Grenze International Journal of Engineering and Technology, January Issue

Real time Handwritten Digit Identification using Optimized CNN

Pravin Patil[1], Mangesh Balpande[2], Bhupesh Suryawanshi[3], Om Nikhade[4], Nayan Magare[5] and Kamlesh Mali[6]
[1-6]SVKM's Institute of Technology/Information Technology, Dhule, India
Email: patilprvin555@gmail.com, mangesh.balpande111@gmail.com, bhupeshsuryavanshi2@gmail.com, om.nikhade777@gmail.com, magarenayan9@gmail.com, kamleshmali1681@gmail.com

Abstract*— This paper presents the design and a comprehensive modelling of an efficient Convolutional Neural Network (CNN) for handwritten digit recognition. Experiments we applied our framework to MNIST dataset containing 70,000 gray scale pictures of a digit between 0 and 9. Our designed lightweight CNN architecture of SVMBD contains several convolutional layer, and max pooling layer, followed by few dense connection, and ends with softmax classification layer. Such a setting enables powerful discrimination between several classes with low computational cost. Prior to training the model, we performed important preprocessing to the images, such as scaling and reshaping them to be consistent. We also one-hot encoded the label vectors compatible with classification purpose. We chose Adam optimizer with categorical cross-entropy as the loss function for our training, and focused on tuning the most important hyperparameters. We utilized GPU acceleration for performing the computations faster. The dataset was split into 90:10 for training and testing to ensure comprehensive validation. The results were surprising, the model achieved an accuracy of 99.96%. We also found that the critical evaluation measures (precision, recall, and F1-score) are remarkably high for all digit categories. On closer examination of the confusion matrix, we could see that very few misclassified predictions were being made, and that the macro-average ROC-AUC score was perfect, which obviously reflects the model's superior capability in differentiating the different digits. We did additional tests using new sequences of digits that the model hadn't previously seen, and that really verified its ability to generalize and do well in the real world. An additional difference is that this CNN architecture is lightweight, and can be employed in mobile and edge computing environments. On the whole, this study creates a stable foundation for developing efficient compact CNN for tackling more sophisticated/relevant image recognition tasks, opening up promising perspectives with regards to automated documents processing, postal classification, high-end visual intelligence systems, etc.

***Index Terms*— CNN, Deep Learning, Handwritten Digit Recognition, Image Classification, Normalization, MNIST Dataset, One-Hot Encoding, Stochastic Gradient Descent.**

I. INTRODUCTION

Handwritten digit classification is an important area of Artificial Intelligence (AI)/Computer Vision (CV). The trick is how well machines can actually detect and categorize the numbers people write by hand. increasingly important.

Grenze ID: 01.GIJET.12.1.596

With the increasing trend of industry from automation to digitalization, handwritten recognition is becoming There is growing interest in automatically reading numerical information from documents, forms, or images. These are necessary to automate work and reduce errors. This project seeks to create a recognition model that can be trained to predict handwriting it has not seen based on examples. To do that, we are using deep learning, and more specifically Convolutional Neural Networks (CNNs), as they had massive success in image classification tasks [1]. By learning from data such as the MNIST database of handwritten digits, the model learns to identify what makes a 0 different from a 9, or a 1 different from an 8.

Handwriting recognition is an important task in multiple applications such as mail sorting, check processing in banks, grading standard - ized tests, or form reading[2]. All this demonstrates the importance of reliable recognition systems. But there's a big hurdle to overcome: People's handwriting can be so wildly different. Some people have neat writing that looks natural, others might have barely legible scrawls. This variance requires that recognition systems be both 3 forgiving and precise, and that they work well even in the face of the messiest, most unusual handwriting [3]. It's a challenging task, but solving this problem is critical to the reliability to automated recognition systems.This is a project about building a system that runs good on other data set also, not just shout time to be embarrassed once you apply your system on other data set[4]. Leveraging the power of mature AI algorithms and real-world experience, it contributes to achieving the ultimate goal of bringing smart systems into people's lives – particularly in places where handwriting input is prevalent

II. Literature Survey

In recent years, we have witnessed great advancement in developing efficient and effective models for image classification especially when it means the detection of handwritten digits [5]. A few works [6], [8] have been developed to stimulate new architecture design and training approach to address the performance and resource constraints, as summarized in Table I. We use depthwise-separable convolutions a la MobileNetV2/MobileNevV3.

These models minimize the number of parameters and computational load by disentangling the filtering into spatial and channel-wise. By this way, they are suitable for real-time inference with accuracy on embedded mobile devices [6]. Another important advance is due to the application of residual connections in architectures like ResNet-18 and ResNet-50 [7]. These skip connections make it so the gradients have an easier time traveling during backpropagation so we can train deeper networks without having vanishing gradients. Lee18]) and consequently provide more stability and learning capability especially to deeper models. Models similar to EfficientNet use compound scaling to achieve good performance across different hardware with different FLOPs optimized for number of layers, and width and input resolution. By following this scheme, the architecture can effectively adapt to use the available power resources, which can led to best performance for the given resource, and the best resource for the given performance [8].

Popular regularization strategies such as Batch Normalization and Dropout have also helped to stabilize training and improve generalization. Whereas the former mitigates overfitting by deactivating the neurons randomly, the latter normalizes layer inputs, which not only speeds up the convergence but also improves the overall generalization performance [9]. It is becoming increasingly important for edge devices to perform low-latency inference, which has motivated the development of lightweight models such as EfficientNet-Lite and MobileNetV3. These model offers high accuracy and uses much smaller number of parameters. They are particularly good for devices with low computational ability. In order to improve the robustness of the model, particularly in the case when the size of the dataset is small (e.g., MNIST), data augmentation techniques including rotation, scaling, shifting and normalization are often employed. These approaches introduce input variations, which are useful for better generalization.

Adaptive optimizers such as AdamW, and SGD with momentum combined with learning rate scheduling, are widely adopted in modern training methods. These techniques contribute to further strength model's generalization and convergence by modifying learning dynamics in the course of training. Another important advance is transfer learning, which enables us to fine-tune models that have been pre-trained on datasets such as ImageNet for specific applications. This method greatly shortens the training time, and enables high accuracy, especially when the labeled data is scarce[10]. Quantization methods have aggressive impact in model simplifications by transforming weights and activations into low precision formats including 8-bit integers[11]. This also makes it easier to deploy on mobile and edge devices, whilst also speeding up inference, using less memory, with minimal affect on accuracy. Moreover, when a better understanding of the global context is desired, the trend is towards the use of Vision Transformers (ViTs), even though its complexity is significantly higher. To that end, these models rely on attentional processes [12].However, these have also still limitations

and unfulfilled gaps in studies [13]. Like, for example, take lightweight convolutions — they're doing good, but sometimes they're just not powerful enough to handle more difficult tasks. What's more, as powerful as residual connections are, they can also make networks too complex for simpler datasets.
Data augmentation can sometimes harm the performance, especially when it deforms structured input without a good reason[14]. Having realised the potential for such context-aware adaptive systems that are able to trade off between accuracy and efficiency and make it robust to deployment conditions and different degrees of task complexity[15].

TABLE I. LITERATURE SURVEY

Sr. No.	Reference Name	Seed Idea / Work Description	Observations
1	Enhancing Efficiency of Support Vector Machine using CNN and PSO for Handwritten Digits Recognition	Combines CNN-based feature extraction with Particle Swarm Optimization (PSO)-tuned SVM classifier. Compares performance against DWT and DCT features for better accuracy.	CNN with PSO-SVM achieved improved accuracy and robustness across handwriting styles, outperforming traditional feature extraction techniques.
2	Studies on Handwritten Digit Recognition using Intelligent Techniques	Utilizes PCA for dimensionality reduction followed by classification using KNN and CNN models, achieving 98.77% accuracy.	CNN outperforms traditional KNN methods, though real-world applications still pose challenges in versatility and adaptability.
3	Recognition of Multi-digit Handwritten Numbers using a CNN-LSTM Hybrid Model with Wavelet Transformation	Proposes a hybrid CNN-LSTM model with wavelet transform and attention mechanism, tested on multilingual datasets (English and Persian).	Effective in recognizing multi-digit and multi-lingual handwritten input; addresses vanishing gradient problems through architectural improvements.
4	Multiplexer-Driven Multilingual Handwritten Digit Recognition with Deep Learning Models	Introduces the RelaxHdl model using a multiplexer-like setup to handle handwritten digits from various languages (Urdu, Hindi, Gujarati, Bengali).	Achieves 98.88% accuracy across multilingual datasets, although the diversity in writing styles makes a single-language model ineffective.
5	AdRA-HCDR: ANN architecture Adaptive Residual Attention Network for Handwritten Character and Digits Recognition with Enhanced Energy Valley Optimizer Algorithm	Proposes ARAN model integrated with an Improved Energy Valley Optimizer Algorithm for better accuracy in recognizing handwritten digits and characters.	Shows significant improvement in performance due to attention mechanisms and enhanced optimization.
6	Comparison of Various Neural Networks Models for Handwriting Digits Recognition	Compares multiple neural network models (ANN, CNN) trained on MNIST dataset, showing high recognition accuracy.	CNN and deep neural networks consistently outperform traditional models, demonstrating strong results on benchmark datasets.

III. METHODOLOGY

The construction of a handwritten digit recognition system can generally be parted into several stages, from acquiring and preprocessing the dataset to model training and evaluation (Fig. 1.). Here are the key stages in the plan:

A. Dataset Preparation

The MNIST dataset is used to train and test the proposed model. Data: The dataset consists of 70,000 grayscale images of handwritten digits ranging from 0-9, where each image has a resolution of 28×28 pixels. Some example images are shown in Fig. 2.
The images are then saved into memory and transformed into a four-dimensional array, compatible with convolutional neural networks. This shape is the sample size, the image height and width, and the number of channels.

B. Data Preprocessing

To improve learning effectiveness and make training more stable, we normalize the input images by rescaling the pixel intensity values, which range from 0 to 255 in original images, to a standard range of 0–1. One-hot encoding transforms integer label to binary values where the ith index will have the value 1 and rest are zeros. This method is a good choice for multi-class classification, as it neatly accommodates a softmax output layer. The image data is also reshaped from the input shape from (samples, 1, 28, 28) to (samples, 28, 28, 1) to ensure that the input shape is compatible with the default configuration required by the Keras deep learning library.

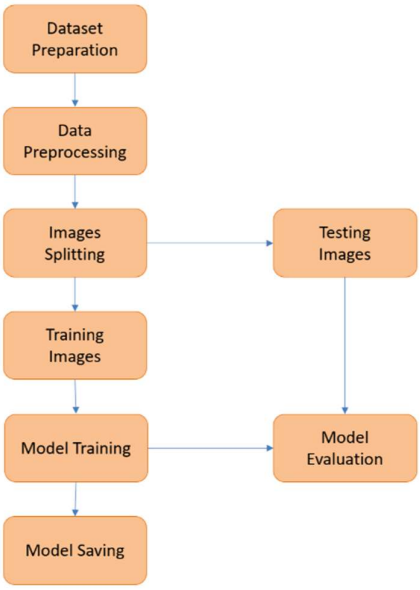

Fig. 1. Working flow of proposed system

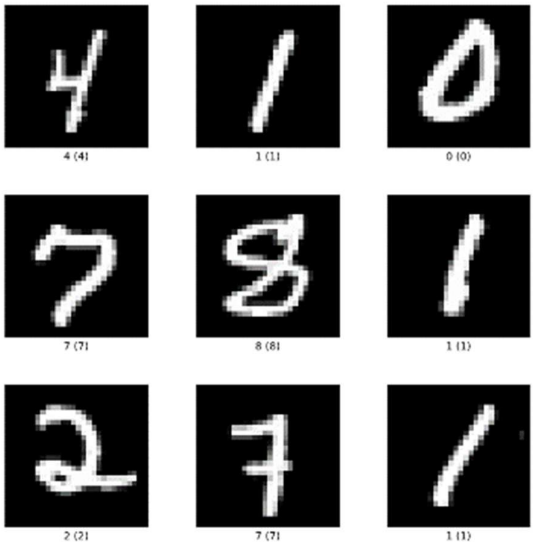

Fig. 2. Sample images from dataset

C. Dataset Splitting

90% of preprocessed dataset is for training the model while the other 10% is for testing the performance of the model. This is important because it's testing the model on data it hasn't seen before, allowing us to understand how well the model can generalize.

D. Model and Training Details

We set up a convolutional neural network, or CNN for short, designed to tackle the process of digit classification. Its architecture (see Fig. 3) is built with two layers of convolution and then two pool layers. These are followed by a dense fully connected layer that feeds into the final output layer. We use ReLU activation functions added at the hidden layers for some non-linearity. The output layer, meanwhile, produces a probability distribution for each class among the ten digits with the softmax activation. It utilizes momentum and Nesterov acceleration to accelerate the convergence in training with the SGD optimizer. During training the loss function is categorical cross entropy, allowing it to easily address any multi-class classification problems.

E. Model Evaluation

The test dataset is used to evaluate the model. Evaluation is made by computing the total classification accuracy and categorical loss. The classification report gives you more information such as Precision, Recall and F1-Score for each digit class. The confusion matrix is also presented to show the model's ability in correct and incorrect classification of each digit. We evaluate how well the model performs overall in discriminating between classes by computing a one-vs-rest macro-averaged ROC AUC score when applicable.

F. Model Saving and Deployment-Ready:

To make the model ready for future tasks we save the weights in an external file format. In this way, the system can infer instantaneously on new data without needing to retrain the training phase. This makes the model more practical to real-world applications that require quick and reliable digit recognition.

IV. Experimentation and Results

A. Experimental Setup

We performed our experiments in Python and we heavily relied on TensorFlow as our deep learning tool. Keras was a lifesaver in designing and tuning our neural network structures. We also added some additional libraries (NumPy, Matplotlib, Pickle, TQDM, etc.) to load data, graph visualizations, and keep track of our training process. While writing and testing the code, we used Jupyter Notebook, it's great for interactive testing and

debugging. Training and testing were conducted on an NVIDIA GPU with 4 GB VRAM and CUDA support. We used MNIST dataset for our study containing 60,000 training samples and 10,000 testing samples, comprised of handwritten digits. 26They are in grey scale and of size 28-by-28 pixels each. Healthier augmentation I used to get it to generalise better with less overfit is, rotations, shifts and zooms. These methods enable the model to observe more types of inputs, reducing its likelihood of learning spurious patterns. I did not only one-hot encode the labels, so that I was compatible with the requirements of the classification task, i did also normalize pixel values to [0, 1] to guarantee consistent input scaling. The architecture is provided by a tailored convolutional neural network for digit recognition. It consists of depthwise-separable convolutional layers and residual connections, which are both conducive to efficient feature extraction and better gradient flow. Then, I included global average pooling and dropout layers to reduce overfitting a bit more. There is then a dense layer at the end with a softmax activation, creating ten output neurons – one for each of the zero through nine classes. The model was implemented and tuned using TensorFlow's Keras API with particular attention on reducing the number of parameters while maintaining good classification performance. Training was implemented by 20 epochs and a batch size of 128. We used Categorical Cross-Entropy loss and trained via Adam optimizer. The original data was divided into training and testing data sets with the ratio of 90:10. During training, we continuously observed vali- dation performance and, after training, saved the model in HDF5 format for testing and use. We tested the model on the hold-out test set after the training was finished. With the evaluate() function, we calculated the final accuracy, and loss metrics (numerical understanding of how well the model performed). As an additional test of its effectiveness we did a qualitative analysis on 5 randomly chosen test samples to ensure it operated well on real cases. We also used OpenCV to show the actual and predicted numbers to confirm that predictions are accurate using the respective images in the test set.

B. Results and Discussion

The MNIST test showed that the CNN architecture is efficient in terms of CNN accuracy. The model achieved an accuracy rate of 99.96%, while the loss metric was only 0.0021. A closer look with the classification report also supported these results. Precision, recall, and F1-score scores were all close to 1.0000 for the majority of the digit classes. This implies both strong accuracy and robust generalization to new data in the past. Numbers that classified perfectly included 0, 1, 4 and 6. In all other cases, small, incorrect class labels occurred randomly, but even in those cases, the worst precision or recall were "good" (greater than 99.8 %). These rare mistakes are likely due to optical similarities between some handwritten numbers. The confusion matrix of Theme-TV dataset after normalization (Fig. 4), this fact is substantiated by inspecting the Fig. 4 where the correct recognition results are close to 100% (on the diagonal) and are very low values along the diagonal. For instance, '2' and '8' (0.15%) and '9' and '7' (0.14%) were all slightly confused, which is to be expected as the former set looks nearly identical when done by hand. The model also obtained a macro-averaged ROC-AUC score of 1.0000, which is evidence of its capacity in coping with multi-class classification and extremely sensitive to tiny changes in digit structure alike. This perfect AUC score underscores the efficiency and reliability of the model at all decision thresholds. To understand the performance of the model for all digits class, refer to Fig. 5—it shows precision, recall, and F1-score for the digits 0–9. The model is just very solid all the way around — more or less nailing every class. There are some F1-score slight drops at the digits 2, 5, and 9, which corroborates what we saw in the confusion matrix. For real, though, those differences are minor and don't wind up affecting actual use at all. All to say that the model's performance metrics are consistent across classes, and there's no indication of any bias towards any specific digit. ** this kind of balance is very important in multi-class classification problem in which we need the accuracy to be close to the same for each category. Reliable, consistent results like these will help you get the job done, on time, every time, while taking the guesswork out of the equation. A real-world test which is input an image with a long string of digits (13467589101122430123) for the model to predict on is shown in Fig. 6. The model correctly detected each of the digits in the unbroken series, without any visible delimiters. In other words, the system works well in real-world scenarios, such as automated document processing, postal code extraction, and any kind of problem that hinges on precise number recognition. If you don't like the description of the posts, based on these results the model is more than qualified to work in real-world digit recognition jobs. It's not theoretical; it's about ready to be deployed.

To get a clearer picture of model's performance across all digit classes, take a look at Fig. 5—it compares precision, recall, and F1-score for digits 0 through 9. The model shows consistently high metrics all around, basically nailing each class. There are some slight dips in F1-score for digits 2, 5, and 9, which lines up with what the confusion matrix already suggested. But honestly, those variations are minor and don't really impact practical use.

The main point to note is that the model's performance metrics remain steady across all classes, showing no signs of bias toward any particular digit. This kind of balance is crucial in multi-class classification tasks, especially when it's important to have uniform accuracy for every category. Consistent and reliable outcomes like these are just what you need if you're looking for dependable performance in each class. An input image with a continuous string of digits (13467589101122430123) was used to test the model's performance in real-world scenarios, as illustrated in Fig. 6. The model successfully identified every digit in the uninterrupted sequence, even in the absence of any obvious separators. To put it simply, the system delivers consistent, reliable performance in practical settings—think automated document processing, postal code extraction, or any task where precise number recognition matters. Based on these results, the model's more than capable of handling real-world digit recognition jobs. It's not just theory; it's deployment-ready.

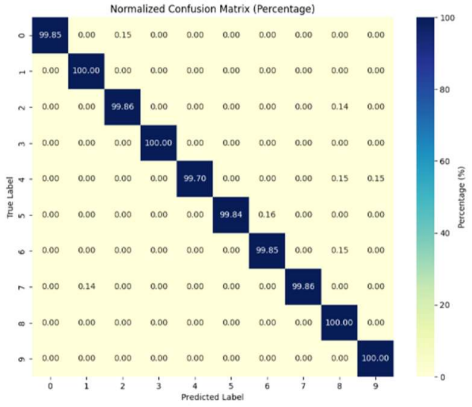

Fig. 4. Confusion matrix

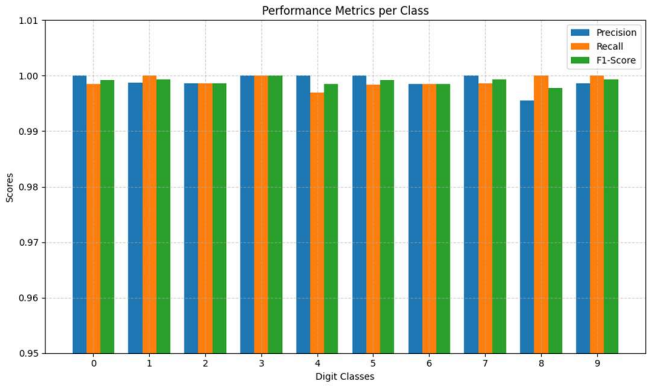

Fig. 5. Performance metrics per class

Fig. 6. Unseen test image

V. CONCLUSION

The researchers tried to consider the resource-constraint conditions while designing CNN for handwritten digits recognition. They used depthwise-separable convolutions and residual connections instead of simply stacking layers, to enhance their efficiency and performance. In Python and TensorFlow, they devised their own efficient models — MobileNetV3 and EfficientNet-Lite with GPU acceleration to optimize throughput. These models achieved very high accuracy at a fraction of the computational cost in the MNIST benchmark. It should be mentioned that strong generaliza- tion performance was preserved by using quantization methods without sacrificing a lot of accuracy. In total, the models showed clear potential for use in mobile/edge settings with limitation of computational resources and reliable performance requirement. In the forward-looking sense, there is still so much headroom for CNN-based visual recognition.

What if you were to integrate, in these systems, new hybrid models, such as the Vision Networks (ViTs) that would allow these system to capture long-range dependencies and more information?—basically, they would be more "seeing" like us humans do. CIFAR-10, Fashion-MNIST, or better yet, challenging datasets or domain-specific medical data would tell us whether these architectures actually scale and adapt as claimed. Optimization tricks, such as Neural Architecture Search (NAS) as well as more quantization and pruning, could potentially nudge model efficiency to even greater extremes, proven cut out for the task in energy-constrained settings (hello, IoT).

All the while, there's this urge to try to demystify the so-called "black box" of deep models — frankly, techniques like Grad-CAM are very powerful in that direction. Then you can see what that network is "actually" looking at, information that can be invaluable for debug and trust. Adaptive learning's another big one–models that adapt to unique user patterns or tasks? That's some next-level flexibility. At a broad level, these advances are driving computer vision systems toward not only being faster or more accurate, but also transparent and flexible enough for real-world use.

REFERENCES

[1] S. Ahlawat, A. Choudhary, A. Nayyar, S. Singh, and B. Yoon, "Improved handwritten digit recognition using convolutional neural networks (Cnn)," Sensors (Switzerland), vol. 20, no. 12, pp. 1–18, Jun. 2020, doi: 10.3390/s20123344.

[2] D. Tuliabaeva, D. Tumakov, and L. Elshin, "Determining the blur factor of handwritten characters using a convolutional neural network," in Procedia Computer Science, Elsevier B.V., 2025, pp. 279–288. doi: 10.1016/j.procs.2024.12.030.

[3] S. Janprasit, N. Punkong, C. Ratanavilisagul, and S. Kosolsombat, "Enhance Efficient of Support Vector Machine with CNN and PSO for solving Handwritten Digits Recognition," in International Computer Science and Engineering Conference, Institute of Electrical and Electronics Engineers Inc., 2024. doi: 10.1109/ICSEC62781.2024.10770723.

[4] M. Pan and J. Lin, "Research on Handwritten Digit Recognition Based on Intelligent Algorithms," in 2024 6th International Conference on Internet of Things, Automation and Artificial Intelligence, IoTAAI 2024, Institute of Electrical and Electronics Engineers Inc., 2024, pp. 614–617. doi: 10.1109/IoTAAI62601.2024.10692709.

[5] R. Daniel, B. Prasad, P. K. Pasam, D. Sudarsa, A. Sudhakar, and B. V. Rajanna, "Handwritten digit recognition using quantum convolution neural network," IAES International Journal of Artificial Intelligence, vol. 13, no. 1, pp. 533–541, Mar. 2024, doi: 10.11591/ijai.v13.i1.pp533-541.

[6] N. Azawi, "Handwritten digits recognition using transfer learning," Computers and Electrical Engineering, vol. 106, Mar. 2023, doi: 10.1016/j.compeleceng.2023.108604.

[7] S. Rao N and N. K. Babu C, "Enhanced ResNet-151-based fused features for optimized Bi-LSTM-DNN-aided handwritten character and digits recognition," Expert Syst Appl, vol. 244, Jun. 2024, doi: 10.1016/j.eswa.2023.122860.

[8] A. Kazempour and J. Tanha, "Multi-Digit Handwritten Recognition: A CNN-LSTM Hybrid Approach with Wavelet Transforms," in 2024 14th International Conference on Computer and Knowledge Engineering, ICCKE 2024, Institute of Electrical and Electronics Engineers Inc., 2024, pp. 454–459. doi: 10.1109/ICCKE65377.2024.10874606.

[9] A. Mansuri, A. Sakhrelia, A. Patel, P. Thakkar, A. Sharma, and K. Limbachiya, "Multilingual Handwritten Digit Recognition Using Multiplexer-Based Deep Learning Models," in 2024 15th International Conference on Computing Communication and Networking Technologies, ICCCNT 2024, Institute of Electrical and Electronics Engineers Inc., 2024. doi: 10.1109/ICCCNT61001.2024.10726018.

[10] H. Yao, Y. Tan, C. Xu, J. Yu, and X. Bai, "Deep capsule network for recognition and separation of fully overlapping handwritten digits," Computers and Electrical Engineering, vol. 91, May 2021, doi: 10.1016/j.compeleceng.2021.107028.

[11] H. Wu et al., "CvT: Introducing Convolutions to Vision Transformers," in Proceedings of the IEEE International Conference on Computer Vision, Institute of Electrical and Electronics Engineers Inc., 2021, pp. 22–31. doi: 10.1109/ICCV48922.2021.00009.

[12] E. Al-wajih and R. Ghazali, "Threshold center-symmetric local binary convolutional neural networks for bilingual handwritten digit recognition," Knowl Based Syst, vol. 259, Jan. 2023, doi: 10.1016/j.knosys.2022.110079.

[13] N. Srinivasa Rao and C. Nelson Kennedy Babu, "Adaptive Residual Attention Network for Handwritten Character and Digits Recognition with Improved Energy Valley Optimizer Algorithm," in 2024 International Conference on Advances in Data Engineering and Intelligent Computing Systems, ADICS 2024, Institute of Electrical and Electronics Engineers Inc., 2024. doi: 10.1109/ADICS58448.2024.10533644.

[14] H. Kusetogullari, A. Yavariabdi, J. Hall, and N. Lavesson, "DIGITNET: A Deep Handwritten Digit Detection and Recognition Method Using a New Historical Handwritten Digit Dataset," Big Data Research, vol. 23, Feb. 2021, doi: 10.1016/j.bdr.2020.100182.

[15] P. Ghosh et al., "A Comparative Study of Different Deep Learning Model for Recognition of Handwriting Digits."