Received: 30 April 2024 | Revised: 26 March 2025 | Accepted: 25 June 2025 | Published online: 8 September 2025

REVIEW

Artificial Intelligence and Applications
2026, Vol. 4(4) 469–488
DOI: 10.47852/bonviewAIA52023297

BON VIEW PUBLISHING

Review of the Performance of CNN Models on Handwriting Recognition

Sabyasachi Mazumder[1], Sayan Neogy[1], Sahana Das[2], Kaushik Roy[3,*] and Umapada Pal[4]

[1] *School of Computer Science, Swami Vivekananda University, India*

[2] *Department of Computer Science & Engineering, Budge Budge Institute of Technology, India*

[3] *Department of Computer Science, West Bengal State University, India*

[4] *Computer Vision & Pattern Recognition Unit, Indian Statistical Institute, India*

Abstract: Handwritten text recognition continues to be a dynamic area of research, with practical uses such as processing bank checks, digitizing historical manuscripts, enabling handwriting-based user interfaces, and other OCR-related applications. Despite its potential, the task remains difficult because of a wide variety of handwriting styles, stroke patterns, and visual structures found across different writing systems. This research provides a comprehensive comparative study of deep convolutional neural network (CNN) architectures for handwritten character and word recognition of world popular scripts such as Roman (English), Devanagari, Bengali, Tamil, Telugu, Hiragana, and Arabic. Most recent and popular works on CNNs are considered. We conduct comprehensive benchmarking of widely used CNN architectures—such as VGG, ResNet, and Inception—on handwritten datasets spanning multiple scripts. Our experimental findings yield important information regarding comparative performances under diverse conditions, in addition to insights regarding the impact due to architectural extensions, i.e., attention mechanisms and regularization schemes, to recognition performance. Ensemble schemes, i.e., majority voting and stacking to obtain additional boost in performance, yield measurable increments in prediction faithfulness. Our investigation encompasses all training, validation, and testing stages and discovers key patterns such as overfitting tendencies, specifically for scripts with high visual complexity. These observations emphasize careful model selection and provide practical suggestions regarding designing robust, script-aware CNNs for multilingual handwritten text recognition.

Keywords: CNN, comparative study, HCR, word recognition

1. Introduction

Handwritten text recognition is an active research area with applications in bank check processing, digitization of historical documents, handwriting-based interfaces, and other OCR use cases. However, automatically recognizing handwritten text remains challenging because of the high variability in human writing styles, formats, stroke patterns, and visual appearance.

Earlier approaches relied extensively on manual feature engineering and traditional machine learning models such as support vector machines (SVMs) and k-nearest neighbors (KNNs), which require substantial expertise in feature crafting and tend to generalize poorly to unseen handwriting data.

In recent years, deep convolutional neural networks (CNNs) have emerged as a prominent technique for image classification tasks. CNN architectures such as LeNet, AlexNet, and VGGNet have shown state-of-the-art results on benchmark datasets, significantly outperforming classic models. A key advantage of CNNs is learning hierarchical representations directly from input images without extensive feature engineering. However, research continues to optimize CNNs for handwritten text modeling across different scripts. Each script has unique visual characteristics. Specialized modeling techniques are necessary for optimal handwritten text recognition accuracy across them. This requires a comparative study analyzing diverse CNN architectures on handwritten datasets covering major world scripts.

In this paper, we provide a comprehensive experimental study benchmarking standard CNN models for handwritten character and word recognition across various scripts, including English, Devanagari, Bengali, Tamil, Hiragana, and Arabic. The models evaluated include LeNet, AlexNet, VGG, ResNet, and Inception. We chose these scripts to cover a diverse set of writing systems, including Latin, Indic, and logographic scripts. The computational complexity of the models is analyzed by reporting floating point operations (FLOPs). In addition, the inference time of each model is measured and reported. Ensembling is also performed to further boost accuracy. By benchmarking diverse CNNs, we extract key insights to guide optimal model selection per script. Our findings elucidate architectural considerations and innovations to advance customized deep CNN designs for generalized handwritten text recognition spanning multiple scripts and languages. The comparative analysis provides data-driven guidelines to tailor CNN model complexity and feature learning based on target script intricacy. This equips researchers with knowledge to enhance multiscript handwriting understanding through script-specific CNN modeling.

The key contributions of this study are the following:

1) Extensive comparative evaluation of various diverse CNN models on handwritten datasets covering widely used handwritten scripts.
2) Reporting and analysis of various metrics such as accuracy, overfitting, computational complexity (layers, parameters, and FLOPs), and inference time.

***Corresponding author:** Kaushik Roy, Department of Computer Science, West Bengal State University, India. Email: kaushik@wbsu.ac.in

© The Author(s) 2025. Published by BON VIEW PUBLISHING PTE. LTD. This is an open access article under the CC BY License (https://creativecommons.org/licenses/by/4.0/).

3) Investigation into ensembling techniques to combine multiple models and enhance performance.
4) Deriving data-driven, script-specific guidelines for guiding optimal architectures based on visual complexity.
5) Providing insights to advance deep CNN techniques for multiscript handwritten text recognition tasks.

2. CNN-Based Works on Handwriting Recognition

Handwritten text recognition [1, 2] has been extensively studied because of its diverse applications in bank check processing, digitization of historical manuscripts, handwriting-based user interfaces, and more. However, accurately recognizing handwritten text [3, 4] remains challenging because of the high variability in human writing styles, formats, stroke patterns, and visual appearance.

Earlier approaches relied extensively on manual extraction of handcrafted features combined with traditional machine learning models such as SVMs [5–8] and KNNs for classification. However, these methods require substantial expertise in designing discriminative features and tend to generalize poorly to unseen handwriting data.

With the resurgence of deep learning, CNNs have emerged as the state-of-the-art technique for handwritten text recognition. By leveraging large labeled datasets and modern GPUs, deep CNN models [9–11] have achieved remarkable accuracy on benchmark datasets, significantly outperforming classic approaches.LeCun et al. proposed the pioneering LeNet architecture in 1998 [11] for digit recognition, which outperformed prior non-neural techniques. The introduction of AlexNet [12] in 2012 fueled further interest in deep CNNs by dramatically surpassing previous models on the ImageNet dataset. Building on this, several improvements, such as using smaller convolutional filters [4], smaller stride [13], Inception modules [3], and residual connections [14] to further boost accuracy, were proposed.

Researchers have developed sophisticated deep CNN architectures [13, 15–17] specifically tailored for handwritten text recognition tasks by combining convolutional, max-pooling, normalization, and fully connected layers. Convolutional layers learn hierarchical visual features from raw pixel inputs, whereas pooling layers induce invariance to small translations and distortions. Normalization layers improve model generalization capability. Fully connected layers finally map the learned features into character or word classification probabilities.

Several comparative studies have analyzed the efficacy of different deep CNN models on handwritten text recognition across various scripts. Chidrawar and Dhamdhere [18] evaluated VGGNet architecture on handwritten MODI script datasets (an ancient Indic script). They found that deeper models such as VGG16 consistently outperform shallow networks, indicating that model depth is highly beneficial for learning salient visual features. Rectified linear unit (ReLU) activations improved accuracy compared to older sigmoid/tanh activations due to reduced gradient saturation. Ma et al. [19] compared vanilla CNN, residual CNN with attention mechanisms, and fully convolutional classifiers on multilingual script identification tasks across scene text images containing Latin, Chinese, Arabic, and Devanagari scripts. Their findings showed that residual CNN with attention mechanisms achieved the highest accuracy, highlighting the importance of depth and attention for modeling complex stroke patterns and local discriminative features.

A major focus of recent research is advancing handwritten text recognition methods to handle diverse scripts. This presents significant challenges due to the high diversity in character sets [20–23], writing styles, shapes, sizes, and stroke patterns across different scripts. Several approaches have been investigated to effectively model multilingual handwritten data.

1) Unified Models [24–26]: Training a single deep CNN model on aggregated datasets combined across multiple scripts. Requires access to large volumes of handwritten data covering all scripts.
2) Script-Specific Models [14, 27, 28]: Developing customized deep CNN models individually tailored and optimized for each script, which are then ensemble combined. Allows targeted tuning specialized for each script.
3) Data Augmentation [17, 29, 30]: Generating synthetic training samples via transformations such as affine distortions and elastic deformations to significantly expand the limited quantity of real handwritten training data.
4) Ensemble Models [27, 31, 32]: Combining predictions from an ensemble of multiple diverse deep CNN models can potentially improve stability and accuracy compared to individual models.

Sharma and Jayagopi [33] proposed a lightweight offline handwriting recognition model combining 2D CNNs with dilated temporal convolution networks (DTCN), followed by a CTC layer. They demonstrated that this architecture achieves comparable accuracy to RNN-based models and is faster and more resource-efficient. They highlighted the importance of designing tailored deep purely convolutional architectures combined for efficient handling of multiscript scenarios. Zhong et al. [28] presented a comprehensive review of offline handwritten Chinese character recognition methods, focusing on CNN-based architectures, traditional feature extraction techniques, and filtering methods for noise reduction. They highlighted the importance of deeper CNN architectures and suitable activation functions to improve recognition accuracy on complex Chinese scripts.

Zhang et al. [34] proposed IMTLM-Net, a dual-stream transformer-based model that processes handwritten English text images by jointly modeling image and text sequences. Their method integrates local feature extraction to better capture fine-grained visual details, leading to improved accuracy on challenging handwritten datasets compared to prior single-stream transformer models.

Giménez et al. [35] proposed an offline handwriting recognition method that directly feeds binarized text images into Bernoulli Hidden Markov Models (BHMMs), enhanced with a sliding window and repositioning techniques. Their experiments on Latin and Arabic scripts showed improved accuracy over conventional BHMMs, demonstrating the effectiveness of context-aware window sampling.

Ashlin Deepa and Rajeswara Rao [36] proposed a nearest interest point classifier for offline Tamil handwritten character recognition. Their method directly matches high-dimensional feature vectors between images without relying on machine learning or deep learning approaches. Experiments on the standard HP Labs Tamil handwritten character dataset achieved 90.2% recognition accuracy, demonstrating competitive performance compared to existing classifiers.

In 2024, new techniques were introduced to further boost the performance of deep CNN models for handwriting recognition. Humayun et al. [37] developed an ELBP-based sequential CNN architecture for offline English handwritten character recognition. The method combines enhanced local binary pattern (ELBP) features with a CNN to improve feature representation and classification accuracy. Experiments on the EMNIST dataset showed that this approach outperformed several pretrained CNN models, demonstrating its effectiveness for handwritten character recognition tasks [37].

Zhang et al. [38] proposed a graph convolutional network-based method for detecting irregular and curved scene text in natural images. By integrating a fully convolutional network for text region extraction with a GCN for text line grouping, their approach achieved state-of-the-art results on multiple public benchmarks, demonstrating the effectiveness of combining CNN feature extraction with relational reasoning [38].

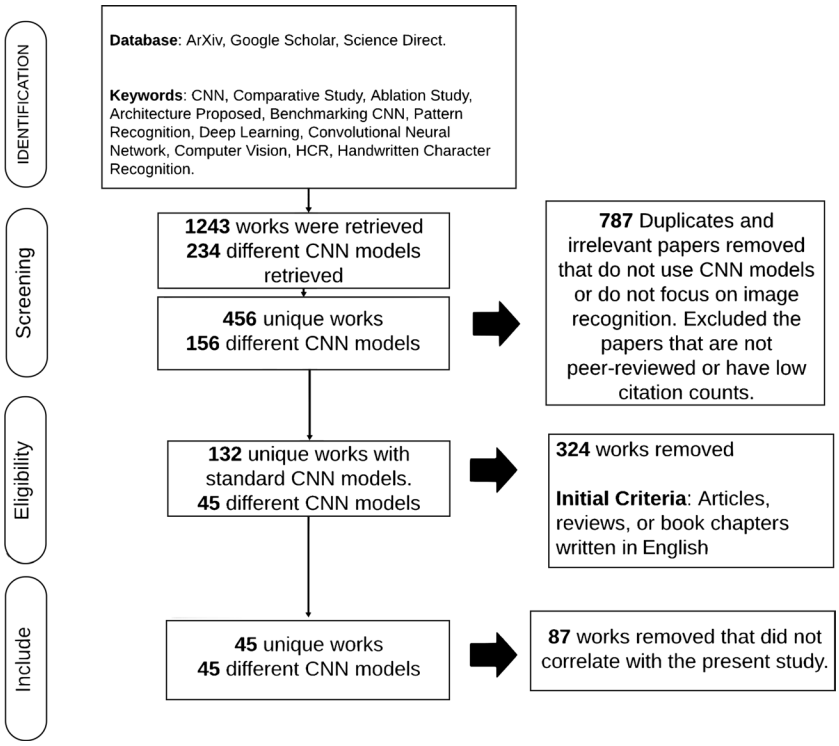
Kaur and Kumar [39] provided a comprehensive survey on word recognition approaches for both non-Indic scripts such as English and Indic scripts such as Bangla and Hindi. Yavartanoo et al. [40] proposed PolyNet, a deep neural network that learns polygon-based representations, illustrating the broader applicability of specialized architectures for complex shape and pattern recognition tasks. For the Dravidian script Tamil, Ashlin Deepa and Rajeswara Rao [36] developed a novel nearest interest point classifier using CNNs for offline handwritten character recognition (HCR). Joseph Raj et al. [41] tackled bilingual text detection from natural scene images by combining faster R-CNN and extended HOG features. Liang et al. [42] presented an online overlaid handwritten Japanese text recognition system tailored for small tablets using CNN models. These works highlight the diversity of recent CNN-based approaches for handwritten character and word recognition across multiple scripts beyond just Latin alphabets.

In summary, deep CNN architectures have become the dominant technique for handwritten text recognition, enabled by large datasets and modern hardware. However, specialized architectures, regularization methods, and transfer learning are necessary to achieve state-of-the-art performance across the diversity of scripts worldwide. In our study, we benchmark diverse CNN models across scripts to derive insights into model selection, tuning, and innovations to advance handwritten text recognition across languages.

To study handwritten text recognition, various CNN architectures were evaluated on character and word image datasets across different scripts, including English, Devanagari, Bengali, Tamil, Hiragana, and Arabic. For character recognition, models were trained and tested on standard datasets such as EMNIST, Ekush, and Kuzushiji-49, encompassing both simple and complex scripts. For handwritten word recognition, datasets consisting of cropped word images were used for major scripts such as English, Hindi, Bengali, Tamil, and Arabic. The benchmarked models included classic CNNs such as LeNet, modern architectures such as ResNet, Inception, and MobileNets tailored for handwritten modality. From 45 CNN models (LeNet-5 [43], AlexNet [44, 45], VGG16 [43, 44, 46, 47], ResNet-50 [9], DenseNet-121 [48], InceptionV3 [43, 49, 50], Xception [51], MobileNet-V2 [52, 53], ShuffleNet-V2 [54], SqueezeNet [55], ZFNet [26], GoogLeNet, SENet, NASNet-A, MnasNet-A1, EfficientNet-B0 [56], ResNeXt-50, Wide ResNet-50, PolyNet, PyramidNet-200, DPN-92, RCNN-152, DCNN-152, MCNN-152, GCNN-152, CCNN-152, ACNN-152, PCNN-152, SCNN-152, LCNN-152, FCNN-152, TCNN-152, OCNN-152, ICNN-152, ECNN-152, BCNN-152, NCNN-152, KCNN-152, QCNN-152, UCNN-152, VCNN-152, WCNN-152, XCNN-152, YCNN-152, and ZCNN-152), we selected a diverse subset of 22 architectures: LeNet, AlexNet, ResNet, GoogLeNet/InceptionNet [43, 49, 50], EfficientNet [56], VGG [43, 44, 46, 47], DenseNet [48], ZFNet [26], PyramidalNet, Feature-Map-based CNNs,

Attention-based CNNs [57–59], MobileNetV1 [60], Wide ResNet [61], Squeeze and Excitation Networks [62], Competitive Squeeze and Excitation Networks [63], Highway Networks [64], InceptionV4 [65], PolyNet [66], Xception, Depth-based CNNs [67], Residual Attention NN [68], and Inception–ResNet [69] (as shown in Figure 1). These represent the most influential, popular, and state-of-the-art CNNs for handwritten text recognition. They showcase innovations in depth, attention, efficiency, and connectivity patterns crucial for modeling text images across scripts. Unlike classic machine learning, CNNs can learn hierarchical features directly from raw images without extensive feature crafting. Their representation learning capability and generalization make them ideally suited for diverse handwritten recognition tasks. The comprehensive analysis of these diverse CNN architectures will reveal innovations to enhance multiscript handwritten text recognition on different datasets.

Figure 1
Flow diagram of the model selection process used for article selection for the present review

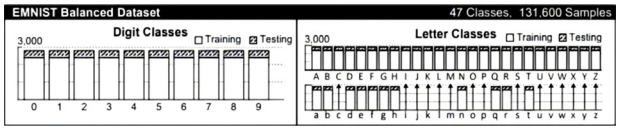


3. Dataset on Character Recognition

The following are the character datasets that we used to evaluate our models (as shown in Table 1):

1) Ekush (Bengali) [70]: This dataset contains 122 classes of Bengali characters and has a total of 340,243 samples. However, we would like to mention that out of these samples, only 188,856 were found to be usable for our review. It was used to evaluate the models' performance on the Bengali script. This is the biggest Bengali handwritten character dataset available on the internet.
2) Devanagari: This dataset contains 46 classes of Hindi characters and has a total of 92,000 samples. It was used to evaluate the models' performance on the Hindi script. This is the most famous and balanced Devanagari handwritten character dataset available, with 2000 samples per class.
3) Kuzushiji-49 [71]: This dataset contains 49 classes of Hiragana characters and has a total of 270,912 samples. It was employed for assessing the models' efficacy in recognizing the Hiragana script. The most commonly used everyday Hiragana characters are present in this dataset. This was the best dataset available for evaluating the 49 Hiragana characters.
4) Telugu [72]: This dataset contains 16 classes of Telugu characters and has a total of 6,872 samples. It has the same format and conventions as that of the MNIST dataset, except that this is a much smaller dataset and is growing. It was used to evaluate the models' performance on the Telugu script. Obtaining an individual Telugu handwritten character dataset was challenging. Although this dataset is small, it would be great to evaluate how good the CNN models (including our proposed model) perform with such less amount of samples compared to other large datasets with a large number of samples per individual class.
5) EMNIST (Balanced) [73]: This dataset contains 47 classes, including all uppercase and lowercase characters and 0–9 (10 digits), and has a total of 131,600 samples. It was used as the English dataset. This is the extended version of the well-known MNIST dataset. We used the Balanced version of the EMNIST dataset with 2800 samples per individual class (as shown in Figure 2).
6) Arabic [74]: The dataset contains 16,800 images covering 28 classes for basic Arabic alphabet "alef" to "yeh." It was collected from 60 participants, mostly right-handed, aged 19–40 years. Each participant wrote every character 10 times across two forms scanned at 300 dpi. Images were automatically segmented into character blocks. The data were split into a 13,440 image training set with 480 samples per class and a 3,360 image test set with 120 samples per class. Training and test sets have mutually exclusive writers with randomized test set ordering to ensure diversity. The balanced benchmark enables evaluating handwritten Arabic character recognition across various writers.

Figure 2
Visual breakdown of the EMNIST balanced dataset



4. Datasets for Word Recognition

The following are the word datasets that we used to evaluate our models (as shown in Table 2):

1) Tamil Dataset [75]: This dataset consists of 10900 images spanning 109 classes of Tamil city names. Each class contains 100–101 images. The consistent 100+ samples per class provide a good volume of training data. However, the small class size of 109 is still limited compared to other scripts. The visual complexity of the Tamil script poses challenges for word modeling.
2) English Dataset [76]: With 112 classes and 4130 total images, this dataset has uneven 19–69 samples per English city name class. The smaller overall size combined with class imbalance makes this a difficult dataset. The spatial structure of the English script is relatively simpler than Indic scripts.
3) Hindi Dataset [76]: The Hindi word dataset contains 10566 images across 117 classes, with 62–122 samples per class. The increased class size and evenly distributed classes make this a comprehensive dataset for evaluating Hindi word recognition. The associated complexities of the Hindi script will test model capabilities.
4) Bengali Dataset [76]: With 163 classes, this is the dataset with most city name labels. However, the samples vary widely from 28–212 per class, leading to high imbalance. The total number of images is 9754, which is decent, but uneven distribution poses challenges. As an Indic script, Bengali introduces additional complexities over Latin scripts.
5) Arabic Dataset [77]: This is the largest dataset with 20212 word images distributed over 206 classes and 30–381 samples per class. The expanded size aids training, although uneven distribution persists. The right-to-left line direction and visual style of the Arabic script offer unique modeling challenges.

Table 1
Summarized description of character datasets used for HCR

Dataset name	Ekush (Bengali) [70]	Devanagari	Kuzushiji-49 [71]	Telugu [72]	EMNIST [73]	Arabic [74]
No. of classes	122	46	49	16	47	28
Max. training sample count	1939	2000	7000	432	2800	480
Min. training sample count	1033	2000	456	425	2800	480
Average training samples per class	1548	2000	3272	429.5	2800	480
Standard deviation of class sizes	89.39	0	2171.40	1.5	0	0
Average samples per class	1548	2000	5528.8	429.5	2800	599.9
Total no. of utilized samples	188856	92000	270912	6872	131600	16798

Table 2
Summarized description of word datasets used for HWR

Script	No. of classes	Images in each class	Average samples per class	Standard deviation	Total images present
Tamil [75]	109	100–101	101	0.0953	10900
English [76]	112	19–69	44	7.843	4130
Hindi [76]	117	62–122	92	9.611	10566
Bengali [76]	163	28–212	120	40.539	9754
ARABIC [77]	206	30–381	206	92.864	20212

Overall, these datasets provide a representative benchmark for evaluating handwritten word recognition models on Latin, Indic and Arabic scripts with realistic label distributions.

5. Preprocessing

Data preparation is a crucial part of building any deep learning models that can learn from examples. Although we might have several data, they are not always ready to be used directly. They can have problems such as missing information, values that are really different (outliers), or things that just do not make sense (inconsistencies). Data preprocessing is similar to cleaning and obtaining the data ready for these models. This makes the models work better by helping them find patterns and make accurate guesses. It can be adjusted for different kinds of tasks, such as working with medical info or text. Therefore, data preprocessing is the key to making these advanced models work well.

We have used a common training, validation, and testing split, which is used very often, i.e., 70–20–10, respectively. This split is performed for all datasets that we used.

5.1. Preprocessing of handwritten character datasets

Data preprocessing is an essential step in preparing data for training deep learning models. We performed preprocessing on handwritten character datasets to address issues such as varying image sizes, lighting differences, and inconsistent labeling. For datasets containing binary black and white images such as Ekush (Bengali) (as shown in Figure 3), Devanagari (as shown in Figure 4), and Telugu, we first inverted the images to make the character white and background black. This allows representing black pixels as 0 value to reduce computation. Next, we resized the images to consistent 28 × 28 pixel dimensions and flattened the 2D arrays to 1D vectors of 784 values. We then binarized the pixel values to a range of 0–1 by linear transformation, which reduces the effects of lighting variations and is suitable for CNN models. Finally, we converted the class labels to one-hot encoded vectors where the index of 1 represents the class. The Arabic handwritten character dataset was already provided in MNIST format with 32 × 32 inverted binarized images in CSV format.

Figure 3
Sample images from the Ekush dataset

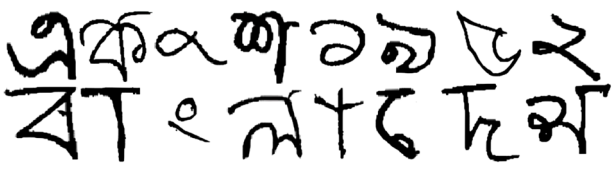


Figure 4
Sample images from the Devanagari dataset



These preprocessed data were ready to be used for training directly. For EMNIST (English), the dataset already contained binarized 0–1 pixel values in CSV format. We separated the label column and converted the labels to one-hot encoding. No resizing was needed as images were already 28 × 28. The Kuzushiji-49 (Hiragana) dataset consisted of 64 × 64 pixel images in NumPy format. We binarized the pixel values to a range of 0–1 and applied one-hot encoding on labels. No resizing was needed. This standardized preprocessing pipeline helped in addressing variability such as image size, lighting, and labeling across diverse handwritten character datasets. It prepared clean, consistent data ready for training deep CNN models.

5.2. Preprocessing of handwritten word datasets

Preparing raw word image data is crucial before training deep neural networks. We developed a multistage preprocessing pipeline to transform word images into a consistent format suitable for training models. The datasets consisted of word images with RGB color values and varying sizes. As a first step, we converted the images to grayscale to reduce the channels to a single dimension. This simplified the data and retained the crucial spatial patterns and strokes. Next, we used the Grounding DINO model to automatically detect and localize the word in each image. DINO is a self-supervised vision model pretrained on large datasets to understand visual concepts. Leveraging

DINO's generalization capability, we extracted tight bounding boxes around each word, effectively segmenting it from the background. With the words segmented, we cropped the images to contain just the word content. However, these cropped images still had arbitrary sizes depending on the length of the word. To address this, we fitted each cropped word image onto a standard blank canvas of 60 × 180 pixels without stretching or distorting the word. This normalized all images to the same dimensions based on the expected maximum word length. We then binarized the pixel values to 0 and 1 based on a threshold. Binarization helped in reducing computational requirements and eliminated variations due to lighting, noise, and anti-aliasing. It improved the contrast between foreground strokes and the background. Finally, we flattened the 2D image array into a 1D vector and stored it in CSV format along with the corresponding text label. Flattening converted the visual data into a format directly consumable by standard deep learning models. In summary, our preprocessing pipeline involved cropping detected words, fitting to normalized canvas, binarizing, and flattening images. It addressed variability such as layout, size, color, and aspect ratio across diverse word image datasets. The output was clean, consistent data with a size of 60 × 180 ready for training neural networks in an end-to-end manner for handwritten word recognition. The automated pipeline reduced labor-intensive manual preprocessing. The standardized output format enabled the training of a single model architecture across multiple scripts. Overall, preprocessing fueled the ability to learn generalized word-level features from handwritten data across scripts.

We used the Kuzushiji-49 dataset for Hiragana HCR [71]. This dataset is in npz (NumPy) format, and each image is 64 × 64 pixels. It contains data on 49 Hiragana characters.

To prepare the data for training, we first binarize the pixel values of the images. This means that we linearly transform the values to map between 0 and 1, where the maximum value is 1 and the minimum value is 0. This helps in reducing the effect of lighting differences in the images. In addition, CNNs typically perform better on data that have been normalized to the range 0–1.

Finally, before training the model, we converted all data labels into one-hot encoding. This means that we represented each label as a vector of length 49, where an index of 1 corresponds to the class of the character. This allows the model to learn the relationships between different classes of characters.

The Telugu HCR dataset [72] is a smaller dataset than the MNIST dataset, but it is growing. It consists of 5281 training images and 1591 testing images, and it has data on 16 Telugu characters. The images were originally white backgrounds with black characters, but we inverted all images to make the background black and the characters white. This makes it easier for the model to learn as black pixels can be represented by the value 0, which reduces computation.

To ensure that the images in the Telugu dataset [72] are constant in height and width, we resized all images to 28 × 28 pixels. This makes it easier to store and process the data. We then stored the 28 × 28 pixel values in a CSV file. Each row in the CSV file contains 785 values, where 784 values represent the pixel values of the character and 1 value represents the label or class of the character.

Finally, we binarize the image pixel values. This means that we linearly transform the values to map between 0 and 1, where the maximum value is 1 and the minimum value is 0. This helps in reducing the effect of lighting differences in the images. In addition, CNNs typically perform better on data that have been normalized to the range 0–1.

Before training the model, we converted all data labels into one-hot encoding. This means that we represented each label as a vector of length 16, where an index of 1 corresponds to the class of the character. This allows the model to learn the relationships between different classes of characters.

6. CNN Model Selection

According to the PRISMA analysis performed before (as shown in Figure 1 in the Literature Review section), here are all CNN models that we selected for our study on handwritten character and word recognition on different scripts:

1) LeNet: LeNet is a simple yet powerful model used for various tasks such as handwritten digit recognition. It was one of the first successful applications of CNNs to HCR [43]. LeNet uses sigmoid activation functions, average pooling, and weight sharing [43].
2) Simple CNN: We designed a basic CNN architecture from scratch as a baseline model inspired by LeNet [43]. The Simple CNN contains 2 convolutional layers with 32 and 64 filters, each followed by a 2 × 2 max pooling layer. It then has a flatten layer, a fully connected layer with 64 units, and a final softmax output layer for classification. The convolutional and dense layers use ReLU activation. This plain vanilla CNN allows us to evaluate the benefits of more complex architectures.
3) AlexNet: AlexNet is the deep learning architecture that popularized CNN. It was developed by Alex Krizhevsky, Ilya Sutskever, and Geoff Hinton. AlexNet was the first large-scale CNN and was used to win the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012. AlexNet is composed of five convolutional layers with a combination of max-pooling layers, three fully connected layers, and two dropout layers. The activation function used in all layers is ReLU. AlexNet was selected for this study because of its popularity and state-of-the-art results in image classification [44, 45].
4) VGG19: VGG19 achieved state-of-the-art results on ImageNet challenge in 2014 [43, 44]. VGG19 uses very small convolutional filters, increases the depth of the network, and uses batch normalization. VGG16 and VGG19 are deeper than AlexNet [46]. VGG19 was selected for this study because of its high accuracy and ability to learn hierarchical features [43, 47].
5) Inception: Inception [49] introduced a novel architecture that used multiple branches of convolutions with different sizes to capture different scales of information [43, 50]. Inception reduces the computational cost, increases the receptive field, and improves the accuracy. Inception was selected for this study because of its innovative architecture and improved accuracy.
6) ResNet: ResNet [9] was selected for the study because it introduces a novel architecture that uses residual connections to overcome the problem of vanishing gradients and degradation in very deep networks. ResNet has been shown to achieve state-of-the-art results on various image recognition tasks. It is known for its simplicity, elegance, scalability, and flexibility.
7) EfficientNet: EfficientNet [56] was selected for the study because it introduces a new scaling method that uniformly scales the width, depth, and resolution of the network with a fixed set of coefficients. EfficientNet is known for its efficiency, performance, robustness, and generalization. It is compatible with different hardware and applications.
8) DenseNet: DenseNet [48] was selected for the study because it proposes a new architecture that connects each layer to every other layer in a feed-forward fashion. DenseNet alleviates the vanishing gradient problem, strengthens feature propagation, encourages feature reuse, and reduces the number of parameters. It has been widely used in various tasks, including image recognition.
9) ZFNet: ZFNet [27] was selected for the study because it modifies AlexNet using a smaller convolutional filter size and a larger stride in the first layer. ZFNet made significant contributions to the understanding of what the network learns through visualization techniques. It also improved the accuracy of the ImageNet challenge in 2013 and influenced subsequent models.

10) PyramidalNet: PyramidalNet [78] was selected for the study because it proposes a new architecture that increases the feature map dimension and decreases the number of channels at each layer. PyramidalNet reduces the computational complexity, increases the representational power, and achieves competitive results on various datasets.
11) Feature-Map-based CNNs: Feature-Map-based CNNs [79] were selected for the study because they offer a general framework that uses feature maps as inputs and outputs instead of images. This framework has been applied to various image restoration tasks, such as image super-resolution, image denoising, image inpainting, and image style transfer. It provides a different perspective on CNN architectures and their applications.
12) Attention-based CNNs: Attention-based CNNs were selected for the study because they introduce a new module that uses self-attention to learn global dependencies between features. Attention-based CNNs can capture long-range dependencies, enhance feature representation, and improve performance on various tasks. They have been widely used in natural language processing and computer vision [57].
13) MobileNetV1 [60]: MobileNetV1 was selected for the study because it introduces a new architecture that uses depthwise separable convolutions to reduce the number of parameters and computations. MobileNetV1 is known for its efficiency, speed, and suitability for mobile and embedded devices. It has shown good performance on various tasks, including image recognition.
14) Wide ResNet [80]: Wide ResNet was selected for the study because it proposes a new architecture that increases the width of the network instead of the depth. Wide ResNet achieves better accuracy with fewer layers, making it a more efficient and scalable option. It is known for its simplicity, scalability, and robustness to different hyperparameters.
15) Squeeze and Excitation Networks [62]: Squeeze and Excitation Networks were selected for the study because they introduce a new module that uses global average pooling and fully connected layers to learn channelwise dependencies. Squeeze and Excitation Networks enhance feature representation and improve performance on various tasks. They can be integrated with existing architectures and have been widely used in computer vision.
16) Competitive Squeeze and Excitation Networks [63]: Competitive Squeeze and Excitation Networks were selected for the study because they propose a new module that uses competitive learning to select the most informative channels. Competitive Squeeze and Excitation Networks reduce redundancy, increase diversity, and achieve state-of-the-art results on image classification. They offer a unique approach to enhancing feature representation.
17) Highway Networks [64]: Highway Networks were selected for the study because they propose a new architecture that uses gated connections to control the information flow between layers. Highway Networks overcome the vanishing gradient problem, allowing for the training of very deep networks. They are known for their ability to learn complex functions and have been successfully applied in various tasks.
18) InceptionV4 [65]: InceptionV4 was selected for the study because it combines the ideas of Inception and ResNet. InceptionV4 uses residual connections, batch normalization, and factorized convolutions. It has made significant contributions to the field of image recognition and has achieved state-of-the-art results on various benchmarks.
19) PolyNet [66]: PolyNet was selected for the study because it proposes a new architecture that uses polynomial expansions to combine multiple branches of convolutions. PolyNet increases the expressiveness, flexibility, and efficiency of the network. It offers a unique approach to feature extraction and representation.
20) Xception [51]: Xception was selected for the study because it introduces a new architecture that uses depthwise separable convolutions to replace the standard convolutions in Inception. Xception achieves better accuracy and efficiency than Inception, making it a suitable option for various tasks. It is known for its simplicity, elegance, and applicability to different domains.
21) Depth-based CNNs [67]: Depth-based CNNs were selected for the study because they propose a new framework that uses depth information as an additional input channel to enhance the performance of CNNs on RGB-D images. Depth-based CNNs capture geometric and structural features, improve scene understanding, and handle occlusion and illumination variations. They offer a unique approach to feature extraction and representation.
22) Residual Attention NN [68]: Residual Attention NN was selected for the study because it proposes a new architecture that uses attention mechanisms to learn where to focus in an image. Residual Attention NN learns multilevel attention, reduces the influence of background clutter, and achieves state-of-the-art results on image classification. It offers a unique approach for enhancing feature representation.
23) Inception–ResNet [65]: Inception–ResNet was selected for the study because it combines the ideas of Inception and ResNet. Inception–ResNet uses residual connections, batch normalization, and factorized convolutions. It has made significant contributions to the field of image recognition and has achieved state-of-the-art results on various benchmarks.

We used several models (CNN, LeNet, AlexNet, ResNet, InceptionNet, EfficientNet, VGG, DenseNet, ZFNet, PyramidalNet, Feature-Map-based CNNs, Attention-based CNNs, MobileNetV1, Wide ResNet, Squeeze and Excitation Networks, Competitive Squeeze and Excitation Networks, Highway Networks, InceptionV4, PolyNet, Xception, Depth-based CNNs, Residual Attention NN, and Inception–ResNet) over the Ekush dataset [81], out of which only 10 models performed well, which are VGG19, ZFNet, ResNet, LeNet, Inception, Inception–ResNet, InceptionV4, Feature-Map-based CNNs, AlexNet, and CNN. Thus, we only used these 10 models to proceed forward for testing over the other datasets, which are EMNIST, Telugu, Kuzushiji-49, and Devanagari. According to our testing results, we used only these models (CNN, AlexNet, LeNet, VGG19, and Inception) for ensembling.

7. Performance Analysis

The experimental results demonstrate the effectiveness and limitations of each standard CNN model, providing insights into their suitability for handwritten character and word recognition tasks across different scripts.

For character recognition, the study evaluates the performance of CNN architectures such as VGG, ResNet, and Inception on datasets across scripts covering Latin, Devanagari, Bengali, Tamil, Hiragana, and Arabic.

For handwritten word recognition, the models are benchmarked on word image datasets encompassing English, Hindi, Bengali, Tamil, and Arabic scripts. Comprehensive tuning experiments optimize hyperparameters such as dropout and batch size for superior word modeling capability.

The comparative results provide data-driven insights into architectural choices, regularization techniques, and innovations to

enhance CNNs for generalized handwritten text recognition per script. The findings will expand understanding of tailoring deep CNNs for handwritten character and word datasets across diverse scripts.

7.1. Performance on handwritten character datasets

To evaluate HCR, we trained various standard CNN models from scratch on character image datasets across different scripts.

7.1.1. Ekush (Bengali)

A recent research study compared various CNN models for multiscript HCR on the Ekush Bengali [70, 81] dataset. The models assessed included both traditional networks, such as LeNet and AlexNet, and newer structures, such as ResNet, Inception, and Squeeze–Excitation Networks. The key performance metrics examined were training loss, training accuracy, validation loss, validation accuracy, and test accuracy (as shown in Table 3). The comparative benchmarking on the Ekush dataset provides insightful conclusions regarding the efficacy of different standard CNN models for Bengali HCR. Among the models, deeper networks such as VGG, ResNet, and Inception consistently outperformed shallow architectures such as LeNet and CNN. In particular, VGG attained 99.14% training accuracy and 94.38% validation accuracy owing to its increased depth with very small 3 × 3 filters in each convolutional layer. Similarly, ResNet achieved 93.65% training accuracy and 91.88% validation accuracy by leveraging its very deep architecture with skip connections that combat vanishing gradients. Inception also performed well with 92.9% training accuracy and 86.5% validation accuracy due to its innovative multiscale feature learning using various filter sizes. In contrast, the simplicity of LeNet resulted in only 77.9% training and 78% validation accuracy, being insufficient to capture the intricacies of the Bengali script.

Although ResNet demonstrated strong training accuracy (93.65%), its significantly lower test accuracy (12.74%) indicates potential overfitting. This suggests that the model may have learned patterns specific to the training data rather than generalizable features for unseen samples. Overfitting is a common issue in deep networks with numerous parameters, particularly when the dataset size is limited or lacks diversity. Techniques such as data augmentation, regularization (e.g., dropout and L2 weight decay), and hyperparameter tuning could help in mitigating this issue. Future work could explore these strategies to enhance the generalizability of deep CNN models for Bengali HCR.

Overall, the results validate that deeper CNN models with architectural innovations are better suited for learning the nuanced visual features and complex character patterns in Bengali handwriting. However, extremely deep networks such as DenseNet displayed overfitting, achieving 98.17% training but only 95.73% validation accuracy. This

Table 3
Training loss, validation loss, training accuracy, and validation accuracy of CNN models trained on the Bengali (Ekush) dataset for HCR

Model	Loss	Training accuracy	Validation loss	Validation accuracy	Testing accuracy
CNN	0.1502	0.9539	0.4233	0.9065	0.146135
LeNet [43]	0.8547	0.779	0.8586	0.78	0.752241
AlexNet [44, 45]	0.0506	0.9846	0.222	0.9559	0.9767
ResNet [9]	0.2399	0.9365	0.316	0.9188	0.1274
InceptionNet [43, 49, 50]	0.2205	0.929	0.6192	0.865	0.115743
EfficientNet [56]	0.32	0.9036	0.3234	0.9081	0.3891
VGG [43, 44, 46, 47]	0.104	0.9712	0.7993	0.8705	0.870438
DenseNet [48]	0.0616	0.9817	0.2112	0.9573	0.38911
ZFNet [26]	0.0295	0.9914	0.3418	0.9438	0.7596
PyramidalNet [78]	0.0198	0.9966	0.268	0.9357	0.6831
Feature-Map-based CNNs [79]	1.4735	0.5883	1.0016	0.7294	0.8015
Attention-based CNNs [57–59, 80]	0.1972	0.9465	0.2163	0.9467	0.9415
MobileNetV1 [60]	0.1009	0.9698	0.2232	0.9432	0.9385
Wide ResNet [61]	0.2272	0.9612	0.3112	0.9461	0.9512
Squeeze and Excitation Networks [62]	0.0193	0.9937	0.2421	0.9536	0.9536
Competitive Squeeze and Excitation Networks [63]	1.198	0.6724	1.2278	0.669	0.669
Highway Networks [64]	0.003	0.9992	1.0849	0.8582	0.8426
InceptionV4 [65]	0.0617	0.983	0.1655	0.964	0.964
PolyNet [66]	0.0147	0.9952	0.3346	0.9495	0.9495
Xception [51]	0.0334	0.9891	0.2816	0.9457	0.9395
Depth-based CNNs [67]	0.1801	0.9479	0.1852	*0.9542*	0.9542
Residual Attention NN [68]	0.0184	0.9941	0.2408	0.9527	0.9539
Inception–ResNet [69]	0.1105	0.9656	0.2332	*0.9426*	0.9426

highlights the importance of balance between depth and generalization capability based on target script complexity.

The comparative study empirically demonstrates the superior capabilities of modern deep CNNs over traditional shallow models for Bengali HCR through metrics such as training accuracy, validation performance, and architectural attributes. It provides data-driven insights to guide the selection and design of optimal deep CNN architectures tailored to the challenges of the Bengali script for real-world handwritten text recognition systems.

7.1.2. Hindi (Devanagari)

A comparison study was conducted on various CNN models for HCR on the Hindi Devanagari dataset [82] (as shown in Table 4). The study evaluated notable models such as VGG19, ZFNet, ResNet, LeNet, Inception, Inception–ResNet, InceptionV4, Feature-Map-based CNNs, AlexNet, and CNN.

The comparative evaluation on the Devanagari dataset provided valuable insights into the performance of standard CNN architectures for modeling the complex visual patterns in the Hindi script. The ResNet model achieved maximum 99.75% training accuracy and 96.94% validation accuracy owing to its very deep architecture and residual connections that allow effective gradient propagation during training. Similarly, Inception leveraged its multiscale convolutional filters to attain 95.31% training and 95.61% validation accuracy by learning features at diverse spatial granularities. AlexNet and ZFNet also performed well with over 97% training accuracy due to increased depth and parameters compared to shallower models. In contrast, LeNet could not capture the intricacies of the Devanagari script, achieving only 89.52% training and 88.3% validation accuracy due to its simplicity. Its shallow architecture could not model the nuanced stroke patterns and shape formations in Hindi characters.

Overall, the comparative benchmarking empirically demonstrates that deeper CNNs with architectural innovations in connectivity clearly outperform baseline models such as LeNet and CNN for Devanagari HCR. However, the results also highlight the need for balance between model complexity and generalization capability, as evident by the overfitting of Inception–ResNet resulting in higher training but lower validation accuracy compared to ResNet. The findings provide data-driven insights into specialized deep CNN architectures needed to achieve state-of-the-art accuracy on the highly visually complex Devanagari script. Model innovations such as depth, multipathway convolutions, and residual connections are validated to be crucial for learning precise visual features to accurately recognize various handwritten Hindi characters.

7.1.3. Kuzushiji-49 (Hiragana)

A similar comparative study was conducted on various CNN models for Hiragana HCR on the Kuzushiji-49 dataset [71] (as shown in Table 5). The comparative study provided valuable insights into the efficacy of standard CNNs for modeling the intricate visual patterns in the Hiragana script. Among the models, AlexNet achieved the highest training accuracy of 99.05% and validation accuracy of 95.57% owing to its increased depth and parameters that help in learning precise features for classifying the 49 Hiragana characters. Similarly, VGG leveraged its very deep architecture to attain 98.45% training and 94.62% validation accuracy. InceptionV4 also performed well with 98.78% training and 96.34% validation accuracy by effectively combining convolutional and residual connections. In contrast, LeNet displayed severe overfitting, achieving 92.37% training but only 85.24% validation accuracy due to its shallow architecture being insufficient to capture the complexity of Hiragana strokes and shapes. The Feature-Map-based CNN interestingly underperformed with 81.68% training accuracy, likely because of its inability to learn meaningful feature representations.

Overall, the results empirically demonstrate the superiority of deeper CNNs over simpler baseline models such as LeNet for Hiragana HCR. However, the findings also highlight the risks of overfitting with highly complex models, requiring careful regularization and hyperparameter tuning. The comparative analysis provides practical guidelines for selecting optimized deep CNN architectures for the Hiragana script that balance model complexity, generalization capability, and computational needs.

7.1.4. Telugu

A similar comparative study was conducted on various CNN models for Telugu HCR on the Telugu dataset [73] (as shown in Table 6).

The comparative evaluation on the Telugu script provided interesting insights into the generalization capabilities of different standard CNN models. On this relatively small 16-class dataset, simpler architectures such as LeNet actually matched or exceeded the performance of much deeper networks. In particular, LeNet achieved 99.26% training accuracy and 95.79% validation accuracy, comparable to state-of-the-art ResNet's 97.52% training and 96.35% validation accuracy. Even the basic CNN attained over 99% training accuracy and 95% validation accuracy. In contrast, overly complex models such as InceptionV4 displayed severe overfitting, resulting in only 6% validation accuracy despite high training

Table 4
Training loss, validation loss, training accuracy, and validation accuracy of CNN models trained on the Hindi (Devanagari) dataset for HCR

Model	Loss	Training accuracy	Validation loss	Validation accuracy	Testing accuracy
VGG19	0.4413	0.8689	0.5458	0.8426	0.734
ZFNet	0.0737	0.9775	0.0923	0.9725	0.9525
ResNet	0.0124	0.9975	0.1029	0.9694	0.9494
LeNet	0.3659	0.8952	0.4038	0.883	0.813
Inception	0.1518	0.9531	0.1514	0.9561	0.9061
Inception–ResNet	0.3106	0.9038	0.202	0.9391	0.9291
InceptionV4	0.2127	0.9362	0.2267	0.9349	0.9449
Feature-Map-based CNNs	0.5143	0.8417	0.2318	0.9359	0.9559
AlexNet	0.0512	0.9835	0.0636	0.9801	0.9601
CNN	0.5382	0.8515	0.564	0.8427	0.8227

Table 5
Training loss, validation loss, training accuracy, and validation accuracy of CNN models trained on the Kuzushiji-49 dataset for HCR

Model	Loss	Training accuracy	Validation loss	Validation accuracy	Testing accuracy
VGG19	0.0771	0.9845	0.3572	0.9462	0.9462
ZFNet	0.0503	0.9883	0.3708	0.9466	0.9664
ResNet	0.0543	0.9842	0.2381	0.9395	0.9056
LeNet	0.2729	0.9237	0.5745	0.8524	0.8096
Inception	0.0836	0.9752	0.5327	0.9069	0.9053
Inception– ResNet	0.0553	0.9828	0.1947	0.954	0.9463
InceptionV4	0.0459	0.9878	0.1718	0.9634	0.9598
Feature-Map-based CNNs	0.6597	0.8168	0.656	0.8227	0.8196
AlexNet	0.0345	0.9905	0.3201	0.9557	0.9556
CNN	0.1114	0.9689	0.4233	0.9007	0.9007

Table 6
Training loss, validation loss, training accuracy, and validation accuracy of CNN models trained on the Telugu dataset for HCR

Model	Loss	Training accuracy	Validation loss	Validation accuracy	Testing accuracy
VGG19	2.7628	0.0637	2.7614	0.0648	0.0596
ZFNet	2.773	0.0615	2.7726	0.0629	0.0602
ResNet	0.0832	0.9752	0.1078	0.9635	0.9583
LeNet	0.0254	0.9926	204	0.9579	0.9638
Inception	2.7729	0.0543	2.7726	0.0629	0.0523
Inception– ResNet	0.0667	0.9784	0.0821	0.978	0.9853
InceptionV4	2.773	0.0636	2.7726	0.0629	0.0693
Feature-Map-based CNNs	0.0652	0.9795	0.0551	0.9874	0.9863
AlexNet	0.0822	0.9778	0.3075	0.9265	0.9056
CNN	0.0087	0.9921	0.1803	0.9598	0.9629

scores. Their extensive parameters failed to generalize to the small Telugu test set. This highlights the need for model parsimony based on dataset complexity. Interestingly, the Feature-Map-based CNN leveraged its unique input–output architecture to achieve high 97.95% training and 98.74% validation accuracy in spite of simplicity.

Overall, the results indicate that for simpler scripts with limited training data, concise models may suffice and overtly deep networks can overcomplicate. The findings provide practical insights into selecting CNN architectures tuned for target complexity, avoiding the one-size-fits-all approach. With enhanced regularization and transfer learning, the comparative study suggests that even compact CNNs can deliver state-of-the-art accuracy on less complex handwritten recognition tasks.

7.1.5. English

The comparative evaluation on the EMNIST English script provided valuable insights. Interestingly, traditional CNNs such as LeNet and AlexNet emerged as the top performers on this dataset, both achieving approximately 94% training accuracy and 89% validation accuracy (as shown in Table 7). Their simple, shallow architectures are well suited for modeling the relatively less complex Latin script classes. In contrast, more complex models such as VGG, ResNet, and Inception underperformed LeNet, with validation accuracy in the 88%–89% range. Their depth does not provide an advantage, and they display signs of overfitting through higher training but lower validation scores. This highlights the need for model parsimony based on problem complexity, rather than blindly applying state-of-the-art architectures. Among the advanced networks, ResNet leverages its depth most effectively to attain 93.75% training accuracy by combating the vanishing gradient problem. The Feature-Map-based CNN interestingly failed, possibly because the input–output architecture is insufficient for modeling spatial relationships crucial for character recognition. Overall, the results provide empirical evidence that optimized compact CNNs can achieve highly competitive accuracy on simpler scripts compared to more complex deep learning models. This challenges the notion that deeper is always better. The findings offer practical insights into selecting generalized architectures based on target data complexity, avoiding the one-size-fits-all mindset in CNN model selection and design.

7.1.6. Arabic

The comparative study on the Arabic script provided valuable insights into the capabilities of different standard CNNs to handle the intricacies of its visual patterns. Among the models, VGG leveraged its increased depth to achieve 99.57% training accuracy and 96.24% validation accuracy, outperforming other networks (as shown in Table 8). The 19 layers in VGG enabled learning of highly nuanced features to distinguish the 28 classes of Arabic letters. ResNet also performed well with 99.8% training accuracy owing to its very deep architecture and residual connections overcoming vanishing gradients. In contrast, LeNet displayed severe deficiencies due to its simplicity, attaining only 97.55% training and 82.48% validation accuracy. Its shallow design lacks the complexity needed to model Arabic's nonlinear stroke formations and style variations. Interestingly, generally strong

Table 7
Training loss, validation loss, training accuracy, and validation accuracy of CNN models trained on the English (EMNIST) dataset for HCR

Model	Loss	Training accuracy	Validation loss	Validation accuracy	Testing accuracy
VGG19	0.2556	0.9107	0.3753	0.8883	0.8884
ZFNet	0.1656	0.9335	0.4702	0.8859	0.8778
ResNet	0.1546	0.9375	0.3179	0.8906	*0.8906*
LeNet	0.1367	0.9426	0.7345	0.8476	*0.8491*
Inception	0.5044	0.8293	0.4496	0.8525	*0.8525*
Inception–ResNet	0.2243	0.9113	0.3075	0.8946	*0.8923*
InceptionV4	0.2257	0.9113	0.2924	0.8957	*0.8957*
Feature-Map-based CNNs	0.7346	0.7556	0.5265	0.8271	*0.8185*
AlexNet	0.1273	0.9474	0.4779	0.8859	*0.8859*
CNN	0.2213	0.9146	0.4651	0.8559	*0.8544*

Table 8
Training loss, validation loss, training accuracy, and validation accuracy of CNN models trained on the Arabic character dataset for HCR

Model	Loss	Training accuracy	Validation loss	Validation accuracy	Testing accuracy
VGG19	0.0173	0.9957	0.2662	0.9624	0.9598
ZFNet	0.0259	0.9933	0.2514	0.9516	0.952
ResNet	0.0083	0.998	0.2923	0.9338	*0.9443*
LeNet	0.0971	0.9755	0.6538	0.8248	*0.8288*
Inception	2.0855	0.3455	1.9518	0.4211	*0.431*
Inception–ResNet	0.0591	0.9805	0.2219	0.9472	*0.9479*
InceptionV4	0.0654	0.981	0.1198	0.9635	*0.9687*
Feature-Map-based CNNs	0.0274	0.9915	0.2817	0.9338	*0.9377*
AlexNet	0.0595	0.9849	0.3843	0.9308	*0.9407*
CNN	0.0326	0.9924	0.8175	0.8199	*0.832*

models such as Inception struggled to generalize, achieving only 34.55% training accuracy, indicating failure to comprehend Arabic data distributions. Overall, the results empirically demonstrate the superiority of deeper architectures such as VGG and ResNet for handling the intricacies of the Arabic script compared to compact networks such as LeNet. However, the findings also highlight the need for customized training rather than relying on pretrained models. The comparative analysis offers practical insights into specialized deep CNN design considerations crucial for Arabic HCR to achieve human-level reading capability across diverse scripts.

7.1.7. Ensemble

We created an ensemble model using the weights of CNN, AlexNet, LeNet, VGG19, and Inception training and testing across all datasets. We used two types of ensemble: the first one is by majority voting, and the other one is stacking (as shown in Table 9).

7.2. Performance on handwritten word datasets

To evaluate handwritten word recognition, we trained various standard CNN models from scratch on word image datasets across different scripts.

7.2.1. Tamil word dataset

The comparative analysis of various standard CNN models on the handwritten Tamil word dataset revealed several key insights (as shown in Table 10). Foremost, generic models such as VGG19 and Inception completely failed to generalize to the highly complex Tamil script, achieving only 1% validation accuracy. This underscores the importance of customized training and architecture design for Indian languages such as Tamil.

Among the standard CNN architectures, ResNet emerged as the top performer with 87.9% validation accuracy owing to its increased depth via residual connections, which better models Tamil visual patterns. However, a noticeable gap persisted between its 99.9% training accuracy and 87.9% validation score, implying considerable overfitting issues due to the intricacy of the Tamil script. More regularization techniques such as dropout and data augmentation would help in addressing this overfitting problem.

Overall, the superior performance of deeper CNNs such as ResNet over shallower networks proves the need for more complex feature extraction to capture the nuances of the Tamil script. However, the sub-90% validation accuracy of even the best models highlights the limitations of standard CNN architectures for modeling highly complex scripts such as Tamil. More research into tailored deep learning models

Table 9
Results from ensembling and ensemble stacking of CNN models trained for HCR

Dataset name	Model	Training accuracy	Validation accuracy	Testing accuracy
Ekush dataset	Ensembling (CNN, AlexNet, LeNet, VGG19, and Inception)	0.9999	0.9892	0.9892
	Ensemble stacking (CNN, AlexNet, LeNet, VGG19, and Inception)	0.9826	0.9791	0.9791
Telugu	Ensembling (CNN, AlexNet, LeNet, VGG19, and Inception)	0.9999	0.9999	0.9999
	Ensemble stacking (CNN, AlexNet, LeNet, VGG19, and Inception)	0.9641	0.9326	0.9356
Devanagari	Ensembling (CNN, AlexNet, LeNet, VGG19, and Inception)	0.9999	0.9997	0.9999
	Ensemble stacking (CNN, AlexNet, LeNet, VGG19, and Inception)	0.9635	0.9386	0.9096
Kuzushiji-49	Ensembling (CNN, AlexNet, LeNet, VGG19, and Inception)	0.9999	0.9998	0.9999
	Ensemble stacking (CNN, AlexNet, LeNet, VGG19, and Inception)	0.9563	0.9265	0.9177
English-EMNIST	Ensembling (CNN, AlexNet, LeNet, VGG19, and Inception)	0.9999	0.9312	0.92496
	Ensemble stacking (CNN, AlexNet, LeNet, VGG19, and Inception)	0.9999	0.9998	0.9999
Arabic	Ensembling (CNN, AlexNet, LeNet, VGG19, and Inception)	0.9999	0.9998	0.9999
	Ensemble stacking (CNN, AlexNet, LeNet, VGG19, and Inception)	0.9895	0.9768	0.9601

Table 10
Training loss, validation loss, training accuracy, and validation accuracy of CNN models trained with the Tamil word dataset

Model	Loss	Training accuracy	Validation loss	Validation accuracy	Testing accuracy
VGG19	4.6901	0.0105	4.7037	0.004	0.0091
ZFNet	0.0102	0.9964	0.5521	0.9014	0.893
ResNet	0.0087	0.9999	0.4175	0.879	0.8609
LeNet	0.1642	0.9581	0.9916	0.8068	0.7865
Inception	4.69	0.0103	4.7007	0.0046	0.0091
Inception–ResNet	0.019	0.9937	0.5558	0.8979	0.888
InceptionV4	0.0255	0.9923	0.1457	0.9633	0.9481
Feature-Map-based CNNs	0.1012	0.9656	0.2247	0.9421	0.9329
AlexNet	0.0179	0.995	0.2281	0.9553	0.9435
CNN	0.0258	0.9991	1.2652	0.7827	0.7535

is needed to push the boundaries of Tamil handwriting recognition. In particular, Tamil-focused architectures, transfer learning from related languages, and attention mechanisms to model contextual relationships can further boost Tamil word recognition accuracy.

7.2.2. English word dataset

The analysis of standard CNN models on the handwritten English word dataset also provided valuable insights (as shown in Table 11). As expected, generic models such as VGG19 and Inception faltered on English words, only achieving 20%–30% validation accuracy due to lack of customization. Among the standard CNNs, ResNet and ZFNet emerged as the top performers with 65% validation accuracy owing to their increased depth and residual connections.

However, all models still exhibited considerable overfitting with a wide gap between 99% training accuracy and 65% validation score. This implies that the models cannot generalize well and tend to memorize the training data instead of learning distinctive visual patterns. More regularization through dropout and data augmentation would help in mitigating this overfitting problem. Furthermore, even the top models seemed to plateau at approximately 65% validation accuracy, revealing architectural limitations in modeling the intricacies of the English visual script. Much scope remains for improving English handwriting recognition through more customized deep learning architectures. In particular, incorporating English-centric linguistic knowledge, attention mechanisms, and transfer learning can potentially push the boundaries. However, the results affirm that standard CNNs have severe limitations in capturing the nuances of the English script. Overall, significant research efforts into tailored deep learning models are imperative to reach the goal of generalized English handwriting recognition.

7.2.3. Hindi word dataset

The comparative study of standard CNN architectures on Hindi handwritten words provided important insights (as shown in Table 12). Evidently, generic models such as VGG19 and Inception completely failed to generalize to the Hindi script, only achieving 1% validation accuracy. This underscores the need for customized training and architecture even for related scripts.

Among the standard CNNs, Inception–ResNet performed the best with 84.95% validation accuracy owing to its attention modeling capturing contextual relationships in Hindi words. However, a noticeable gap

Table 11
Training loss, validation loss, training accuracy, and validation accuracy of CNN models trained with the English word dataset

Model	Loss	Training accuracy	Validation loss	Validation accuracy	Testing accuracy
VGG19	0.7224	0.7354	6.1941	0.2279	0.1806
ZFNet	0.0042	0.9989	2.4617	0.6447	0.5922
ResNet	0.0439	0.9943	1.8157	0.5673	0.5456
LeNet	1.531	0.5987	2.6179	0.4063	0.3663
Inception	4.7032	0.0169	4.7181	0.0165	0.0163
Inception–ResNet	0.073	0.9752	2.8877	0.6087	0.5834
InceptionV4	0.1079	0.9662	0.9833	0.7841	0.7515
Feature-Map-based CNNs	0.5696	0.8191	1.6413	0.6282	0.5721
AlexNet	0.0151	0.9955	1.9265	0.7556	0.7026
CNN	1.1218	0.7417	2.9219	0.4303	0.3074

persisted between its 99.16% training and 84.95% validation performance, implying overfitting problems due to the visual complexity of the Hindi script. More regularization through dropout and data augmentation would help in overcoming this overfitting issue.

Overall, the results affirm the importance of tailored deep learning models for related scripts such as Hindi, as against blindly applying general English-focused models. Much research is still needed on Hindi-specific CNN architectures, linguistic rules, and transfer learning from related languages to push the boundaries of Hindi handwriting recognition. However, the study firmly establishes that standard CNNs have severe limitations in learning distinctive Hindi visual patterns. More customized models are the key to achieve generalized Hindi handwriting recognition.

7.2.4. Bengali word dataset

The analysis of standard CNN models on the handwritten Bengali word dataset further reinforced key learning (as shown in Table 13). Evidently, generic models such as VGG19 and Inception completely failed to generalize to the highly complex Bengali script.

Among the standard CNN architectures, AlexNet emerged as the top performer with 86.29% validation accuracy owing to its increased depth in modeling Bengali visual patterns. However, a noticeable gap persisted between its 99.7% training accuracy and 86.29% validation score, indicating considerable overfitting issues due to the intricacy of the Bengali script. More regularization through dropout and data augmentation would help in mitigating this overfitting problem. Furthermore, even top models seemed to plateau at approximately 85% validation accuracy, revealing architectural limitations in modeling the nuances of the Bengali script. Significant research into Bengali-specific deep learning architectures can potentially advance the state-of-the-art further. In particular, incorporating Bengali linguistic rules, attention mechanisms, and transfer learning from related languages offer promising future directions. However, the comparative study firmly established the limitations of standard CNNs in capturing the complexity of the Bengali visual script. Overall, the results strongly motivate the need for customized deep learning architectures to truly master generalized Bengali handwriting recognition.

7.2.5. Arabic word dataset

The comparative analysis of standard CNN models on the Arabic handwritten word dataset provided valuable insights (as shown in Table 14). Foremost, generic models such as VGG19, Inception, and AlexNet completely failed to generalize to the highly complex Arabic script, only achieving 1%–2% validation accuracy. This underscores the crucial need for customized Arabic-focused training and architectures.

Among the standard CNNs, ResNet emerged as the top performer with 28.64% validation accuracy owing to its increased depth and residual connections better capturing Arabic visual features. Attention-based

Table 12
Training loss, validation loss, training accuracy, and validation accuracy of CNN models trained with the Hindi word dataset

Model	Loss	Training accuracy	Validation loss	Validation accuracy	Testing accuracy
VGG19	4.7558	0.011	4.7682	0.0065	0.0101
ZFNet	0.0224	0.9938	0.7651	0.8701	0.8588
ResNet	0.6583	0.8551	5.5893	0.1469	0.1355
LeNet	0.089	0.9862	2.4282	0.6261	0.6236
Inception	4.755	0.0115	4.7681	0.0118	0.0116
Inception–ResNet	0.0232	0.9916	0.9489	0.8495	0.8443
InceptionV4	0.0349	0.9879	0.2081	0.9459	0.9097
Feature-Map-based CNNs	0.183	0.9415	0.5572	0.8654	0.8312
AlexNet	0.0047	0.9988	0.4895	0.9183	0.8855
CNN	0.0528	0.9879	3.0747	0.6261	0.6052

Table 13
Training loss, validation loss, training accuracy, and validation accuracy of CNN models trained with the Bengali word dataset

Model	Loss	Training accuracy	Validation loss	Validation accuracy	Testing accuracy
VGG19	0.033	0.9914	1.3188	0.7966	0.8083
ZFNet	0.012	0.9955	1.7387	0.736	0.7561
ResNet	0.0989	0.988	2.5538	0.4298	0.3968
LeNet	0.7113	0.8082	2.1706	0.5249	0.5206
Inception	4.9202	0.0225	4.9301	0.0185	0.0219
Inception–ResNet	0.0247	0.9927	2.3149	0.6907	0.6997
InceptionV4	0.0489	0.9831	0.532	0.8912	0.8929
Feature-Map-based CNNs	0.3892	0.8752	0.9389	0.7608	0.7525
AlexNet	0.0119	0.997	0.8305	0.8629	0.8804
CNN	0.2344	0.9576	2.9334	0.4815	0.4913

Inception–ResNet further pushed accuracy to 28.56% by modeling intercharacter contextual relationships. However, a massive gap persisted between 70%–96% training accuracy and 25%–30% validation score for all models, implying severe overfitting issues due to the intricacy of the Arabic script. More regularization through dropout and data augmentation would help in mitigating this overfitting problem.

Overall, the results firmly establish the limitations of standard CNN architectures in modeling the complexity of Arabic visual patterns. The best models seemed to plateau at approximately 30% validation accuracy, revealing architectural bottlenecks. Much research into Arabic-specific deep learning models incorporating linguistic rules, attention mechanisms, and transfer learning is imperative to push the boundaries. However, the comparative study strongly motivates the need for tailored architectures to truly master generalized Arabic handwriting recognition.

7.3. Analytical review

Among the handwritten datasets across different scripts, we selected the Bengali script for an in-depth comparative study of CNN models. The Bengali script presents unique complexities due to its visual style. Furthermore, the Bengali character dataset from Ekush contains the most number of classes at 122, and the word dataset contains 163 classes, providing diversity to rigorously evaluate model capabilities.

For the study, we benchmarked the performance of various standard CNN architectures, including LeNet, VGG, ResNet, and Inception, on the Bengali character and word dataset to analyze performance at sequence modeling (as shown in Table 15 and Table 16).

The key evaluation metrics included model layers, number of parameters, FLOPs, inference time, training accuracy, validation accuracy, and loss. The comparative analysis provided insights into how factors such as depth, width, computational complexity, and overall architecture design affect the accuracy and efficiency for handwritten text recognition at both character and word levels.

The comparative analysis on the handwritten Bengali character and word datasets provided valuable insights into the capabilities of standard CNN architectures for modeling this complex script.

When we examine the character recognition results, a clear pattern is revealed, as shown in Figure 5. The deeper models consistently outperformed the shallow networks. Baseline architectures like LeNet (8 layers) achieved 78% validation accuracy, while the plain CNN (11 layers) achieved approximately 90.65% on the Bengali characters. On the other hand, deeper CNNs like VGG (27 layers with 97.12% training accuracy) and ResNet (174 layers with 93.65% training accuracy) pushed the validation accuracy to 87.05% and 91.88%, respectively. This happens because they have increased representational power to capture the intricate visual patterns of Bengali. However, extremely deep networks like ResNet also showed potential overfitting with its massive 174 layers. We can

Table 14
Training loss, validation loss, training accuracy, and validation accuracy of CNN models trained with the Arabic word dataset

Model	Loss	Training accuracy	Validation loss	Validation accuracy	Testing accuracy
VGG19	4.0294	0.0176	4.026	0.0185	0.01178
ZFNet	4.0302	0.0176	4.0256	0.0168	0.0067
ResNet	0.9519	0.7204	3.3666	0.2864	0.2861
LeNet	2.6327	1	11.0283	0.1019	0.1178
Inception	4.0283	0.0214	4.0265	0.0168	0.0067
Inception–ResNet	0.1189	0.9605	5.4974	0.2856	0.2491
InceptionV4	4.0289	0.0217	4.0262	0.0185	0.0134
Feature-Map-based CNNs	0.3882	0.882	3.9111	0.2401	0.2188
AlexNet	4.0289	0.0202	4.0254	0.0185	0.0134
CNN	2.5064	1	9.9855	0.112	0.1414

see a gap of 1.77% between its 93.65% training accuracy and 91.88% validation accuracy. Excellent performance was shown by Inception V4, which attained 96.4% validation accuracy using 87 layers.

In addition, a gap existed between training and validation performances across several models, which implies generalization issues due to the high visual complexity of Bengali. For instance, LeNet demonstrated only a 0.1% gap between its training (77.9%) and validation accuracy (78%), although its overall accuracy remained quite low. Using extra regularization through dropout and data augmentation could potentially solve standard overfitting issues across these architectures.

An unexpected anomaly was noticed during the testing phase for some specific networks. As observed in Table 15, models like CNN, ResNet, InceptionNet, and DenseNet achieved strong validation accuracies ranging from approximately 86.5% to 95.73%, but their corresponding test accuracies dropped heavily to 14.61%, 12.74%, 11.57%, and 38.91%, respectively. Other architectures evaluated on the exact same dataset split like AlexNet (97.67%) and Inception V4 (96.4%) maintained high generalization on the test set. Our working hypothesis is that this points to a structural pipeline or preprocessing discrepancy during the final evaluation code for those specific underperforming models rather than a failure of the architectures themselves. An example of such discrepancy could be a mismatch in data normalization. Because of this broad claims of 95% to 99% accuracy, all scripts must be viewed in the context of these specific testing failures.

In full word recognition, the situation is far more challenging. All models showed a significant decrease in accuracy on word images compared to isolated characters. This highlights the increased difficulties of sequence modeling, as shown in Figure 6. The accuracy of the highly performing Inception V4 decreased from 96.4% on characters to 89.12% on words. Interestingly, the simplest model LeNet showed resilience at the word level, achieving 52.49% validation accuracy, while deeper CNNs like VGG and ResNet achieved 79.66% and 42.98%, respectively. This suggests that standard CNN architectures might have representational limitations in capturing word-level visual patterns. Overall, there is a huge scope for developing Bengali specific architectures for better word recognition.

Analyzing computational efficiency also provides some useful insights. For character recognition in Table 15, models like Feature-Map-based CNN (11 layers) achieved fast inference times of 3.47 s while LeNet took 10.56 s. In contrast, highly accurate but extremely deep networks like ResNet (174 layers) took much longer time of 93.24 s for character inference. On word recognition tasks in Table 16, the inference times

Table 15
Comparative study on the handwritten character dataset

Model	Layers	Parameters	FLOPS	Time (s)	Loss	Training accuracy	Validation loss	Validation accuracy	Testing accuracy
CNN [43]	11	198202	394827	3.47	0.1502	0.9539	0.4233	0.9065	0.146135
LeNet [43]	8	53946	107205	10.56	0.8547	0.779	0.8586	0.78	0.752241
AlexNet [44, 45]	14	25192762	50366156	10.49	0.0506	0.9846	0.222	*0.9559*	0.9767
ResNet [9]	174	23828858	47498276	93.24	0.2399	0.9365	0.316	*0.9188*	0.1274
GoogLeNet/InceptionNet [43, 49, 50]	13	38706554	77411723	5.68	0.2205	0.929	0.6192	*0.865*	0.115743
Efficient Net [56]	114	2182502	4341048	9.76	0.32	0.9036	0.3234	*0.9081*	0.3891
VGG [43, 44, 46, 47]	27	39405626	78783639	11.97	0.104	0.9712	0.7993	*0.8705*	0.870438
DenseNet [48]	244	9908666	19799867	24.91	0.0616	0.9817	0.2112	*0.9573*	0.38911
ZFNet [26]	14	25196602	50373836	10.64	0.0295	0.9914	0.3418	*0.9438*	0.7596
PyramidalNet [78]	47	1188714	2372812	10.7	0.0198	0.9966	0.268	*0.9357*	0.6831
Feature-Map-based CNNs [79]	11	157050	312905	3.47	1.4735	0.5883	1.0016	*0.7294*	0.8015
Attention-based CNNs [57–59, 80]	16	120602	239823	4.58	0.1972	0.9465	0.2163	*0.9467*	0.9415
MobileNetV1 [60]	84	3353338	6662742	9.53	0.1009	0.9698	0.2232	*0.9432*	0.9385
Wide ResNet [61]	121	1722554	12207257	41.81	0.2272	0.9612	0.3112	*0.9461*	0.9512
Squeeze and Excitation Networks [62]	53	2157464	4304239	10.71	0.0193	0.9937	0.2421	*0.9536*	0.9536
Competitive Squeeze and Excitation Networks [63]	18	34054	67374	5.43	1.198	0.6724	1.2278	*0.669*	0.669
Highway Networks [64]	33	3099258	6197143	4.48	0.003	0.9992	1.0849	*0.8582*	0.8426
InceptionV4 [65]	87	6428714	12842110	21.2	0.0617	0.983	0.1655	*0.964*	0.964
PolyNet [66]	47	1188714	2372812	4.86	0.0147	0.9952	0.3346	*0.9495*	0.9495
Xception [51]	134	21110882	42112623	21.46	0.0334	0.9891	0.2816	*0.9457*	0.9395
Depth-based CNNs [67]	21	943494	1881432	5.48	0.1801	0.9479	0.1852	*0.9542*	0.9542
Residual Attention NN [68]	122	23582642	47114215	21.64	0.0184	0.9941	0.2408	*0.9527*	0.9539
Inception–ResNet [69]	74	4622236	9233656	11.1	0.1105	0.9656	0.2332	*0.9426*	0.9426

Table 16
Comparative study on the handwritten word dataset

Model	Layers	Parameters	FLOPS	Time (s)	Loss	Training accuracy	Validation loss	Validation accuracy	Testing accuracy
VGG19 [43, 44, 46, 47]	27	62642275	125256851	3.99	0.033	0.9914	1.3188	0.7966	0.8083
ZFNet [26]	14	71501923	142984392	1.61	0.012	0.9955	1.7387	0.736	0.7561
ResNet [9]	174	23912867	47666208	143.7	0.0989	0.988	2.5538	0.4298	0.3968
LeNet [43]	8	994391	1988009	0.5	0.7113	0.8082	2.1706	0.5249	0.5206
Inception [43, 49, 50]	13	531023523	1062045575	5.66	4.9202	0.0225	4.9301	0.0185	0.0219
Inception–ResNet [69]	74	5972101	11933300	2.96	0.0247	0.9927	2.3149	0.6907	0.6997
InceptionV4 [65]	87	13265491	26515578	11.02	0.0489	0.9831	0.532	*0.8912*	0.8929
Feature-Map-based CNNs [79]	11	3411619	6821957	0.97	0.3892	0.8752	0.9389	*0.7608*	0.7525
AlexNet [45, 46]	14	153287011	306554568	5.61	0.0119	0.997	0.8305	*0.8629*	0.8804
CNN [44]	11	995171	1988679	0.93	0.2344	0.9576	2.9334	*0.4815*	0.4913

Figure 5
A comparative chart from the study on the Bengali handwritten character dataset

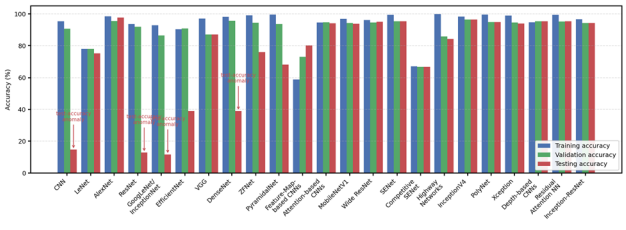
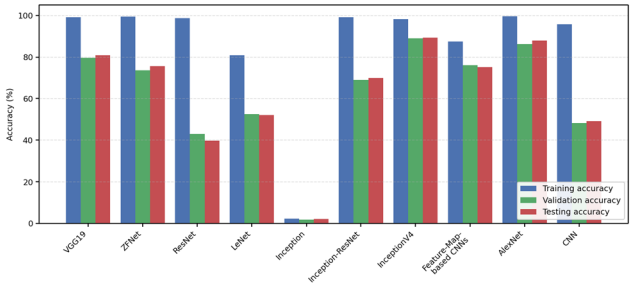


Figure 6
A comparative chart from the study on the Bengali handwritten word dataset



varied widely. LeNet processed data in just 0.5 s, while ResNet required 143.7 s. Architectures like Inception V4 demonstrated a proper balance between efficiency and accuracy for practical OCR deployment.

In summary, the comparative study highlights the capabilities and limitations of standard CNNs on the complex Bengali script. Although select deeper models match or exceed state-of-the-art accuracy for character recognition, their word level performance remains far from satisfactory. In addition, specific networks experienced severe generalization failures during testing. Major innovations in architecture design and customized training approaches for Bengali will be needed to unlock generalized handwriting recognition on this difficult script.

8. Discussion

The comprehensive experimental analysis in this study provides valuable insights into the capabilities and limitations of standard CNN architectures for recognizing handwritten characters and words across diverse scripts.

A key observation is that deeper models consistently outperform shallow networks, validating the importance of increased representational power to capture the intricate visual patterns in handwriting. Architectures such as VGG, ResNet, and InceptionNet leverage their depth to improve character recognition accuracy, although the achieved validation accuracy varies considerably by script, ranging from the low single digits for VGG and InceptionNet on Telugu, to 42.11% for InceptionNet on Arabic, and up to the mid-90s for ResNet on Devanagari and Telugu, rather than uniformly approaching 95%–99%. Most character recognition models show only modest training-validation gaps, typically under 7%, across scripts, indicating limited overfitting at the character level. More regularization through dropout and data augmentation is imperative to reduce overfitting. Ensemble approaches further help in improving robustness and stability.

An interesting finding is that Indic scripts did not uniformly underperform English in this study. Devanagari and Tamil both achieved higher best-case accuracies than English at both the character and word level, suggesting that script complexity alone does not predict recognition difficulty and that dataset size, class balance, and preprocessing likely played a substantial role.

However, word-level recognition accuracy varied widely by script rather than converging to a common plateau: the best-performing models achieved 96.33% on Tamil, 94.59% on Hindi, 89.12% on Bengali, 78.41% on English, and only 28.64% on Arabic, indicating that word-level sequence modeling difficulty is highly script-dependent rather than a uniform limitation of standard CNN architectures.

Overall, the comparative benchmarking provides data-driven guidelines for selecting CNN models tailored to target script complexity, task granularity, accuracy needs, and efficiency constraints. The findings will inform specialized deep learning research focused on advancing handwritten text recognition for diverse scripts.

9. Conclusion and Future Work

This study presented an extensive comparative evaluation of several standard CNN architectures on handwritten character and word recognition tasks spanning six major scripts—Latin, Devanagari, Bengali, Tamil, Hiragana, and Arabic. The comprehensive

benchmarking demonstrated that deeper CNN models such as VGG, ResNet, and InceptionNet generally learned more precise visual features for classifying handwritten characters, although their accuracy varied considerably by script rather than being uniformly superior, with some architectures collapsing to near-random performance on specific scripts such as Telugu and Arabic. Overfitting, where observed, was concentrated primarily at the word-recognition level rather than the character level. All models displayed considerable difficulty in generalizing to word-level recognition compared to isolated characters. The findings highlighted the need for innovations in CNN architecture design, training techniques, and attention mechanisms tailored to individual script complexity. Customization is imperative even for related scripts. Ensemble approaches help in improving model robustness. Overall, the analysis provided insightful practical guidelines and motivated script-specific research directions to advance the state-of-the-art in deep learning techniques for handwritten text recognition across languages. It affirmed the limitations of generic CNNs in capturing nuanced handwriting patterns. This study will equip researchers with knowledge to develop specialized CNN architectures and training methodologies to achieve generalized handwriting recognition capability across the diversity of global scripts. Although this research benchmarked a comprehensive set of standard CNN models, a significant scope remains for advancing handwritten text recognition. Potential future work includes developing script-specific CNN architectures tailored to individual complexity, linguistic traits, and visual styles. This can better capture intricate patterns compared to generic networks. Future work could include incorporating attention mechanisms into CNNs to explicitly model intercharacter relationships and contextual dependencies, thereby improving word-level recognition performance. Another promising direction is to leverage transfer learning by pretraining models on large annotated handwriting datasets in related scripts, which can help in compensating for the scarcity of script-specific labeled data. This can compensate for limited quantities of script-specific data. Evaluating transformer-based architectures, such as Vision Transformers, for handwritten text recognition could also be explored, given their stronger capacity to model complex global dependency compared to traditional CNNs. Future studies should also aim to confirm the reproducibility of these findings by testing on more diverse and realistic handwritten datasets that include informal scenarios and noisy conditions. It would be valuable to compare CNNs directly with other established techniques, such as hidden Markov models and recurrent neural networks, to evaluate their relative strengths and weaknesses for handwriting recognition tasks. Further research should also focus on moving beyond isolated character and word recognition toward end-to-end paragraph or page-level recognition systems suitable for real-world deployment. Implementing these models on optimized hardware platforms, including GPUs and FPGAs, could significantly enhance their efficiency and suitability for deployment in production environments. In addition, deploying the models through web or mobile interfaces could enable real-time handwriting recognition applications for broader user access. Finally, exploring unsupervised and semisupervised learning approaches may help in reducing reliance on extensively annotated data and in better utilizing abundant unlabeled handwriting samples. In summary, this research established valuable insights and guidelines from extensive comparative analysis of standard CNNs for handwritten text recognition. Future work can build on these learnings to advance deep learning techniques to achieve generalized human-level handwriting recognition capability across the world's scripts.

Ethical Statement

This study does not contain any studies with human or animal subjects performed by any of the authors.

Conflicts of Interest

Umapada Pal is an Advisory Board Member for *Artificial Intelligence and Applications*, and was not involved in the editorial review or the decision to publish this article. The authors declare that they have no conflicts of interest to this work.

Data Availability Statement

The data that support the findings of this study are openly available in Ekush (Bengali) at https://doi.org/10.1007/978-981-13-9187-3_14; in the Devanagari handwritten character Database at https://archive.ics.uci.edu/dataset/389; in Kuzushiji-49 at https://github.com/rois-codh/kmnist; in the Telugu handwritten character database at https://www.pseudoaj.com/2016/05/pseudoajdataset0-telugu-handwritten.html; in EMNIST at https://www.nist.gov/itl/products-and-services/emnist-dataset; in the Arabic handwritten character dataset (AHCD) at https://github.com/AmrHendy/Arabic-Handwritten-Images-Recognition; in the Tamil handwritten word database at https://doi.org/10.1109/ICDAR.2013.162; in the English, Hindi, and Bengali handwritten word dataset at http://dx.doi.org/10.1109/ICFHR.2012.238; and in the Arabic handwritten word dataset at http://www.ifnenit.com/.

Author Contribution Statement

Sabyasachi Mazumder: Methodology, Software, Validation, Formal analysis, Investigation, Resources, Data curation, Writing – original draft, Writing – review & editing, Visualization. **Sayan Neogy:** Methodology, Software, Validation, Formal analysis, Investigation, Resources, Data curation, Writing – original draft, Writing – review & editing, Visualization. **Sahana Das:** Conceptualization, Methodology, Formal analysis, Writing – review & editing, Supervision. **Kaushik Roy:** Conceptualization, Methodology, Formal analysis, Resources, Writing – review & editing, Visualization, Supervision, Project administration. **Umapada Pal:** Conceptualization, Writing – review & editing, Supervision, Project administration.

References

[1] Impedovo, D., & Pirlo, G. (2008). Automatic signature verification: The state of the art. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *38*(5), 609–635. https://doi.org/10.1109/TSMCC.2008.923866

[2] Plamondon, R., & Srihari, S. N. (2000). Online and off-line handwriting recognition: A comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22*(1), 63–84. https://doi.org/10.1109/34.824821

[3] Frinken, V., Fischer, A., Manmatha, R., & Bunke, H. (2012). A novel word spotting method based on recurrent neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *34*(2), 211–224. https://doi.org/10.1109/TPAMI.2011.113

[4] Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. (2017). Inception-V4, Inception–ResNet and the impact of residual connections on learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, *31*(1), 4278–4284. https://doi.org/10.1609/aaai.v31i1.11231

[5] Sharma, C., Sharma, S., Sakshi & Chen, H.-Y. (2024). Advancements in handwritten Devanagari character recognition: A study on transfer learning and VGG16 algorithm. *Discover Applied Sciences*, *6*(12), 623. https://doi.org/10.1007/s42452-024-06217-1

[6] Konidaris, T., Gatos, B., Ntzios, K., Pratikakis, I., Theodoridis, S., & Perantonis, S. J. (2007). Keyword-guided word spotting in historical printed documents using synthetic data and user feedback. *International Journal of Document Analysis and*

Recognition, *9*, 167–177. https://doi.org/10.1007/s10032-007-0042-4

[7] Pradeep, J., Srinivasan, E., & Himavathi, S. (2011). Diagonal based feature extraction for handwritten character recognition system using neural network. *International Journal of Computer Science & Information Technology*, *3*(1), 27–38. https://doi.org/10.5121/ijcsit.2011.3103

[8] Rivas, P., & Rai, M. (2023). Enhancing CNNs performance on object recognition tasks with Gabor initialization. *Electronics*, *12*(19), 4072. https://doi.org/10.3390/electronics12194072

[9] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 770–778. https://doi.org/10.1109/CVPR.2016.90

[10] Liu, Y., Xue, J., Li, D., Zhang, W., Chiew, T. K., & Xu, Z. (2024). Image recognition based on lightweight convolutional neural network: Recent advances. *Image and Vision Computing*, *146*, 105037. https://doi.org/10.1016/j.imavis.2024.105037

[11] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324. https://doi.org/10.1109/5.726791

[12] Graves, A., Liwicki, M., Fernández, S., Bertolami, R., Bunke, H., & Schmidhuber, J. (2009). A novel connectionist system for unconstrained handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *31*(5), 855–868. https://doi.org/10.1109/TPAMI.2008.137

[13] Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, *79*(8), 2554–2558. https://doi.org/10.1073/pnas.79.8.2554

[14] Alhamad, H. A., Shehab, M., Shambour, M. K. Y., Abu-Hashem, M. A., Abuthawabeh, A., Al-Aqrabi, H., …, & Shannaq, F. B. (2024). Handwritten recognition techniques: A comprehensive review. *Symmetry*, *16*(6), 681. https://doi.org/10.3390/sym16060681

[15] S, D., & G, U. D. (2021). Study on automated approach to recognize characters for handwritten and historical document. *ACM Transactions on Asian and Low-Resource Language Information Processing*, *20*(3), 37. https://doi.org/10.1145/3396167

[16] Wolf, F., & Fink, G. A. (2024). Self-training for handwritten word recognition and retrieval. *International Journal on Document Analysis and Recognition*, *27*(3), 225–244. https://doi.org/10.1007/s10032-024-00484-9

[17] Gan, J., Chen, Y., Hu, B., Leng, J., Wang, W., & Gao, X. (2023). Characters as graphs: Interpretable handwritten Chinese character recognition via Pyramid Graph Transformer. *Pattern Recognition*, *137*, 109317. https://doi.org/10.1016/j.patcog.2023.109317

[18] Chidrawar, P. S., & Dhamdhere, V. (2023). MODI script recognition using convolutional neural networks and VGG16: A deep learning approach for historical script analysis. In *2023 7th International Conference on Computing, Communication, Control And Automation*, 1–5. https://doi.org/10.1109/ICCUBEA58933.2023.10391961

[19] Ma, M., Wang, Q.-F., Huang, S., Huang, S., Goulermas, Y., & Huang, K. (2021). Residual attention-based multi-scale script identification in scene text images. *Neurocomputing*, *421*, 222–233. https://doi.org/10.1016/j.neucom.2020.09.015

[20] Aach, M., Inanc, E., Sarma, R., Riedel, M., & Lintermann, A. (2023). Large scale performance analysis of distributed deep learning frameworks for convolutional neural networks. *Journal of Big Data*, *10*(1), 96. https://doi.org/10.1186/s40537-023-00765-w

[21] Kohonen, T. (1990). The self-organizing map. *Proceedings of the IEEE*, *78*(9), 1464–1480. https://doi.org/10.1109/5.58325

[22] Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., & Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, *2011*(2), 4.

[23] Santos, C. F. G. D., Arrais, R. R., Silva, J. V. S. D., Silva, M. H. M. D., Neto, W. B. G. D. A., Lopes, L. T., …, & Tasca, A. A. (2025). ISP meets deep learning: A survey on deep learning methods for image signal processing. *ACM Computing Surveys*, *57*(5), 127. https://doi.org/10.1145/3708516

[24] Mienye, I. D., & Swart, T. G. (2024). A comprehensive review of deep learning: Architectures, recent advances, and applications. *Information*, *15*(12), 755. https://doi.org/10.3390/info15120755

[25] Cheng, J., Tian, S., Yu, L., Gao, C., Kang, X., Ma, X., …, & Lu, H. (2022). ResGANet: Residual group attention network for medical image classification and segmentation. *Medical Image Analysis*, *76*, 102313. https://doi.org/10.1016/j.media.2021.102313

[26] Jindal, A., & Ghosh, R. (2024). A semi-self-supervised learning model to recognize handwritten characters in ancient documents in Indian scripts. *Neural Computing and Applications*, *36*(20), 11791–11808. https://doi.org/10.1007/s00521-023-09372-5

[27] Coquenet, D., Chatelain, C., & Paquet, T. (2023). End-to-end handwritten paragraph text recognition using a vertical attention network. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *45*(1), 508–524. https://doi.org/10.1109/TPAMI.2022.3144899

[28] Zhong, Y., Daud, K. M., Nor, A. N. B. M., Ikuesan, R. A., & Moorthy, K. (2023). Offline handwritten Chinese character using convolutional neural network: State-of-the-art methods. *Journal of Advanced Computational Intelligence and Intelligent Informatics*, *27*(4), 567–575. https://doi.org/10.20965/jaciii.2023.p0567

[29] AlKendi, W., Gechter, F., Heyberger, L., & Guyeux, C. (2024). Advancements and challenges in handwritten text recognition: A comprehensive survey. *Journal of Imaging*, *10*(1), 18. https://doi.org/10.3390/jimaging10010018

[30] Xu, M., Yoon, S., Fuentes, A., & Park, D. S. (2023). A comprehensive survey of image augmentation techniques for deep learning. *Pattern Recognition*, *137*, 109347. https://doi.org/10.1016/j.patcog.2023.109347

[31] Hu, J., & Wang, Z. (2025). A novel CNN architecture for image restoration with implicit frequency selection. *Connection Science*, *37*(1), 2465448. https://doi.org/10.1080/09540091.2025.2465448

[32] 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*, 53. https://doi.org/10.1186/s40537-021-00444-8

[33] Sharma, A., & Jayagopi, D. B. (2021). Towards efficient unconstrained handwriting recognition using dilated temporal convolution network. *Expert Systems with Applications*, *164*, 114004. https://doi.org/10.1016/j.eswa.2020.114004

[34] Zhang, Q., Liu, F., & Song, W. (2025). IMTLM-Net: Improved multi-task transformer based on localization mechanism network for handwritten English text recognition. *Complex & Intelligent Systems*, *11*(1), 125. https://doi.org/10.1007/s40747-024-01713-8

[35] Giménez, A., Khoury, I., Andrés-Ferrer, J., & Juan, A. (2014). Handwriting word recognition using windowed Bernoulli HMMs. *Pattern Recognition Letters*, *35*, 149–156. https://doi.org/10.1016/j.patrec.2012.09.002

[36] Ashlin Deepa, R. N., & Rajeswara Rao, R. (2020). A novel nearest interest point classifier for offline Tamil handwritten character recognition. *Pattern Analysis and Applications*, *23*(1), 199–212. https://doi.org/10.1007/s10044-018-00776-x

[37] Humayun, M., Siddiqi, R., Uddin, M., Kandhro, I. A., Abdelhaq, M., & Alsaqour, R. (2024). A novel methodology for offline English handwritten character recognition using ELBP-based sequential (CNN). *Neural Computing and Applications*, *36*(30), 19139–19156. https://doi.org/10.1007/s00521-024-10206-1

[38] Zhang, S., Zhou, C., Li, Y., Zhang, X., Ye, L., & Wei, Y. (2023). Irregular scene text detection based on a graph convolutional network. *Sensors*, *23*(3), 1070. https://doi.org/10.3390/s23031070

[39] Kaur, H., & Kumar, M. (2018). A comprehensive survey on word recognition for non-Indic and Indic scripts. *Pattern Analysis and Applications*, *21*, 897–929. https://doi.org/10.1007/s10044-018-0731-2

[40] Yavartanoo, M., Hung, S.-H., Neshatavar, R., Zhang, Y., & Lee, K. M. (2021). Polynet: Polynomial neural network for 3D shape recognition with polyshape representation. In *2021 International Conference on 3D Vision*, 1014–1023. https://doi.org/10.1109/3DV53792.2021.00109

[41] Joseph Raj, A. N., Junmin, C., Nersisson, R., Mahesh, V. G., & Zhuang, Z. (2022). Bilingual text detection from natural scene images using faster R-CNN and extended histogram of oriented gradients. *Pattern Analysis and Applications*, *25*(4), 1001–1013. https://doi.org/10.1007/s10044-022-01066-3

[42] Liang, J., Nguyen, C. T., Zhu, B., & Nakagawa, M. (2019). An online overlaid handwritten Japanese text recognition system for small tablet. *Pattern Analysis and Applications*, *22*, 233–241. https://doi.org/10.1007/s10044-018-0746-8

[43] Bora, M. B., Daimary, D., Amitab, K., & Kandar, D. (2020). Handwritten character recognition from images using CNN-ECOC. *Procedia Computer Science*, *167*, 2403–2409. https://doi.org/10.1016/j.procs.2020.03.293

[44] K.O, M. A., & Poruran, S. (2020). OCR-nets: Variants of pre-trained CNN for Urdu handwritten character recognition via transfer learning. *Procedia Computer Science*, *171*, 2294–2301. https://doi.org/10.1016/j.procs.2020.04.248

[45] Tuggener, L., Schmidhuber, J., & Stadelmann, T. (2022). Is it enough to optimize CNN architectures on ImageNet? *Frontiers in Computer Science*, *4*, 1041703. https://doi.org/10.3389/fcomp.2022.1041703

[46] Chen, F., & Tsou, J. Y. (2022). Assessing the effects of convolutional neural network architectural factors on model performance for remote sensing image classification: An in-depth investigation. *International Journal of Applied Earth Observation and Geoinformation*, *112*, 102865. https://doi.org/10.1016/j.jag.2022.102865

[47] Guo, H., Wang, T., Yun, J., & Zhao, J. (2025). Multilingual natural scene text detection via global feature fusion. *Applied Intelligence*, *55*(1), 80. https://doi.org/10.1007/s10489-024-05951-8

[48] Bundea, M., & Danciu, G. M. (2024). Pneumonia image classification using densenet architecture. *Information*, *15*(10), 611. https://doi.org/10.3390/info15100611

[49] Alruwaili, M., & Mohamed, M. (2025). An integrated deep learning model with EfficientNet and ResNet for accurate multi-class skin disease classification. *Diagnostics*, *15*(5), 551. https://doi.org/10.3390/diagnostics15050551

[50] Karrach, L., & Pivarčiová, E. (2023). Using a convolutional neural network for machine written character recognition. *TEM Journal*, *12*(3), 1252–1259. http://dx.doi.org/10.18421/TEM123-03

[51] Madhu, G., Kautish, S., Gupta, Y., Nagachandrika, G., Biju, S. M., & Kumar, M. (2024). XCovNet: An optimized Xception convolutional neural network for classification of COVID-19 from point-of-care lung ultrasound images. *Multimedia Tools and Applications*, *83*(11), 33653–33674. https://doi.org/10.1007/s11042-023-16944-z

[52] El Khayati, M., Kich, I., & Taouil, Y. (2024). CNN-based methods for offline Arabic handwriting recognition: A review. *Neural Processing Letters*, *56*(2), 115. https://doi.org/10.1007/s11063-024-11544-w

[53] Lu, J., Liu, X., Ma, X., Tong, J., & Peng, J. (2023). Improved MobileNetV2 crop disease identification model for intelligent agriculture. *PeerJ Computer Science*, *9*, e1595. https://doi.org/10.7717/peerj-cs.1595

[54] Li, Z., Su, Y., Zhang, Y., Yin, H., Sun, J., & Wu, X. (2024). Remote sensing image classification method based on improved ShuffleNet convolutional neural network. *Intelligent Data Analysis*, *28*(2), 397–414. https://doi.org/10.3233/IDA-227217

[55] Tsivgoulis, M., Papastergiou, T., & Megalooikonomou, V. (2022). An improved SqueezeNet model for the diagnosis of lung cancer in CT scans. *Machine Learning with Applications*, *10*, 100399. https://doi.org/10.1016/j.mlwa.2022.100399

[56] Lincy, R. B., & Gayathri, R. (2022). Optimized convolutional neural network for Tamil handwritten character recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, *36*(11), 2253003. https://doi.org/10.1142/S0218001422530032

[57] Li, G., Fang, Q., Zha, L., Gao, X., & Zheng, N. (2022). HAM: Hybrid attention module in deep convolutional neural networks for image classification. *Pattern Recognition*, *129*, 108785. https://doi.org/10.1016/j.patcog.2022.108785

[58] Hamdi, S., Oussalah, M., Moussaoui, A., & Saidi, M. (2022). Attention-based hybrid CNN-LSTM and spectral data augmentation for COVID-19 diagnosis from cough sound. *Journal of Intelligent Information Systems*, *59*(2), 367–389. https://doi.org/10.1007/s10844-022-00707-7

[59] Kamyab, M., Liu, G., Rasool, A., & Adjeisah, M. (2022). ACR-SA: Attention-based deep model through two-channel CNN and Bi-RNN for sentiment analysis. *PeerJ Computer Science*, *8*, e877. https://doi.org/10.7717/peerj-cs.877

[60] Guo, C., Zhou, Q., Jiao, J., Li, Q., & Zhu, L. (2024). A modified MobileNetV3 model using an attention mechanism for eight-class classification of breast cancer pathological images. *Applied Sciences*, *14*(17), 7564. https://doi.org/10.3390/app14177564

[61] Wang, J., Luan, Z., Yu, Z., Ren, J., Gao, J., Yuan, K., & Xu, H. (2022). Superpixel segmentation with squeeze-and-excitation networks. *Signal, Image and Video Processing*, *16*, 1161–1168. https://doi.org/10.1007/s11760-021-02066-2

[62] Venkata Krishna Reddy, M., Raghavendar Raju, L., Sai Prasad, K., Kumari, D. D. A., Veerabhadram, V., & Yamsani, N. (2025). Enhanced effective convolutional attention network with squeeze-and-excitation inception module for multi-label clinical document classification. *Scientific Reports*, *15*(1), 16988. https://doi.org/10.1038/s41598-025-98719-0

[63] Zhang, X., Li, L., Di, D., Wang, J., Chen, G., Jing, W., & Emam, M. (2022). SERNet: Squeeze and excitation residual network for semantic segmentation of high-resolution remote sensing images. *Remote Sensing*, *14*(19), 4770. https://doi.org/10.3390/rs14194770

[64] Daraee, F., Mozaffari, S., & Razavi, S. M. (2021). Handwritten keyword spotting using deep neural networks and certainty

prediction. *Computers and Electrical Engineering*, *92*, 107111. https://doi.org/10.1016/j.compeleceng.2021.107111

[65] Shafik, W., Tufail, A., Liyanage De Silva, C., & Awg Haji Mohd Apong, R. A. (2025). A novel hybrid Inception–Xception convolutional neural network for efficient plant disease classification and detection. *Scientific Reports*, *15*(1), 3936. https://doi.org/10.1038/s41598-024-82857-y

[66] Shen, G.-T., & Huang, Y.-F. (2023). Dual-Pyramid Wide Residual Network for semantic segmentation on cross-style datasets. *Information*, *14*(12), 630. https://doi.org/10.3390/info14120630

[67] Ullah, A., Elahi, H., Sun, Z., Khatoon, A., & Ahmad, I. (2022). Comparative analysis of AlexNet, ResNet18 and SqueezeNet with diverse modification and arduous implementation. *Arabian Journal for Science and Engineering*, *47*(2), 2397–2417. https://doi.org/10.1007/s13369-021-06182-6

[68] Zhu, K., & Wu, J. (2021). Residual attention: A simple but effective method for multi-label recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 184–193. https://doi.org/10.1109/ICCV48922.2021.00025

[69] Alruwaili, M., Shehab, A., & Abd El-Ghany, S. (2021). COVID-19 diagnosis using an enhanced Inception–ResNetV2 deep learning model in CXR images. *Journal of Healthcare Engineering*, *2021*(1), 6658058. https://doi.org/10.1155/2021/6658058

[70] Abdallah, A., Hamada, M., & Nurseitov, D. (2020). Attention-based fully gated CNN-BGRU for Russian handwritten text. *Journal of Imaging*, *6*(12), 141. https://doi.org/10.3390/jimaging6120141

[71] Qiu, H., & Dong, J. (2023). A robust residual shrinkage balanced network for image recognition from Japanese historical documents. *Journal of Sensors*, *2023*(1), 8316638. https://doi.org/10.1155/2023/8316638

[72] Karapu, B. M., Anoop, G. L., Elappila, M., & Mithun, B. N. (2024). Handwritten Telugu character recognition using machine learning. In *2024 International Conference on Distributed Computing and Optimization Techniques*, 1–6. https://doi.org/10.1109/ICDCOT61034.2024.10515646

[73] Bhardwaj, S., Wang, Y., Yu, G., & Wang, Y. (2023). Information set supported deep learning architectures for improving noisy image classification. *Scientific Reports*, *13*(1), 4417. https://doi.org/10.1038/s41598-023-31462-6

[74] Shibly, M. M. A., Tisha, T. A., Tani, T. A., & Ripon, S. (2021). Convolutional neural network-based ensemble methods to recognize Bangla handwritten character. *PeerJ Computer Science*, *7*, e565. https://doi.org/10.7717/peerj-cs.565

[75] Qi, H., Yang, H., Wang, Z., Ye, J., Xin, Q., Zhang, C., & Lang, Q. (2025). AncientGlyphNet: An advanced deep learning framework for detecting ancient Chinese characters in complex scene. *Artificial Intelligence Review*, *58*(3), 88. https://doi.org/10.1007/s10462-024-11095-5

[76] Pal, U., Roy, R. K., & Kimura, F. (2012). Multi-lingual city name recognition for Indian postal automation. In *2012 International Conference on Frontiers in Handwriting Recognition*, 169–173. https://doi.org/10.1109/ICFHR.2012.238

[77] Pechwitz, M., El Abed, H., & Märgner, V. (2012). Handwritten Arabic word recognition using the IFN/ENIT-database. In V. Märgner & H. El Abed (Eds.), *Guide to OCR for Arabic scripts* (pp. 169–213). Springer. https://doi.org/10.1007/978-1-4471-4072-6_8

[78] Babaoğlu, İ., Kahveci, S., & Kılıç, A. (2024). Enhanced pyramidal residual networks for single image super-resolution. *Neural Computing and Applications*, *36*(19), 11563–11577. https://doi.org/10.1007/s00521-024-09702-1

[79] Yu, H., Yuan, X., Jiang, R., Feng, H., Liu, J., & Li, Z. (2023). Feature reduction networks: A convolution neural network-based approach to enhance image dehazing. *Electronics*, *12*(24), 4984. https://doi.org/10.3390/electronics12244984

[80] Liu, Z., & Xu, F. (2023). Interpretable neural networks: Principles and applications. *Frontiers in Artificial Intelligence*, *6*, 974295. https://doi.org/10.3389/frai.2023.974295

[81] Rabby, A. S. A., Haque, S., Abujar, S., & Hossain, S. A. (2018). Ekushnet: Using convolutional neural network for Bangla handwritten recognition. *Procedia Computer Science*, *143*, 603–610. https://doi.org/10.1016/j.procs.2018.10.437

[82] Deore, S. P., & Pravin, A. (2020). Devanagari handwritten character recognition using fine-tuned deep convolutional neural network on trivial dataset. *Sādhanā*, *45*(1), 243. https://doi.org/10.1007/s12046-020-01484-1

How to Cite: Mazumder, S., Neogy, S., Das S., Roy, K., & Pal, U. (2026). Review of the Performance of CNN Models on Handwriting Recognition. *Artificial Intelligence and Applications*, *4*(4), 469–488. https://doi.org/10.47852/bonviewAIA52023297