IAES International Journal of Artificial Intelligence (IJ-AI)
Vol. 14, No. 6, December 2025, pp. 5038~5048
ISSN: 2252-8938, DOI: 10.11591/ijai.v14.i6.pp5038-5048

Kannada handwritten numeral recognition through deep learning and optimized hyperparameter tuning

Ujwala B. S.[1], Pramod Kumar S.[1], H. R. Mahadevaswamy[2], Sumathi K.[1]
[1]Department of Electronics and Communication Engineering, Faculty of Electronics and Communication, Jawaharlal Nehru New College of Engineering, Visvesvaraya Technological University, Shivamogga, India
[2]Technical Education Division, Jagadguru Sri Shivarathreeshwara University, Noida, India

Article Info

Article history:

Received Mar 10, 2025
Revised Sep 12, 2025
Accepted Oct 18, 2025

Keywords:

Deep learning
Handwritten recognition
Kannada numerals
Optimized hyperparameters
Transfer learning

ABSTRACT

The classification of handwritten numerals is a vital and challenging task in developing automated systems, including postal address sorting and license plate recognition. The present study elucidates a new methodology for recognizing Kannada handwritten numerals using deep learning ResNet and VGG architecture with transfer learning. The challenge in Kannada handwritten recognition is complicated structural hierarchy and large vocabulary. The major problem in deep neural networks is vanishing gradient, which can lead to degradation in character recognition, and was addressed using our new methodology using ResNet architecture. We apply the proposed ResNet method in various real-world applications and compare it with convolutional neural networks (CNN) architecture, VGG. The experiment was implemented with the Google Colab software version on a self-created dataset, with handwritten Kannada numerals fed as the input to the recognition process. Our proposed method achieved a high accuracy of 99.20% on training samples and a generalization accuracy of 97.5% on test samples, indicating our method's effectiveness in recognizing handwritten Kannada numerals.

Corresponding Author:

Ujwala B. S.
Department of Electronics and Communication Engineering, Faculty of Electronics and Communication
Jawaharlal Nehru New College of Engineering
Shivamogga, Karnataka State, India
Email: ujwalaravi2004@jnnce.ac.in

1. INTRODUCTION

In many business and workflow platforms paper forms from document digitization and management to law enforcement and security systems. Maintaining large amounts of physical documents is tedious to manage and store. Paperless management of data is an ideal solution that involves scanning the document into an image, but it calls for a lot of manual intervention, and the process takes a lot of time. The best solution is to convert text images into text data using optical character recognition (OCR) technology. OCR transforms scanned documents into machine-encoded text. It is required to reduce the labor cost, to enhance the efficiency, and to save the time. This process is a need of the day for all languages, especially for Indic scripts, as there are a lot of compound characters that need to be considered along with basic characters. In spite of tremendous advances that have been incorporated in recent years in recognizing numerals written by hand, they still need to be solved in accurate recognition. Many traditional techniques are used for OCR; feature detection techniques like gradient features (Sobel operator) and density features were used [1]. Machine learning techniques like histogram of oriented gradients (HoG) for extracting features, K-nearest neighbors (KNN), and support vector machine (SVM) algorithms are used to classify Devanagari and

Bengali characters [2]. A work on Kannada numeral identification is implemented using HOG descriptors for extracting features and SVM for categorization [3]. A novel approach for recognizing Sanskrit characters using the convents' deep learning approach as classifiers for Indic OCRs is implemented [4]. Different machine learning approaches have been incorporated for handwritten digit recognition, such as SVM and KNN. In the last few years, researchers have witnessed a deepening of deep learning research, the training time of deep learning networks has greatly reduced with graphics processing units (GPUs), and the efficiency [5] and accuracy have been greatly improved [6].

The official language used for administrative purposes in Karnataka state is Kannada, with nearly 56.9 million speakers worldwide. The Kannada language is derived from the official Kannada script. The Kannada script consists of a total of forty-nine letters. Kannada script consists of vowels (swaras) and consonants. Letters representing vyanjanas and yoga vahaks are combined with non-intervening vowels to form digraphs (kagunitas). Several researchers have made significant contributions to automating OCR. However, handwritten character recognition still needs to be completed. It continues to be a pivotal challenge in the field of pattern recognition, attributed to the extensive vocabulary, complex structural hierarchy, and the diverse array of handwriting styles exhibited by individuals. Handwriting recognition, also known as handwriting OCR or cursive OCR, constitutes a subset of OCR technology focused on deciphering handwritten text, encompassing both manuscript and cursive forms. Manuscript-style text, characterized by separate block letters, is typically easier to recognize compared to cursive writing. However, cursive handwriting is difficult as it consists of joined characters. Since handwritten character recognition is very essential for multiple applications, our work will help to identify the handwritten character. Languages other than widely used languages, like area specific and vernacular languages, preserve our culture and heritage of various communities and maintain the legacy of history, and create a positive impact that strengthens global interconnectedness.

Kannada language is most commonly used. Dravidian languages propagated mainly in Karnataka, along with Tamil Nadu, Andhra Pradesh, and Maharashtra States [7]. More than 50 million individuals contribute to the language vitality, and it is written using the Kannada script. The Kannada script consists of a total of forty-nine letters. Kannada script consists of vowels and consonants. Kannada character recognition is challenging because Kannada characters have a larger character set and similarities of characters. The width of characters varies—the presence of kagunitas and ottaksharas leads to the uneven spacing between characters and sentences. The conversion of handwritten Kannada numerals on documents into machine-readable form is very significant in the electronic field. Many algorithms and techniques have already been developed for individual digit identification. Figure 1 shows sample images of handwritten ones.

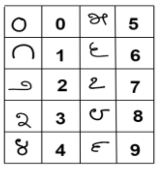

Figure 1. Sample images of Kannada handwritten digits

OCR is an essential technology for recognizing and extracting text from image datasets, which has been used in various domains such as document digitization, identification, and data entry. Handwritten OCR is more challenging than machine-printed OCR due to the high degree of difference and variability in writing styles and individual variations. In this literature survey, we reviewed classical, deep learning, transfer learning, sparse learning and combined approaches for recognizing handwritten Kannada numerals using OCR. The earliest approaches to handwriting recognition were based on rule-based methods and pattern recognition techniques. Very commonly used pattern recognition techniques for handwriting recognition include nearest gradient features, density features, neighbor classification, decision trees, and SVMs [8]. These techniques are based on manually crafted features and are constrained in their capability to capture complex features in the data. HoG features can recognize handwritten vowels and consonants.

Deep learning has transformed the field of handwriting recognition in recent years. Deep learning-based approaches have automatic feature extraction, which is one of the salient features, and extract data from it and have shown formidable performance when compared to traditional pattern recognition techniques. Convolutional neural networks (CNNs) [9] are most extensively used in handwriting recognition tasks because they capture spatial patterns or features in the images. Transfer learning facilitates the learning from one environment and generalizing to other but related problem, serving as a valuable technique in machine learning. Transfer learning has shown prominent results in handwriting recognition tasks, as it allows using pre-initialized models with large corpora to optimize outcomes on smaller datasets. In transfer learning, the pre-developed model is calibrated on the target dataset to adapt to the specific task. Some popular pre-learned models used for transfer learning in handwriting recognition tasks include ResNet [10] and VGG [11]. Sparse learning approaches have been majorly used in image recognition tasks, including handwritten digit recognition. These approaches aim to acquire sparse feature representation of the input data. One of the popular sparse learning algorithms is the sparse autoencoder (SAE) [12]. An SAE is an unsupervised learning algorithm that minimizes the difference between input data and reconstructed output. By reconstructing the input, the model learns a compact and sparse feature representation. Another popular approach for handwritten recognition of the character is the combined approach, which combines multiple feature extraction methods or models to boost the recognition performance. For example, Montazeri *et al.* [13] outlined a technique merging deep learning and dictionary learning techniques to analyze English handwritten digits. Through a multi-layered framework, they attained an accuracy rate exceeding 99%. Their innovative approach integrates singular value decomposition within a multi-layer CNN model. In this paper, one of the most popular and widely used models, ResNet, is playing a key factor in solving the problem of vanishing gradient/exploding gradient, which is most common in CNN-based architectures. The most challenging part is to handle Kannada numerals due to slants, strokes, curves, and loops.

2. METHOD

As shown in Figure 2, our novel architecture contains two phases: training and testing. The training part has its first step as pre-processing the data, which later builds the network architecture. The distinct steps in erecting the network architecture are:

– Split data set (Kannada handwritten images) into training, validation, and testing datasets.
– Select loss function and optimization algorithm (e.g., cross-entropy loss, adaptive moment estimation (Adam), and optimizer).
– Train the model by adjusting hyperparameters as needed.

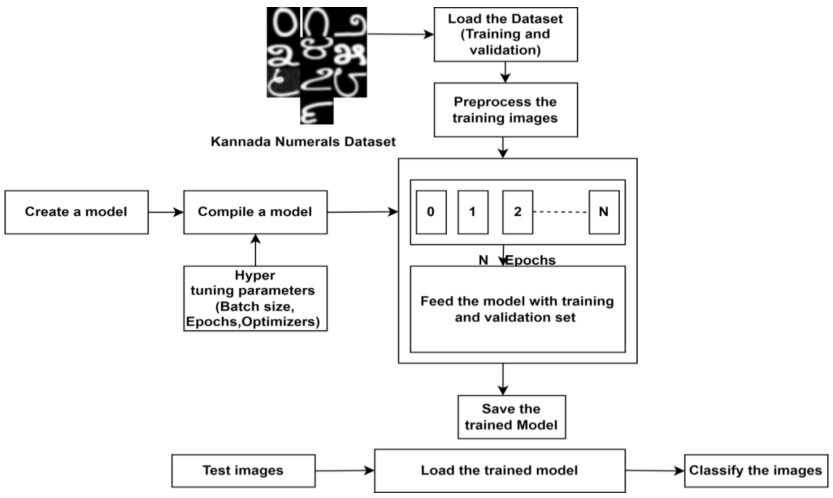
Kannada Numerals Dataset
Load the Dataset (Training and validation)
Preprocess the training images
Create a model
Compile a model
Hyper tuning parameters (Batch size, Epochs,Optimizers)
0
1
2
N
N Epochs
Feed the model with training and validation set
Save the trained Model
Test images
Load the trained model
Classify the images

Figure 2. The proposed architecture

2.1. Pre-processing

Here, the dataset was cleaned and normalized (e.g., resizing images, converting to grayscale), and augmentation techniques (e.g., rotation, flipping) were applied to enhance model robustness. Rescaling is used to reduce the target values. The original images are represented using RGB coefficients ranging from 0 to 255; such values would be too high for processing, so we use scaling to convert pixel intensities to (0, 1) by multiplying with 1/255 [14]. This technique is called normalization. Some of the pre-processing techniques used here are shear range, zoom range, and horizontal flip.

2.2. Residual network

Deep neural networks (DNN) are predominantly used to extract features and classify images based on the learning of multiple levels of representations of the data or features. However, DNN has some issues that make it difficult to work satisfactorily in all scenarios. One of the drawbacks in training DNN is vanishing gradients. ResNet is one of the methods used to solve the problem. It uses the skip connection concept to overcome the problem. The vanishing gradient problem arises when gradients are backpropagated to earlier layers and become extremely small, impeding effective learning. When complexity increases in DNN, training accuracy increases, but validation accuracy decreases because there is no learning over the system due to the vanishing gradient problem. To address the challenge of vanishing or exploding gradients, this architecture introduced residual blocks, which employ skip connections. These connections link the layer's activations propagated to subsequent layers by bypassing certain intermediate layers, thereby forming a residual block. ResNets are constructed by stacking these residual blocks. The concept of skip connection is to add the input of block to the output of the convolution block. The ResNet is comprised of residual blocks. Figure 3 illustrates the structure of these residual blocks, showcasing certain layers connected via skip connections. Consider a neural network that has x as the input and Y(x) as the output of the block. The output Y(x) is defined in (1).

$$Y(x) = F(x) + x \tag{1}$$

F(x) is the residual function that learns the residual mapping between the input x and the output Y(x). F(x) is represented by convolutional layers coupled with normalization and activation functions, such as batch normalization and rectified linear unit (ReLU). The main logic behind residual networks is to make F(x)=0, because the target is to make Y(x)=x. The idea here is to add the actual input x to your loss function F(x) and try to make the loss function zero. In many CNN architectures, x is the input, Y(x) is the output, and Y=F(x). Here, the algorithm learns or is trained from the value of Y. But in ResNet, the algorithm learns or is trained from the value of F(x). This observation leads us to recognize that the layers within the residual block aim to learn this residual function, which in turn yields an (2).

$$F(x) = Y(x) - \mathrm{x} \tag{2}$$

With the incorporation of skip connections, the original function transforms into F(x)+x as illustrated in Figure 3. These skip connections facilitate the transmission of larger gradients to earlier layers, enabling them to learn at a comparable pace to the final layers. Consequently, this empowers the training of DNN, effectively addressing the issue of vanishing gradients [15]. ResNet comprises two block types: the "identity block" (standard ResNet) and the "convolution block" (modified ResNet). An "identity block" features a skipped connection running parallel to the main sequential connection path, maintaining identical dimensions at the splitting and merging nodes. Conversely, a convolution block within the skip paths closely resembles the identity block but integrates a convolution layer.

ResNet50 constitutes a modified variant of the ResNet architecture that has 50 layers in total. It consists of 5 stages, each containing several residual blocks. The overall structure of ResNet50, shown in Figure 4, can be summarized in Table 1.

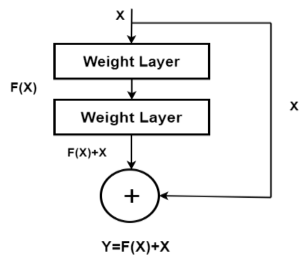

Figure 3. Skip connection of ResNet

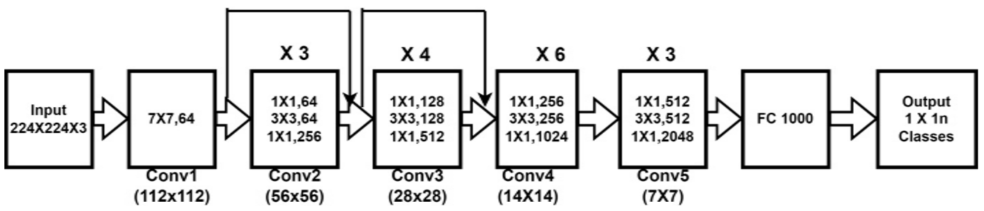

Figure 4. ResNet50 architecture

Table 1. Architecture details of ResNet50

Layer type	Details
Input layer	Input image resized and normalized to a fixed size.
Convolution layers	7×7 convolution, stride =2 → reduces spatial resolution.
Max pooling layers	Pool size 2×2, stride =2→ reduces spatial resolution
Residual blocks	Four stages of residual blocks, each with different numbers of blocks. Each block =
Fully connected layers	multiple convolutional layers + batch normalization + ReLU activation + skip connection
	Reduces each feature map to a single value → outputs a compact vector.
Global average pooling	Dense layer for classification into categories
Fully connected layer	SoftMax activation, assigns probabilities across categories.
Output layer	

2.3. VGG16 architecture

VGG represents a deep convolutional feature learning network architecture used for image classification, proposed by the visual geometry group at the University of Oxford in 2014. The VGG network is notable for its simplicity and the use of small convolutional filters (3×3) throughout the architecture, as shown in Figure 5. The original VGG architecture includes 16 or 19 layers and is named VGG16 or VGG19, respectively. VGG16 comprises 16 layers, including 13 convolutional layers and three fully connected layers [16]. The general architecture of the VGG16 can be summarized in Table 2.

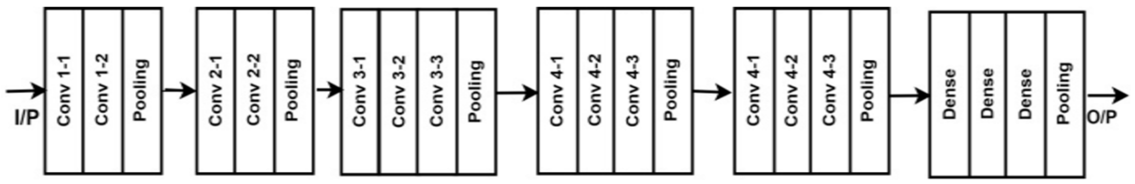

Figure 5. VGG16 model

Table 2. Architecture details of VGG16

Layer type	Details
Input layer	Input image resized to fixed size, normalized
Convolution layers	13 conv layers, filter size 3×3, stride = 1, each followed by ReLU activation and Batch normalization. Depth of feature maps increases with layers
Max pooling layers	Pool size 2×2, stride = 2, reduces spatial resolution of feature maps.
Fully connected layers	Flattened feature maps → 3 dense layers, each with 4096 neurons, ReLU activation, and Batch Normalization
Output layer	SoftMax activation, outputs class probabilities

2.4. Optimizers

With the objective of obtaining optimal solution and to reduce the losses in front and back passes, optimizers [17] guide the model to update model-parameters of neural networks by moving model weights using the gradients that minimize the loss, modulate bias-variance tradeoff, and control step size in each iteration to obtain optimal learning rates. Optimizers are the backbone for any type of recognition. Without proper optimization, the model may take too long to learn patterns or fail to converge.

2.4.1. Gradient descent

To minimize prediction error or cost functions in the model, a fundamental optimization technique called gradient descent [18] is used. First, it starts with an initial guess of the parameters and stage by stage it optimizes the parameters in towards the steepest decrease of the cost function. This happens with the help of learning rate until the minimum is reached. It is described in (3).

$$W_{New} = W_{Old} - K\frac{\partial L}{\partial_{Old}} \quad (3)$$

2.4.2. Adaptive gradient descent

This optimizer dynamically changes the learning rate ‘η’ at every time step ‘t’ based on gradient information. This helps in fast convergence and improves stability. The principle of adaptive gradient descent

(Adagrad) is that it works on cumulative sum of squared gradients. This optimization is very much suitable for sparse data [19]. and this is shown from (4) to (6).

$$W_t = W_{t-1} - \eta_t^{|} \frac{\partial L}{\partial W_{t-1}} \tag{4}$$

$$\eta_t^{|} = \frac{\eta}{\sqrt{\alpha_t + \varepsilon}} \tag{5}$$

$$\alpha_t = \sum_{i=1}^{t} \left(\frac{\partial L}{\partial W_i}\right)^2 \tag{6}$$

2.4.3. Root mean square propagation

Root mean square propagation (RMSprop) [20] is a practical and robust optimization algorithm that tries to improve Adagrad. It takes the 'exponential decaying moving average' for each parameter. Moving average concept prevents erratic movements, uses simple hyper parameters and promotes stability. The functionality is described from (7) to (10).

$$S_{dw} = \beta S_{dw} + (1-\beta) dw^2 \tag{7}$$

$$S_{db} = \beta S_{db} + (1-\beta) db^2 \tag{8}$$

$$W_N = W_0 - K\left[\frac{dw}{\sqrt{S_{dw}+\xi}}\right] \tag{9}$$

$$W_N = W_0 - K\left[\frac{db}{\sqrt{S_{db}+\xi}}\right] \tag{10}$$

2.4.4. Adaptive moment estimation

It works with exponential decaying averages and past squared gradients. The adaptive learning rate, integration of the momentum, and reliability across architectures has made Adam a default optimizer [21]. Adam is the combination of RMSProp and momentum shown from (11) to (18). The different hyperparameters are K, ξ, β1, and β2.

$$S_{db} = 0, S_{dw} = 0 \; V_{db} = 0 \quad V_{dw} = 0 \tag{11}$$

$$S_{dw} = \beta_1 S_{dw} + (1-\beta_1) dw^2 \tag{12}$$

$$S_{db} = \beta_1 S_{db} + (1-\beta_1) db^2 \tag{13}$$

$$V_{dw} = \beta_2 V_{dw} + (1-\beta_2) dw \tag{14}$$

$$V_{db} = \beta_2 V_{db} + (1-\beta_2) dw \tag{15}$$

$$S_{dw}^{|} = \frac{S_{dw}}{1-\beta_1^t} \; S_{db}^{|} = \frac{S_{db}}{1-\beta_1^t} \tag{16}$$

$$V_{db}^{|} = \frac{V_{db}}{1-\beta_2^t} \qquad V_{dw}^{|} = \frac{V_{dw}}{1-\beta_2^t} \tag{17}$$

$$W_N = W_0 - K\left[\frac{V_{dw}^{|}}{\sqrt{S_{dw}^{|}+\xi}}\right] \qquad W_N = W_0 - K\left[\frac{V_{db}^{|}}{\sqrt{S_{db}^{|}+\xi}}\right] \tag{18}$$

3. RESULTS AND DISCUSSION

3.1. Accuracy and loss comparison

The experiment is implemented using Google Colab software version; the proposed ResNet method receives input as the Kannada handwritten numerals (0 to 9) document image by taking images of different users in various styles. In the self-dataset, a total of 2500 images are taken: 200 images of each numeral for training, 25 for validation, and 25 for testing in the ratio 80:10:10. Handwritten character images from the dataset were converted to 28×28 images. The experimental result obtained after all the epochs for the given input is pictured below. At first, the pre-processing procedures are completed; after that, the feature extraction technique is applied to the pre-processed image. After observing the trend of training accuracy, testing, and loss values, the number of epochs is configured to values for both validation and training samples to optimize the tradeoff between accuracy and epoch. In this experiment, the ResNet50 model is successful in obtaining an accuracy of 98.20% during the training phase, an accuracy on the validation set is 97.5%, and unseen test data is of 97.2%, as shown in Figures 6 and 7, and the VGG16 model is successful in obtaining evaluation accuracy over the training samples is of 98.10%, a recognition rate on validation samples is of 97.33%, and a generalization accuracy on the test dataset is of 97%, as shown in Figures 8 and 9. By employing this methodology, we successfully achieved the best accuracy and a low loss rate. But due to the similarity between some characters, accuracy is a little less. Table 3 shows the architecture and performance comparison between ResNet50 and VGG16.

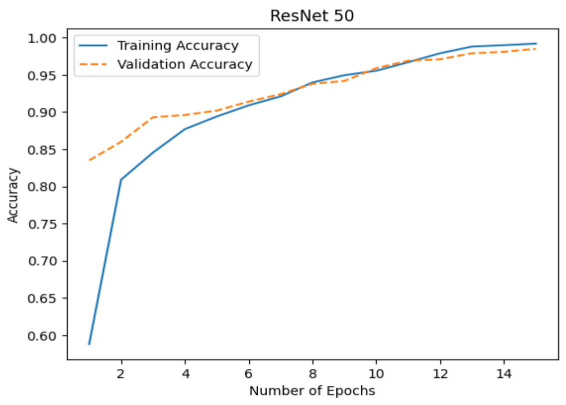

Figure 6. Training and validation epoch wise accuracy graph for ResNet50 on the Kannada numbers dataset

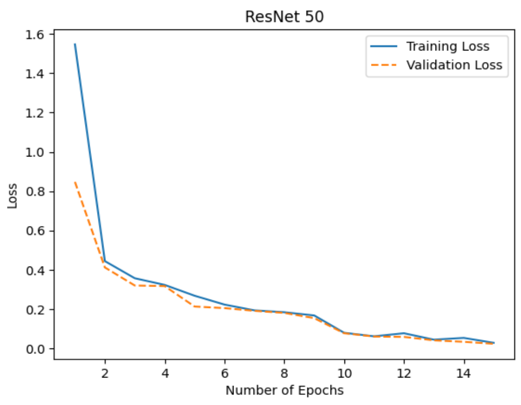

Figure 7. Training and validation loss visualization graph for ResNet50 on the Kannada numbers dataset

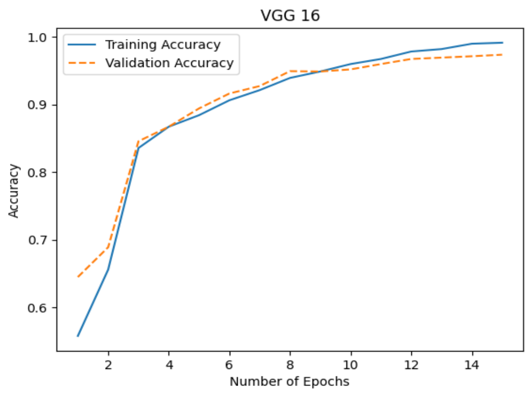

Figure 8. Training and validation epoch wise accuracy graph for VGG16 on the Kannada numbers dataset

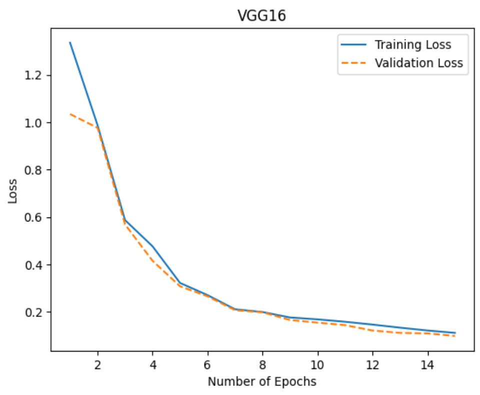

Figure 9. Training and validation loss visualization graph for VGG16 on the Kannada numbers dataset

Table 3. Architecture and performance comparison between ResNet50 and VGG16

Optimizers	ResNet50	VGG16
Total parameters	2,56,96,138	1,52,50,250
Trainable parameters	21,08,426	5,35,562
Non-trainable parameter	2,35,87,712	1,47,14,688
Epochs	15	15
Training samples	2000	2000
Training accuracy	98.20%	98.10%
Validation accuracy	97.5%	97.33%
Testing accuracy	97.2%	97%
Optimizer	Adam	Adam

3.2. Optimizer analysis

Out of different deep learning parameters, optimizers play a crucial role when configuring a neural network [22]. The table provides results of accuracy comparison for 15 epochs. The best optimizer here is the Adam optimizer [23]. Table 2 shows an accuracy comparison of ResNet50 and VGG16. Figure 10 shows the comparison of accuracy with different optimizers. Table 4 shows batch size analysis for different optimizers.

Table 4. Accuracy comparison of ResNet50 and VGG16

Optimizers	ResNet50	VGG16
SGD	94.52	82.70
Adagrad	90.75	66.45
RMS prop	94.80	82.75
Adam	98.20	98.10

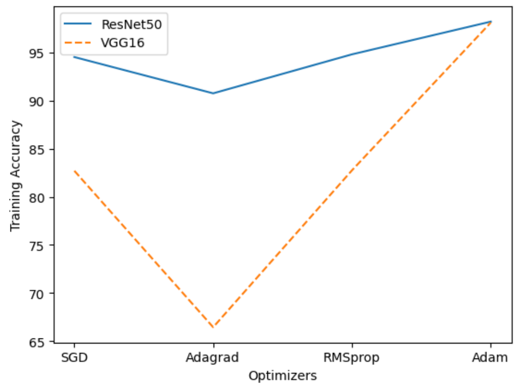

Figure 10. Comparison of optimizers for ResNet50 and VGG16 for Kannada numerals

3.3. Batch size analysis

Batch size is an important hyperparameter that affects the performance of the network. It is defined as the total number of samples passed through the network in both forward and backward passes [24]. Table 5 shows the comparison of accuracy and batch size for both ResNet50 and VGG16 transfer learning methods. Figure 11 shows the comparison of batch size analysis for ResNet50 and VGG16 for Kannada numerals.

Table 5. Batch size comparison of ResNet50 and VGG16

Optimizer	Accuracy (%) ResNet50 Batch size=32	Accuracy (%) ResNet50 Batch size=64	Accuracy (%) VGG16 Batch size =32	Accuracy (%) VGG16 Batch size =64
SGD	94.52	90.58	82.70	77.14
Adagrad	90.75	85.41	66.45	59.04
RMS prop	94.80	89.69	82.75	79.36
Adam	98.20	97.67	98.10	89.78

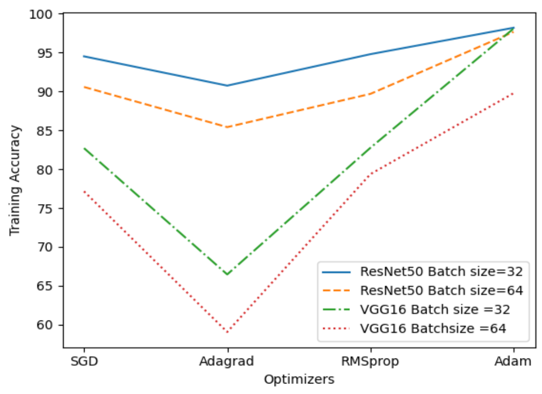

Figure 11. Comparison of batch size analysis for ResNet50 and VGG16 for Kannada numerals

3.4. Benchmarking the obtained results with earlier research

The proposed methodology has shown the best results compared with earlier research works. The best training accuracy is 98.20%, as shown in Table 6. The use of the Google Colab laboratory and the optimization techniques of the hyperparameters provide deeper insights in learning a reliable character recognition framework with robust, superior accuracy and efficient processing speed.

Table 6. Performance comparison from literature review

Authors	Feature extraction architecture	Classifier	Accuracy (%)
Hu [25]	ResNet50	ResNet50	97.38
Chandrakala and Thippeswamy [26]	ResNet50	ResNet50	15
Dutta *et al.* [27]	ResNet50	SVM	90
Proposed	ResNet50	ResNet50	98.20

4. CONCLUSION

This manuscript proposes a new methodology for recognizing Kannada handwritten numerals using deep learning ResNet architecture with transfer learning. The proposed ResNet and VGG methodologies were evaluated using performance parameters on a self-created dataset, and the obtained results underwent comparison with state-of-the-art literature. The proposed ResNet method addressed the issue of the vanishing gradient problem, which can lead to degradation in character recognition. The experiment was implemented using the Google Colab software version, with handwritten Kannada numeral images fed as the input to the recognition process. The proposed ResNet method shows the results of high evaluation accuracy over the training samples of 98.2% and generalization accuracy on the test dataset of 97.2% which indicates the accomplishment and efficiency of recognizing handwritten Kannada numerals, which can be potentially applied in various real-life applications, such as digit recognition, OCR, and document digitization. Overall, the proposed ResNet method provides a promising solution for automated systems requiring accurate handwritten Kannada numeral recognition.

FUNDING INFORMATION

Authors state no funding involved.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Ujwala B. S.	✓	✓	✓	✓	✓	✓		✓	✓	✓			✓	
Pramod Kumar S.		✓				✓		✓		✓	✓	✓		
H. R. Mahadevaswamy	✓		✓	✓		✓	✓		✓		✓	✓	✓	
Sumathi K.			✓		✓		✓			✓		✓		✓

C	: **C**onceptualization	I	: **I**nvestigation	Vi	: **Vi**sualization
M	: **M**ethodology	R	: **R**esources	Su	: **Su**pervision
So	: **So**ftware	D	: **D**ata Curation	P	: **P**roject administration
Va	: **Va**lidation	O	: Writing - **O**riginal Draft	Fu	: **Fu**nding acquisition
Fo	: **Fo**rmal analysis	E	: Writing - Review & **E**diting		

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding author, [UBS], upon reasonable request.

REFERENCES

[1] A. M. M. O. Chacko and P. M. Dhanya, "Multiple classifier system for offline Malayalam character recognition," *Procedia Computer Science*, vol. 46, pp. 86–92, 2015, doi: 10.1016/j.procs.2015.01.061.

[2] D. P. Yadav and M. Kumar, "Kannada character recognition in images using histogram of oriented gradients and machine learning," in *Proceedings of the 2nd International Conference on Computer Vision & Image Processing*, Singapore: Springer Singapore, 2018, vol. 704, pp. 265–277, doi: 10.1007/978-981-10-7898-9_22.

[3] S. Karthik and K. S. Murthy, "Handwritten Kannada numerals recognition using histogram of oriented gradient descriptors and support vector machines," in *Emerging ICT for Bridging the Future – Proceedings of the 49th Annual Convention of the Computer Society of India CSI*, 2015, pp. 51–57, doi: 10.1007/978-3-319-13731-5_7.

[4] M. Avadesh and N. Goyal, "Optical character recognition for Sanskrit using convolution neural networks," in *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*, Apr. 2018, pp. 447–452, doi: 10.1109/DAS.2018.50.

[5] C. Boufenar, A. Kerboua, and M. Batouche, "Investigation on deep learning for off-line handwritten Arabic character recognition," *Cognitive Systems Research*, vol. 50, pp. 180–195, Aug. 2018, doi: 10.1016/j.cogsys.2017.11.002.

[6] B. R. Kavitha and C. Srimathi, "Benchmarking on offline handwritten Tamil character recognition using convolutional neural networks," *Journal of King Saud University*, vol. 34, no. 4, pp. 1183–1190, Apr. 2022, doi: 10.1016/j.jksuci.2019.06.004.

[7] V. C. Hallur and R. S. Hegadi, "Handwritten Kannada numerals recognition using deep learning convolution neural network (DCNN) classifier," *CSI Transactions on ICT*, vol. 8, no. 3, pp. 295–309, 2020, doi: 10.1007/s40012-020-00273-9.

[8] T. K. Bhowmik, P. Ghanty, A. Roy, and S. K. Parui, "SVM-based hierarchical architectures for handwritten Bangla character recognition," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 12, no. 2, pp. 97–108, Jul. 2009, doi: 10.1007/s10032-009-0084-x.

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

[10] M. Mhapsekar, P. Mhapsekar, A. Mhatre, and V. Sawant, "Implementation of residual network (ResNet) for Devanagari handwritten character recognition," in *Advanced Computing Technologies and Applications*, 2020, pp. 137–148, doi: 10.1007/978-981-15-3242-9_14.

[11] C. Sharma, S. Sharma, Sakshi, and H.-Y. Chen, "Advancements in handwritten Devanagari character recognition: a study on transfer learning and VGG16 algorithm," *Discover Applied Sciences*, vol. 6, no. 12, Nov. 2024, doi: 10.1007/s42452-024-06217-1.

[12] M. Aamir, N. M. Nawi, H. B. Mahdin, R. Naseem, and M. Zulqarnain, "Auto-encoder variants for solving handwritten digits classification problem," *International Journal of Fuzzy Logic and Intelligent Systems*, vol. 20, no. 1, pp. 8–16, Mar. 2020, doi: 10.5391/IJFIS.2020.20.1.8.

[13] A. Montazeri, M. Shamsi, and R. Dianat, "Using a new approach in deep dictionary learning to handwriting number classification," in *2020 25th International Computer Conference, Computer Society of Iran (CSICC)*, Tehran, Jan. 2020, pp. 1–8, doi: 10.1109/CSICC49403.2020.9050068.

[14] P. Sharma, P. Hans, and S. C. Gupta, "Classification of plant leaf diseases using machine learning and image preprocessing techniques," in *2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, Noida, India, Jan. 2020, pp. 480–484, doi: 10.1109/Confluence47617.2020.9057889.

[15] L.-K. Huang, H.-T. Tseng, C.-C. Hsieh, and C.-S. Yang, "Deep learning based text detection using ResNet for feature extraction," *Multimedia Tools and Applications*, vol. 82, no. 30, pp. 46871–46903, Dec. 2023, doi: 10.1007/s11042-023-15449-z.

[16] S. Singh, N. K. Garg, and M. Kumar, "VGG16: Offline handwritten Devanagari word recognition using transfer learning," *Multimedia Tools and Applications*, vol. 83, no. 29, pp. 72561–72594, Feb. 2024, doi: 10.1007/s11042-024-18394-7.

[17] D. Wu, B. Sun, and M. Shang, "Hyperparameter learning for deep learning-based recommender systems," *IEEE Transactions on Services Computing*, vol. 16, no. 4, pp. 2699–2712, Jul. 2023, doi: 10.1109/TSC.2023.3234623.

[18] X. Wang, L. Yan, and Q. Zhang, "Research on the application of gradient descent algorithm in machine learning," in *2021 International Conference on Computer Network, Electronic and Automation (ICCNEA)*, Xi'an, China, 2021, pp. 11–15. doi: 10.1109/ICCNEA53019.2021.00014.

[19] L. Shen, C. Chen, F. Zou, Z. Jie, J. Sun, and W. Liu, "A unified analysis of adagrad with weighted aggregation and momentum acceleration," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 10, pp. 14482–14490, Oct. 2024, doi: 10.1109/TNNLS.2023.3279381.

[20] R. Elshamy, O. A.-Elnasr, M. Elhoseny, and S. Elmougy, "Improving the efficiency of RMSProp optimizer by utilizing Nestrove in deep learning," *Scientific Reports*, vol. 13, no. 1, May 2023, doi: 10.1038/s41598-023-35663-x.

[21] S. Bock and M. Weis, "A proof of local convergence for the Adam optimizer," in *2019 International Joint Conference on Neural Networks (IJCNN)*, Budapest, Hungary, Jul. 2019, pp. 1–8, doi: 10.1109/IJCNN.2019.8852239.

[22] S. M. Zaman, Md. M. Hasan, R. I. Sakline, D. Das, and Md. A. Alam, "A comparative analysis of optimizers in recurrent neural networks for text classification," in *2021 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE)*, Brisbane, Australia, Dec. 2021, pp. 1–6, doi: 10.1109/CSDE53843.2021.9718394.

[23] K. M. Hosny, D. Elshoura, E. R. Mohamed, E. Vrochidou, and G. A. Papakostas, "Deep learning and optimization-based methods for skin lesions segmentation: a review," *IEEE Access*, vol. 11, pp. 85467–85488, Aug. 2023, doi: 10.1109/ACCESS.2023.3303961.

[24] I. A. Usmani, M. T. Qadri, R. Zia, F. S. Alrayes, O. Saidani, and K. Dashtipour, "interactive effect of learning rate and batch size to implement transfer learning for brain tumor classification," *Electronics*, vol. 12, no. 4, Feb. 2023, doi: 10.3390/electronics12040964.

[25] Q. Hu, "Evaluation of deep learning models for Kannada handwritten digit recognition," in *2020 International Conference on Computing and Data Science (CDS)*, Stanford, CA, USA, Aug. 2020, pp. 125–130, doi: 10.1109/CDS49703.2020.00031.

[26] H. T. Chandrakala and G. Thippeswamy, "Deep convolutional neural networks for recognition of historical handwritten Kannada characters," in *Frontiers in Intelligent Computing: Theory and Applications*, vol. 1014. 2020, pp. 69–77, 2020, doi: 10.1007/978-981-13-9920-6_7.

[27] K. K. Dutta, A. Herle, L. Appanna, A. Tushar, and K. Tejaswini, "Classification of Kannada handwritten alphabets using multi-class support vector machine with convolution neural networks," in *Intelligent Computing Paradigm and Cutting-edge Technologies (ICICCT 2020)*, Cham, Switzerland: Springer International Publishing, Apr. 2021, pp. 455–463, doi: 10.1007/978-3-030-65407-8_40.

BIOGRAPHIES OF AUTHORS

Ujwala B. S. received her B.E. degree in Electronics and Communication Engineering from Visvesvaraya Technological University Belagavi and M.Tech. first rank with gold medal in Digital Electronics and communication systems from MCE, Hassan under Visvesvaraya Technological University Belgaum, and from 2005 working as an Assistant Professor in Department of ECE, Jawaharlal Nehru New College of Engineering, Shivamogga, India. Her areas of interest are artificial intelligence and Python programming. She can be contacted at email: ujwalaravi2004@jnnce.ac.in.

Dr. Pramod Kumar S. received his B.E. degree in Electronics and Communication Engineering from VEC Bellay and M.Tech. in VLSI design and embedded systems from UTL Technologies Limited, Bangalore. Ph.D. from Visvesvaraya Technological University-RRC Belgaum and working as Department in Electronics and Communication, Jawaharlal Nehru New College of Engineering, Shivamogga, Karnataka, India. He can be contacted at email: pramod.s86@gmail.com.

Dr. H. R. Mahadevaswamy working as Joint Director, Technical Education Division (TED), JSS Mahavidyapeetha, Technical Institution campus, Jagadguru Sri Shivarathreeshwara University. He has received award for his contribution in ‘Continuous Improvement Process-CIP’, distinguished chief guest and key note speaker chief guest and keynote speaker at 3D printing - the future in 2016 at networking & workshop. Published more than 16 research papers. He co-founded a company, ‘Applied Invention India Pvt Ltd’, in 2012. He has worked as a director at Curio System Pvt Ltd. Worked as a Program Manager at Satyam, Hyderabad. Worked as a Group Manager at Robert Bosch Engineering and Business Solutions Ltd. Engineering. He can be contacted at email: drhrmswamy@gmail.com.

Sumathi K. received her B.E. degree in Electronics and Communication Engineering from Visvesvaraya Technological University, Belagavi, in Digital Electronics and Communication Systems from Jawaharlal Nehru New College of Engineering, Shivamogga, under Visvesvaraya Technological University, Belgaum, and is presently working as an assistant professor in Department of ECE, Jawaharlal Nehru New College of Engineering, Shivamogga. Her areas of interest are artificial intelligence and Python programming. She can be contacted at email: sumathik@jnnce.ac.in.