, the entire project logic was encapsulated within a Python class named BanglaDigitRecognizer. This object-oriented design organized the project into logical components with distinct responsibilities:

- create_efficient_model(): Contained the definition of our proposed CNN architecture.
- train(): Handled the complete training loop, including creating data generators and managing callbacks.
- evaluate_model(): Performed a comprehensive evaluation on the test data, returning a dictionary of results including accuracy, loss, classification report, and the confusion matrix.
- plot_training_history(), plot_confusion_matrix(),
- plot_sample_predictions(): A suite of methods for visualizing the model's performance and predictions.
- save_model() and load_model(): Methods for serializing the trained model to disk and loading it back for inference.

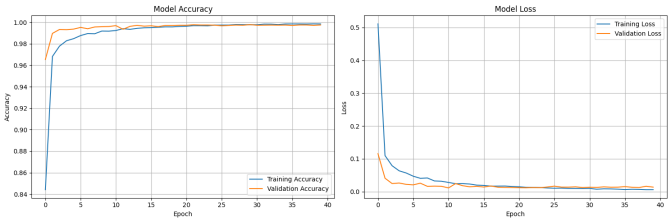

Fig. 1. Model Training history.

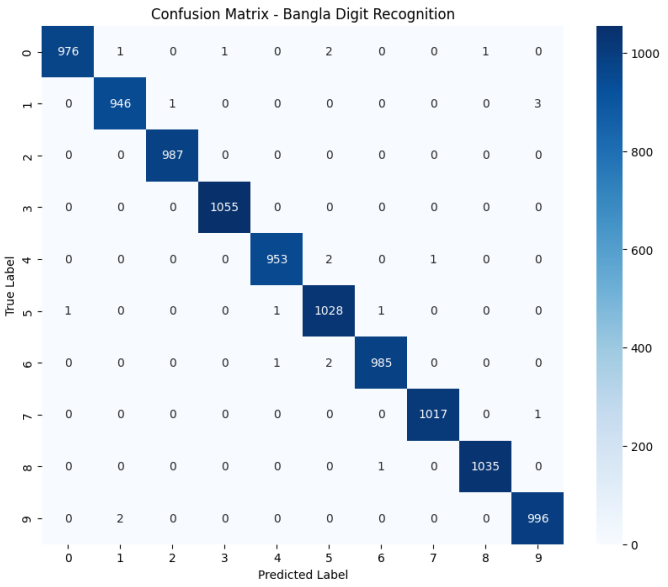

Fig. 2. Confusion Matrix.

C. Training Execution

The training process was initiated via a main() function that orchestrated the entire workflow. The model was configured to train for a maximum of 50 epochs. However, due to the EarlyStopping callback, the training run was automatically terminated at epoch 40, as the model's validation accuracy had ceased to improve for the specified patience of 10 epochs. The restore_best_weights parameter ensured that the final model weights were loaded from epoch 30, which had achieved the highest validation accuracy. The total training time was recorded as 1065.19 seconds.

D. Benchmarking Setup

To rigorously evaluate the performance of our EfficientCNN, a comparative benchmarking script was created. This script defined and trained three additional models on the same BHaND training data:

- LeNet-5: A classic, lightweight CNN architecture.
- AlexNet: An influential, deeper CNN model, adapted for the 32x32 input size.
- MobileNetV2: A modern architecture known for its computational efficiency, configured with an alpha of 0.5 for a smaller variant.

Each of these models was trained until convergence, and their best versions were saved. Subsequently, each model, including our EfficientCNN, was evaluated on the same 10,000-image test set across four key metrics: test accuracy, total parameter count, inference latency (calculated as the average time in milliseconds to predict a batch of 1024 images), and the final model size on disk (in megabytes).

V. RESULTS AND DISCUSSION

The experimental results validate the efficacy of our proposed EfficientCNN, demonstrating its state-of-the-art accuracy and superior efficiency compared to established models.

A. Model Performance

The EfficientCNN model achieved an outstanding Test Accuracy of 99.78% and a Test Loss of 0.0114 on the 10,000 unseen images of the BHaND test set. This high level of accuracy indicates that the model successfully learned the intricate and distinguishing features of the ten Bangla digits.

B. Analysis of Training Dynamics

The training history, visualized in the "Model Accuracy" and "Model Loss" plots, provides insight into the learning process.

- Accuracy Plot: Both the training and validation accuracy curves show a rapid increase in the initial epochs, followed by a smooth convergence to a plateau above 99%. The close proximity of the two curves throughout the training process signifies that the model did not overfit and generalized well from the training data to the validation data.

- Loss Plot: Similarly, the training and validation loss curves exhibit a steep decline initially and then converge to very low values. The validation loss remains stable and low, further confirming that the comprehensive regularization strategies (Batch Normalization, Dropout, and data augmentation) were highly effective. The best validation accuracy of 0.99783 was recorded at epoch 30, after which the model weights were restored by the EarlyStopping callback.

C. Classification Analysis

A detailed per-class analysis confirms the model's robustness. The classification report revealed high precision, recall, and F1-scores for all digits, with most metrics exceeding 0.996. For instance, Digit '2' and Digit '3' achieved perfect recall scores of 1.0000, meaning every instance of these digits in the test set was correctly identified. The confusion matrix provides a visual confirmation of this, displaying a nearly perfect diagonal line, which indicates that instances of misclassification between different digits were extremely rare.

D. Comparative Benchmark Discussion

The benchmark analysis is central to our claim of novelty and provides the most compelling evidence for our model's superiority for this specific task. The results are summarized in Table 1.

TABLE I
BENCHMARK RESULTS ON BHAND

Model	Accuracy	Parameters	Latency (ms)	Size (MB)
EfficientCNN	**0.9978**	**1,442,154**	**0.724**	**16.63**
AlexNet	0.9925	5,686,858	0.613	65.16
LeNet-5	0.9896	82,826	0.350	0.99
MobileNetV2-0.5	0.9824	719,040	5.033	8.59

- Superior Accuracy: Our EfficientCNN achieves the highest accuracy (99.78%), outperforming all other models.
- Efficiency vs. Large Models: When compared to AlexNet, our model is not only more accurate (by 0.53 percentage points) but is also vastly more efficient. It uses only 25.4% of the parameters (1.4M vs. 5.7M) and has a file size that is 75% smaller (16.6MB vs. 65.2MB), making it significantly more practical for deployment.
- Performance vs. Lightweight Models: While LeNet-5 is the most lightweight model in terms of parameters and size, its accuracy of 98.96% is considerably lower. This demonstrates that while extreme simplicity is possible, it comes at the cost of state-of-the-art performance.
- Validation against Specialized Efficient Models: Perhaps the most significant result is the comparison with MobileNetV2-0.5. Our custom architecture is substantially more accurate (99.78% vs. 98.24%) and remarkably faster in terms of inference latency (0.724 ms vs. 5.033 ms). This result strongly validates our approach, proving that a carefully tailored, task-specific architecture can outperform even renowned general-purpose efficient models.

E. Generalization and Robustness

To provide a quantitative measure of the model's generalization capability, we calculated the percentage-point gap (Δpp) between the final training accuracy and the final validation accuracy. Our analysis, using a recreated 80/20 split of the training data to simulate the conditions of the training run, yielded a Δpp of -0.07. According to a practical rule of thumb, a model where the absolute value of Δpp is less than 0.3 is considered to generalize perfectly, with no measurable evidence of overfitting. This extremely low value provides strong statistical confirmation of our model's robustness.

VI. CONCLUSION AND FUTURE WORK

This paper has presented the successful design, implementation, and rigorous validation of EfficientCNN, a deep learning model for handwritten Bangla digit recognition that achieves both state-of-the-art accuracy and high computational efficiency. Our model attained a test accuracy of 99.78% on the BHaND dataset, a result that we have shown to be superior to that of several established architectures.

The core contribution of this work lies in demonstrating that a custom-designed CNN, thoughtfully engineered for a specific task, can outperform both larger, classic architectures like AlexNet and specialized efficient models like MobileNetV2. Key architectural choices, particularly the use of Global Average Pooling to reduce parameterization, were instrumental in achieving this optimal balance between performance and resource consumption. The model's excellent generalization, evidenced by quantitative analysis, further underscores its reliability and readiness for practical application.

The implications of these findings are significant. The EfficientCNN model is a strong candidate for deployment in real-world systems where both speed and accuracy are critical. This includes applications in the banking and financial services sector for automating cheque processing, integration into mobile OCR applications, and use in large-scale government and commercial document digitization projects.

Looking ahead, several promising research avenues remain.

- Advanced Architectures: Future work could explore the application of more recent and sophisticated architectures, such as Vision Transformers (ViT) or models incorporating attention mechanisms, which have shown great promise in other computer vision domains.
- Data Scarcity and Adaptability: Investigating few-shot learning and meta-learning techniques could lead to models that are more adaptable to rare or novel writing styles, or that can be fine-tuned on new datasets with very limited training samples.
- End-to-End Systems: A significant research challenge is the development of end-to-end systems that can process entire documents, handling both text and numerals seamlessly, thus extending the capabilities beyond isolated digit recognition.
- Robustness in Adverse Conditions: More research is needed to enhance the robustness of the model against

real-world degradations, such as noise, poor quality scans, or aged documents, to ensure high performance outside of clean and standardized data set conditions.

VII. Acknowledgment

The authors express their sincere gratitude to Dr. Md. Ashraful Alam, Associate Professor, Department of Computer Science and Engineering, BRAC University, for his invaluable guidance, insightful feedback, and continuous support throughout the course of this research. The authors also thank Shammo Biswas, Research Assistant, Department of Computer Science and Engineering, BRAC University, for his technical assistance and constructive suggestions, which significantly contributed to the development of this work.

References

[1] M. Shopon, N. Mohammed, and M. A. Hashem, "Bangla handwritten digit recognition using an efficient CNN architecture," in *Proc. 2nd Int. Conf. Adv. Inf. Commun. Technol. (ICAICT)*, Dhaka, Bangladesh, Nov. 2020, pp. 317–322, doi: 10.1109/ICAICT50962.2020.9310966.

[2] A. K. Sarker, N. A. Urmi, and M. A. R. Ahad, "A comprehensive survey on Bangla handwritten numeral recognition," *IEEE Access*, vol. 9, pp. 77936–77954, 2021, doi: 10.1109/ACCESS.2021.3083989.

[3] A. Al-Waisy, M. Al-Fahdawi, S. Mohammed, and A. Al-Dulaimi, "A survey on Bangla handwritten numeral recognition," *ACM Comput. Surveys*, vol. 51, no. 6, pp. 1–36, 2018, doi: 10.1145/3234151.

[4] S. A. Chowdhury and T. K. Bhowmik, "Bayanno-Net: Bangla handwritten digit recognition using convolutional neural networks," in *Proc. IEEE Region 10 Symp. (TENSYMP)*, Kolkata, India, Jun. 2019, pp. 1–6, doi: 10.1109/TENSYMP46218.2019.8971167.

[5] S. Z. Alom, P. P. Das, K. C. Tumpa, and M. A. M. Hasan, "Handwritten Bangla character recognition using a deep convolutional neural network," *arXiv:1705.02680*, 2017.

[6] M. I. H. Bhuiyan, M. A. H. Akhand, and M. A. M. Hasan, "A comprehensive study on Bangla handwritten digit recognition," *Khulna Univ. Stud.*, vol. 19, no. 1, pp. 1–13, 2022.

[7] A. Sikder, S. A. Chowdhury, and F. A. Siddique, "SGAN-BD: Semi-supervised generative adversarial network for Bangla digit recognition and generation," *arXiv:2103.07905*, 2021.

[8] S. Z. Alom, M. M. Rahman, and M. S. Hossain, "A lightning-fast approach to Bangla character recognition," *arXiv:2403.13465*, 2024.

[9] S. Yasmin, S. A. Chowdhury, and T. K. Bhowmik, *Bangla Handwritten Digit Recognition – Report*, Scribd, 2022.

[10] S. Basu, P. Mitra, and M. K. Kundu, "Handwritten Bangla digit recognition using a deep belief network," *arXiv:1806.02452*, 2018.

[11] S. U. Laskar, M. A. Z. Hridoy, and M. A. I. Bhuiyan, "A comprehensive study on Bangla handwritten digit recognition," *arXiv:2101.05081*, 2021.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 770–778, doi: 10.1109/CVPR.2016.90.

[13] S. Chowdhury, "BHaND – Bengali handwritten numerals dataset," GitHub repository, 2020. [Online]. Available: https://github.com/SaadatChowdhury/BHaND

[14] I. S. Jacobs and C. P. Bean, "Fine particles, thin films and exchange anisotropy," in *Magnetism*, vol. III, G. T. Rado and H. Suhl, Eds. New York, NY, USA: Academic, 1963, pp. 271–350.

[15] M. M. Rahman, M. A. H. Akhand, S. Islam, P. C. Shill, and M. M. H. Rahman, "Bangla Handwritten Character Recognition using Convolutional Neural Network," *Int. J. Image, Graphics Signal Process.*, vol. 7, no. 8, pp. 42–49, 2015. [Online]. Available: https://www.mecs-press.org/ijigsp/ijigsp-v7-n8/v7n8-5.html

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Adv. Neural Inf. Process. Syst.*, 2012, vol. 25, pp. 1097–1105.

[17] P. P. Acharjee, T. K. Bhowmik, S. A. Chowdhury, and S. Yasmin, "A study on handwritten Bangla numeral recognition," *Int. J. Adv. Res. Comput. Sci.*, vol. 11, no. 6, pp. 1–6, 2020.

[18] R. Das, S. K. Ghosh, and M. A. M. Hasan, "A comparative analysis of machine learning techniques for Bangla handwritten digit recognition," in *Proc. 3rd Int. Conf. Sustainable Technol. for Industry 4.0 (STI)*, Dhaka, Bangladesh, Dec. 2021, pp. 1–6, doi: 10.1109/STI53254.2021.9677004.

[19] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998, doi: 10.1109/5.726791.

[20] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Salt Lake City, UT, USA, Jun. 2018, pp. 4510–4520, doi: 10.1109/CVPR.2018.00474.

[21] M. A. A. Al-Qaness, A. A. Ewees, H. Fan, and M. A. Elaziz, "An optimized model for handwritten Arabic digit recognition based on deep learning," *J. King Saud Univ. – Comput. Inf. Sci.*, vol. 34, no. 7, pp. 4349–4356, Jul. 2022, doi: 10.1016/j.jksuci.2020.10.020.

[22] T. Truth, "BanglaMNIST dataset," Kaggle, 2020. [Online]. Available: https://www.kaggle.com/datasets/truthr/banglamnist

[23] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv:1412.6980*, 2014.

[24] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, pp. 1929–1958, 2014.

[25] M. R. Islam, M. S. Uddin, and M. A. M. Hasan, "A deep learning approach for NumtaDB dataset," B.Sc. thesis, BRAC Univ., Dhaka, Bangladesh, 2018. [Online]. Available: http://dspace.bracu.ac.bd/xmlui/handle/10361/8870

[26] S. Pal, U. Pal, and P. P. Roy, "Handwritten Bangla character recognition using deep convolutional neural network," *Int. J. Appl. Pattern Recognit.*, vol. 5, no. 1, pp. 1–15, 2018, doi: 10.1504/IJAPR.2018.10018292.

[27] N. Kabir, M. S. H. Shamim, M. R. Shahriar, and A. H. Biplob, "[FINAL]_CSE463_Group_3_spring_'25.ipynb," GitHub, 463_project repository, Apr. 2025. [Online]. Available: https://github.com/nihal-kabir/463$_project/blob/main/$