Image Anal Stereol 2021;40:181-191 doi: 10.5566/ias.2586
Original Research Paper

IMPROVED MODEL CONFIGURATION STRATEGIES FOR KANNADA HANDWRITTEN NUMERAL RECOGNITION

GOPAL D. UPADHYE[1], U. V. KULKARNI[2] AND DEEPAK T. MANE[3]

[1]Pimpri Chinchwad College of Engineering, Pune, Maharashtra, 411044, India, [2]Shri Guru Gobind Singhji Institute of Engineering and Technology, Nanded, Maharashtra, 431606, India, [3]JSPM's Rajarshi Shahu College of Engineering, Maharashtra, 411033, India
e-mail: gopalupadhye@gmail.com, uvkulkarni@sggs.ac.in, dtmane@gmail.com
(Received June 9, 2021; revised December 5, 2021; accepted December 7, 2021)

ABSTRACT

Handwritten numeral recognition has been an important area in the domain of pattern classification. The task becomes even more daunting when working with non-Roman numerals. While convolutional neural networks are the preferred choice for modeling the image data, the conception of techniques to obtain faster convergence and accurate results still poses an enigma to the researchers. In this paper, we present new methods for the initialization and the optimization of the traditional convolutional neural network architecture to obtain better results for Kannada numeral images. Specifically, we propose two different methods- an encoder-decoder setup for unsupervised training and weight initialization, and a particle swarm optimization strategy for choosing the ideal architecture configuration of the CNN. Unsupervised initial training of the architecture helps for a faster convergence owing to more task-suited weights as compared to random initialization while the optimization strategy is helpful to reduce the time required for the manual iterative approach of architecture selection. The proposed setup is trained on varying handwritten Kannada numerals. The proposed approaches are evaluated on two different datasets: a standard Dig-MNIST dataset and a custom-built dataset. Significant improvements across multiple performance metrics are observed in our proposed system over the traditional CNN training setup. The improvement in results makes a strong case for relying on such methods for faster and more accurate training and inference of digit classification, especially when working in the absence of transfer learning.

Keywords: Numeral recognition, particle swarm optimization, convolutional autoencode, Kannada numerals.

INTRODUCTION

Pattern classification involves modeling of data to extract relevant features that indicate a particular pattern. Research has been actively pursued on handwritten numeral recognition, an important application of pattern classification. Given an input image, the task objective is to correctly predict one of the ten numerals to which the image belongs. Most of the contemporary methods, including state-of-the-art algorithms, work well on Latin numerals. There is still a vast scope for improvements in the performance of numeral recognition systems on non-English numerals. With the range of data becoming more diverse, systems working on non-English numerals are the need of the hour. (El-Sawy *et al.*, 2017)'(Mane and Kulkarni, 2020)

Kannada is a native Indian language with a rich history spoken in the state of Karnataka and other neighboring regions. The numerals in Kannada are known for their peculiar curves and angles in the writing style.(Prabhu, 2019) Further, every individual will have a different way of writing these numbers. Handwritten Kannada numeral recognition thus becomes a daunting task.(Hu, 2020) The use of contemporary machine learning and deep learning techniques for Kannada numeral recognition algorithms has been limited. (Karthik and Murthy, 2015)'(Shettar *et al.*, 2015)

Majority of previous approaches in this domain have used traditional statistical feature-based methods or preliminary machine learning algorithms (Saini *et al.*, 2021). Deep learning approaches have focused on using basic ANNs or CNNs (Ganesh *et al.*, 2016). Advanced AI techniques are gaining adoption across multiple domains (Ratadiya and Mishra, 2019; Ratadiya *et al.*, 2020). In this paper, we present two different approaches for enhancing the performance of the convolutional neural networks for Kannada handwritten numeral recognition. Specifically, we work on the designing of two methods, one that focuses on improved initialization of the architecture, while the other focuses on improved optimization strategy to decide the best-suited model configuration.

Our main contributions through this paper can be enlisted as follows:

1. We have contributed a new Kannada numeral

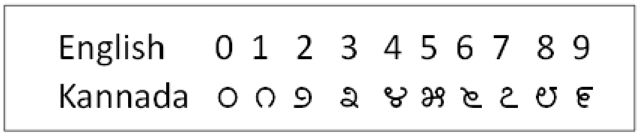
dataset to the community created under a standard setup to promote further research in this area.

2. We use an encoder-decoder set up to improve the weight initialization of the model for better feature extraction from Kannada numerals instead of random initialization.
3. We incorporate an evolutionary algorithm to decide the architecture setup that is otherwise derived empirically in traditional techniques. Thus, any possibility of bias is eliminated. Further, we also evaluate the performance of the proposed approaches on Dig-MNIST, a publicly available standard sized dataset.

The paper structure is as follows: A brief overview of the previous related work and background is done in section 2. The presented techniques are described in depth in section 3. Section 4 focuses upon describing the used datasets while the obtained results are analyzed in section 5. The paper is concluded in section 6.

BACKGROUND AND RELATED WORK

The previous approaches for Kannada languages have relied on the use of derived or handcrafted features or inputs fed to the algorithms. The contributions in the domain of deep learning, especially using advanced convolutional networks have been paltry.



English	0	1	2	3	4	5	6	7	8	9
Kannada	೦	೧	೨	೩	೪	೫	೬	೭	೮	೯

Fig. 1. *Equivalence between English and Kannada numerals.*

The self created dataset of size 20200 samples used in this paper. It contains handwritten samples of handwritten Kannada digits from 0 to 9, which are collected from peoples of the regions where this language is mainly spoken in states of India. The digits written on A-4 sized paper by peoples. Such 505 total samples collected and its corresponding sets created which results into 5050 total handwritten digits. The RGB images of these digits with .jpg extension now converted into gray scale and cropped at size 28 x 28. For transformation, random scaling and vertical, horizontal skewing of 0.5 factor methods applied on these images. Resultant dataset now increased from 5050 to 20200 samples. The similarity mapping between English digit verses Kannada digit represented in Fig. 1.

One of the first approaches for this task used features based on moments to predict the numerals.(Ragha and Sasikumar, 2010) These were passed as input to a feed-forward neural network to train the final system. Kumar (Prasanna Kumar, 2013) presented an approach that involved splitting the input image into four sections and extracting the required components from each of these sections. This approach required very less trainable parameters and inference time but this speed of prediction did take a toll on the accuracy of the system. A clustering approach using the K means algorithm was used by Sheshadri et al. (Sheshadri *et al.*, 2010) for conducting OCR for printed characters in Kannada. Kavya et al. made use of zoning that involves feature extraction from demarcated zones, fused with a classifier to get the final output.(Kavya *et al.*, 2016) Recognition of isolated digits was worked upon by Gurudath and Ravi.(Gurudath and Ravi, 2016) Karthik et al.(Karthik and Murthy, 2015) applied SVM-based kernels and gradient descriptors for this task. Killedar and Deshpande(Killedar, 2015) proposed a template-based matching strategy for recognition as well as translation of Kannada numbers. The nearest neighbor approach has also been tried out to produce satisfactory results.(Shettar *et al.*, 2015) Hallur and Hegadi earlier proposed a holistic approach followed by a feed-forward neural network approach for performing the task. (Hallur and Hegadi, 2014)·(Hallur and Hegadi, 2013) Isolated Kannada numerals were recognized using a framework built on data fusion by Mamatha et al.(Mamatha *et al.*, 2013)

Dhandra et al. made use of zoning to derive features for subsequent recognition of both handwritten and print numerals, and also vowels.(Dhandra *et al.*, 2011)·(Mukarambi *et al.*, 2011) A relatively unorthodox approach involving Fourier descriptor plates and crack codes was used by Rajput (Rajput *et al.*, 2010) to further improve the performance. Recently, deep neural networks were used by Ganesh et al. (Ganesh *et al.*, 2016) for this task, a welcome improvement considering the use of recent architecture.

An observable trend in the majority of previous works is that they have focused on using primitive methods and traditional approaches, not looking beyond these concepts to use contemporary superior architectures or methods. Further, the results achieved have not been up to the mark of those achieved on English or other language numerals.

PROPOSED METHODOLOGIES

We are proposing two different strategies for Kannada handwritten numeral recognition. Given an input image, we predict the class label of the input image into one of the ten possible numeral classes. We elaborate on each of the proposed methodologies in the following subsections:

Approach	SVM (Dhandra *et al.*, 2011)	CNN (Killedar, 2015)	K-NN (Mamatha *et al.*, 2013)	BP-NN (Hallur and Hegadi, 2013)
Dataset samples	1000	200	1000	50
Result(%)	97.4	86.28	91	95

Table 1. *Comparison of previous works in terms of accuracy.*

CONVOLUTIONAL AUTOENCODER FOR BETTER INITIALIZATION

Convolutional autoencoder (CAE) is used as an unsupervised learning algorithm to provide the initialization weights to the CNN architecture. The input images are passed through the autoencoder to reconstruct back the same input at the output layer. This helps in retaining only the relevant features and eliminating any other noise, (Chen *et al.*, 2017) thus providing a better initialization and in turn faster convergence during the CNN training. The Convolutional autoencoder architecture consists of two main modules-the the encoder and the decoder. The reconstruction of input involves first scaling it down to a low dimension intermediate representation and then scaling it back up to its original size. The downscaling is done by the Encoder that itself consists of multiple layers stacked together. The decoder does the job of returning the original dimension reconstructed input. The CAE is trained by mapping the input images to themselves. There is no X to Y mapping, but rather X to X mapping.

The encoder consists of a combination of convolutional layers and pooling layers stacked together such that maximum information retention takes place while still gradually decreasing the dimensions of the input image. Our input image is of dimension 28x28 that we scale down to a size of 7x7 across 16 filters. While convolution layers focus more upon extracting relevant feature maps from the inputs, pooling layers are more responsible for dimension reduction via sub-selection from amongst these feature maps before passing them to the next layers

The decoder consists of a combination of convolutional layers and upscaling layers. Convolutional layers are used to ensure consistent feature extractions. In this stage, we use the same padding during convolution operations to ensure that the dimension does not decrease. In the decoder, the succeeding layers have at least the same dimension as the previous layer.

Algorithm 1 CAECNN

```
Input: Image I($i_w$x$i_h$), Filter f, Learning rate $\alpha$, Number of channels $n_c$, Stride S, padding P, weights w, bias b
Output: Output Feature map
 1: {CAE training}
 2: Initialize weights and bias between 0 and 1
 3: $w_{im} = i_w$ {Reconstructed input width dimension}
 4: $h_{im} = i_h$ {Reconstructed input height dimension}
 5: for each image I in training data do
 6:     read(I)
 7:     downsample(I) {Encoder}
 8:     upsample(I) {Decoder}
 9:     Update weights w and bias b
10: end for
11: Store w and b
12: {Forward propagation of CNN}
13: Initialize weights and bias to w and b values obtained from CAE training
14: $w_u$ = int($(w_{im} - f)$/S) + p
15: $h_u$ = int($(h_{im} - f)$/S) + p
16: for each h,w in $(h_u, w_u)$ pixel do
17:     for each c in $n_c$ channels do
18:         layer_slc = prev_layer[$v_s : v_e, h_s : h_e$,:]
19:         orp = sum(layer_slc * w) + b
20:     end for
21: end for
22: output[output $\leq$ 0] = 0 {ReLU layer}
23: output = max(pool_window(output)) {Max Pooling}
24: layer_slc = prev_layer[$v_s : v_e, h_s : h_e$,:]
25: dw = dw + layer_slc * orp[h,w,c]
26: db = db + orp[h,w,c]
27: w -= $\alpha$ * dw
28: b -= $\alpha$ * db
29: return Output weights
```

Upscaling layer operates exactly opposite to that of the pooling layer. Similar to a pooling window, there is an upscaling window beneath which the nearest neighbors or bilinear sampling is considered to increase the dimension of the image. As this is a function-based upscaling, no separate provision of

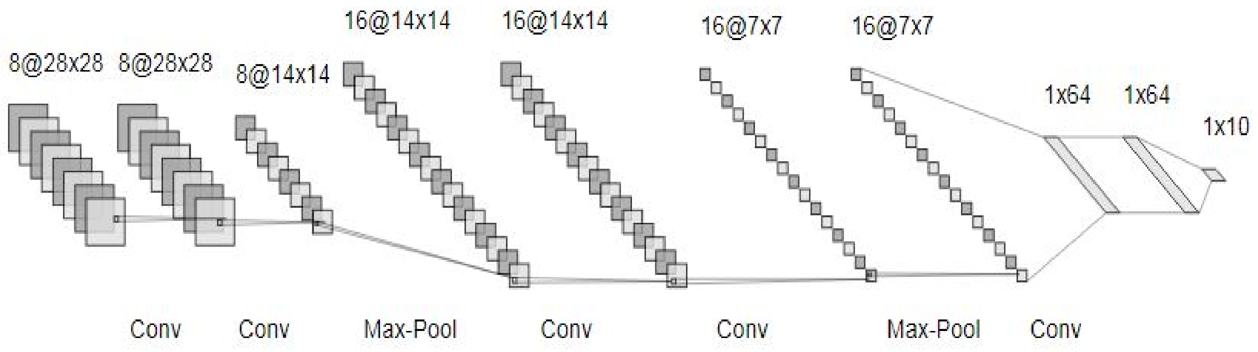


Fig. 2. *Block diagram of proposed CNN architecture used after CAE training.*

training or parameters is required for this layer. As a result, this layer is quite efficient in use and does not add extra complexity to the architecture.

The CCNN architecture contains only the encoder layers, or the first 7 layers of the CAE architecture, further connected to three dense layers. The weights obtained from the training of CAE are used to initialize the CNN architecture and this collectively becomes our final CAECNN topology. The CAE is trained in an unsupervised manner first on a separate subset of the original dataset by mapping the input image to itself in the output layer of the decoder.(Erhan *et al.*, 2010) The model is trained on binary cross-entropy loss and optimized using the Adadelta optimizer. The ReLU (Nair and Hinton, 2010) activation function is used for all layers except the last one, and the Dropout layer is also introduced amongst the dense layers for explicit regularization.(Srivastava *et al.*, 2014)

Table 2 indicates the model summary of the proposed CAE architecture. The first parameter of output shape is set to None as it is based on the batch size. It can be seen that the pooling layers and upsampling layers do not contribute any trainable parameters in the CAE architecture thus making it more efficient. while Fig. 2 depicts the block diagram of the proposed CNN architecture. It can be observed that the first seven layers from the CAE architecture, that constituted the encoder part are connected with the feed-forward dense layers to constitute the final CNN architecture.

PARTICLE SWARM OPTIMIZATION FOR ARCHITECTURE CONFIGURATION

The architecture configuration of CNN is derived iteratively until the best possible configuration is not obtained. This leads to an introduction of human bias as well as an additional time requirement before obtaining the best configuration of the involved layers. Particle Swarm Optimization (PSO) is an evolutionary algorithm that helps to solve this issue. We propose the use of PSO as an optimization technique to derive the ideal CNN architecture.

PSO derives inspiration from the way swarms operate in nature. The algorithm itself consists of two stages that are:

– Communication
– Learning

Table 2. *Model summary of the proposed CAE.*

Layer	**Output shape**	**# of parameters**
Conv layer 1	(None, 8, 28, 28)	80
Conv layer 2	(None, 8, 28, 28)	584
Max layer 1	(None, 8, 14, 14)	0
Conv layer 3	(None, 16, 14, 14)	1168
Conv layer 4	(None, 16, 14, 14)	2320
Max layer 2	(None, 16, 7, 7)	0
Conv layer 5	(None, 16, 7, 7)	2320
Upsampling 1	(None, 784)	0
Conv layer 6	(None, 16, 14, 14)	2320
Conv layer 7	(None, 16, 14, 14)	2320
Upsampling 2	(None, 16, 28, 28)	0
Conv layer 8	(None, 8, 28, 28)	1160
Conv layer 9	(None, 8, 28, 28)	584
Conv layer 10	(None, 1, 28, 28)	73
Total parameters		12,929

Every possible architecture configuration is considered as a particle. In the communication phase, every particle transmits information to all the other particles present in the swarm. The algorithm needs to keep moving towards the best possible configuration i.e. the global minima. If at any stage, it can find a configuration better than the current one, it is rightfully following the concept of better. This is where the learning phase begins, and if the system can move towards a better state, eventually it will achieve the best possible state. (D. T. Mane, 2018) Such a tuned system will be able to work on optimizing problems of any type.(D. T. Mane, 2018)'(Roy *et al.*, 2013)

Various configurations of CNN are possible to solve the problem in the hand of handwritten digit recognition. The PSO algorithm will consider each of them as a particle, and these particles are defined using two different entities or vectors.(Kennedy and Eberhart, 1995)'(D. T. Mane, 2018) One is the position vector which focuses upon the current state coordinates of the particle. The other is the velocity vector which takes into consideration both, the intensity of the particle and also its potential direction.

As we keep moving towards a better state, both the position vector and velocity vector of the particle are updated using the following equations:

$$X_i^{t+1} = X_i^t + V_i^{t+1} \quad (1)$$

$$V_i^{t+1} = wV_i^t + c_1 r_1 (P_i^t - X_i^t) + c_2 r_2 (G_i^t - X_i^t) \quad (2)$$

where X_i^{t+1} is the position vector for next iteration, V_i^{t+1} is the velocity vector for next iteration, t is the current iteration, w is the inertia factor related to the velocity, P is the personal best solution of given particle, G is the global best solution of particle in the population, $c1$ and $c2$ are the learning factors, and $r1$ and $r2$ are random weights in the range of 0 and 1.

PSO keeps adjusting all the possible solutions by modifying their vectors and keeps propagating until it does not find the best solution. Individual changes in velocity also depend upon other particles present in the swarm. The position is updated based on its velocity and current state, and also considering the distance from the personal best P and global best solutions G.

Table 3 indicates the configuration range through which the PSO optimizer searches to find the best architecture. Once these values are derived, the architecture is trained using the train set and tested on the test subset to check precision in its predictions. It can be seen that two different types of searches are possible. The first is to derive the distinct quantity as seen for filters, the second is to derive the best option from a predetermined category of options as in the case of activation functions. We have also fixed certain values as in the case of model train epochs or loss. Algorithm 2 indicates the procedure followed by the optimizer to derive the ideal configuration.

It should be noted that this configuration is not a fixed one and hence we cannot represent it diagrammatically in advance. The optimizer looks to derive these values based on empirical intermediate results. Thus, the training, in this case, differs by the fact that a heuristic-based approach has been used to achieve the ideal goal state (i.e. best possible results) whereas, in the case of traditional CNN, a pre-defined architecture has to be trained and then altered manually every time.

The parameter values related to the updation of optimizers (Eberhart and Kennedy, 1995) are as follows: the number of iterations and the number of particles are both set to 2. The inertia factor is set to the value of 0.7. While both c_1 and c_2 values are set to 2. A 5 fold cross-validation strategy is deployed with a split of train and test set of 4:1.

DATASET DESCRIPTION

The proposed approaches are evaluated on two different sets. The first is the Dig-MNIST dataset, which is one of the few standard-sized test datasets available for Kannada numerals in the public domain.(Prabhu, 2019) Secondly, we also create our dataset following standard practices to demonstrate the generalization ability of our proposed architectures.

The Dig-MNIST dataset consists of over 10000 handwritten digits in the Kannada language.(Prabhu, 2019) It was created using handwritten digits filled by adult volunteers and later scanned to get grayscale images. Cropping and dimension reduction were performed to derive the image size to a resolution of 28x28. Every Kannada numeral has 1024 images in this dataset. Some sample numerals in different font styles are depicted in Fig. 3.

However there are certain shortcomings in this dataset. The dataset size is not very vast enough thus making the training and feature extraction process difficult. While we do evaluate the approaches on the Dig-MNIST dataset, these limitations also encourage us to create a new dataset.

Handwritten Kannada digits were collected from native speakers of varying age categories. An A4-size sheet consisting of a 4x3 rectangular grid was used for the volunteers to write the ten numerals. 505 individuals volunteered for this cause, thereby creating a dataset of a total of 5050 digit instances as images.

Some samples of the handwritten numerals are shown in Fig. 4.

A couple of preprocessing steps were performed on this raw dataset. These include converting the image to grayscale format and also the application of skewing and transformations for data augmentation. These transformations increased the dataset size by four times, leading to 20200 images, 2020 images for each numeral digit. We augment our created dataset further using similar steps to reach a size of 70000 images. Every image in this dataset was now of size 28x28x1, and all of them were combined and saved together in a 70000 x 785 sized multidimensional array in a .csv file. This dataset was more suitable for modeling.

Algorithm 2 Algorithm for PSO optimized CNN

```
Input: Population set PS, Personal best $p_{best}$ and global best $g_{best}$
Output: Velocity vector $v_{vec}$ and Position vector $p_{vec}$
1: Initialize Particle set PS
2: for each iteration do
3:     for a constituent c in PS do
4:         $f_c$ = f(c);
5:         if $f_c > f(p_{best})$ then
6:             $p_{best}$ = c;
7:         end if
8:     end for
9:     $g_{best}$ = best c in PS;
10:    for individual particle c in PS do
11:        $v_{vec}$ = v + c1 * r1 * ($p_{best}$ - c) + c2 * r2 * ($g_{best}$ - c);
12:        $p_{vec} = p_{vec}$ + v
13:    end for
14: end for
15: {Forward propagation of CNN}
16: Initialize the CNN configuration based on PSO output
17: for each h,w in $(h_u, w_u)$ pixel do
18:     for each c in $n_c$ channels do
19:         layer_slc = prev_layer[$v_s : v_e, h_s : h_e$,:]
20:         orp = sum(layer_slc * w) + b
21:     end for
22: end for
23: output[output $\leq$ 0] = 0 {ReLU layer}
24: output = max(pool_window(output)) {Max Pooling}
25: layer_slc = prev_layer[$v_s : v_e, h_s : h_e$,:]
26: dw = dw + layer_slc * orp[h,w,c]
27: db = db + orp[h,w,c]
28: w -= $\alpha$ * dw
29: b -= $\alpha$ * db
30: return Output model
```

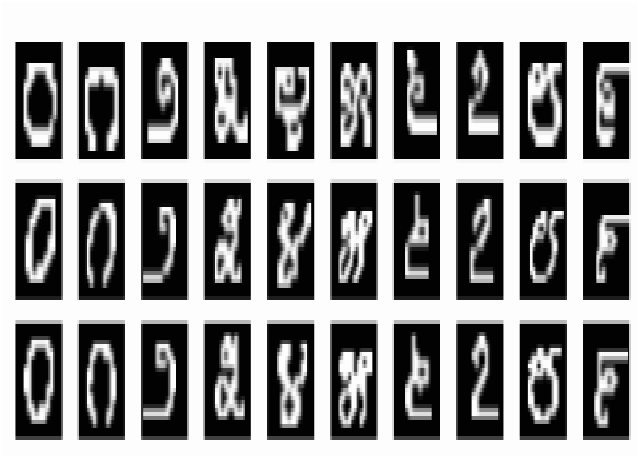


Fig. 3. *Sample renderings of the numerals in different font styles.*

Table 3. *Search configuration of CNN for PSO.*

Model parameter	**Parameter**	**Possible value range**
Conv layer 1	Filters	(3,101)
	Kernel size	[3,5,7]
	Activation	[Sigmoid, ReLU, tanh]
	Input shape	[(28,28,1)]
Max layer 1	Pool size	[(2,2)]
Conv layer 2	Filters	(3,101)
	Kernel size	[3,5,7]
	Activation	[Sigmoid, ReLU, tanh]
Max layer 2	Pool size	[(2,2)]
Flatten layer		
Dense layer 1	Units	(3,101)
	Activation	[Sigmoid, ReLU, tanh]
Dense layer 2	Units	(3,101)
	Activation	[Sigmoid, ReLU, tanh]
Dense layer 3	Units	[10]
	Activation	[Softmax]
Model compile	Loss	[Categorical crossentropy]
	Optimizer	[Adam, SGD]
Model fit	Epochs	[10]
	Batch size	(9,101)
	Verbose	[2]

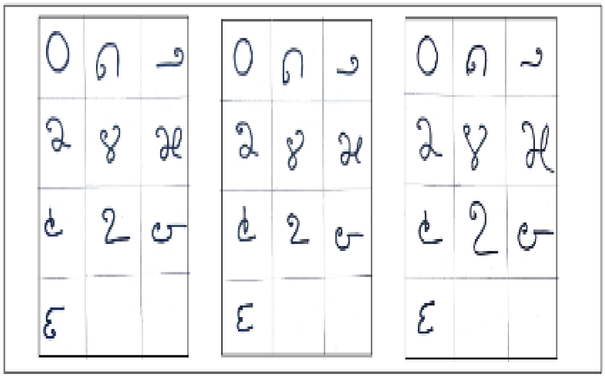


Fig. 4. *Samples of handwritten Kannada digits in the grid.*

Both the datasets are split into training and testing sets by standard procedure. For the CAE approach, in the Dig-MNIST dataset, 7000 images were used for the unsupervised training of the CAE, including 6500 for training and 500 images for validation. Out of the remaining images, 3760 images were used for training the CNN while 1240 were used for testing the CNN. From our created dataset, 45000 images were used for training of CAE, and 5000 images were used for validation while 16000 images were used for training the CNN, and 4000 images were used for testing purposes. In the case of the PSO optimization approach, in our created dataset, 65000 images are used for training while 5000 images are used for testing purposes. From the Dig-MNIST dataset, 8000 images were used for training while 2240 images were used for testing purposes.

RESULTS OBTAINED

We discuss the obtained results separately for each of the two datasets on which the performance of the approaches was evaluated. Results are calculated and analyzed on multiple performance metrics.

RESULTS ON DIG-MNIST DATASET

We compare the total accuracy of both the proposed approaches with standard baseline methods in Table 6. Further, the class-wise accuracy for each of the ten numeral digits are tabulated in Table 4 and Table 5. These numbers give an overview of the generalization ability of the model.

We observe that the class-wise as well as the total accuracy has significantly improved and has crossed the mark of 90% in terms of the overall average accuracy. It can be observed that we have further gained over 2-3% in terms of accuracy with the use of PSO and it has helped the model get considerable gains in its performance. Further, we show the variation in training loss to the number of epochs during the training of the CAE in Fig. 6. A sample reconstructed input obtained from the CAE on the Dig-MNIST dataset is shown in Fig. 5. As a part of ablation studies, we also tabulate the accuracy achieved on each fold during the k-fold cross validation training of the PSO optimized CNN in Table 7. We compare the performance of both our proposed approaches across all the task-related metrics in Table 8.

Table 6. *Comparison of proposed methods with various methods on Dig-MNIST dataset.*

Classifier	Accuracy(%)
Decision tree	72.02
Multilayer Perceptron	86.13
Linear SVM	81.29
Non linear SVM	84.19
LeNet CNN	76.61
CNN (random initialization)	87.02
CAECNN (Proposed)	89.44
CNNPSO (Proposed)	**91.75**

Table 7. *Variation in 5-fold training accuracy of CNNPSO on Dig-MNIST dataset.*

Number of fold during training	Accuracy
1	92.88
2	92.59
3	98.84
4	90.00
5	98.58
Final	92.53

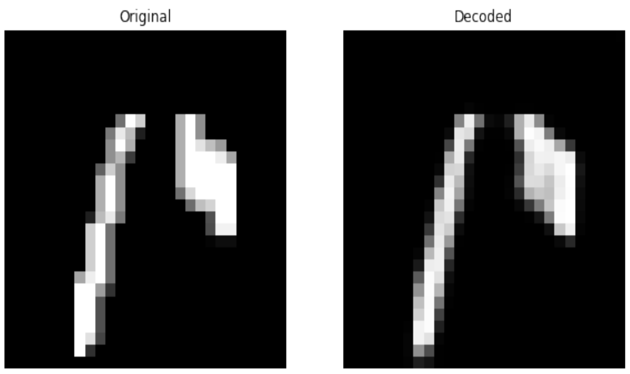


Fig. 5. *Sample input and reconstructed output of CAE for Dig-MNIST dataset.*

Table 4. *Accuracy of the CAECNN approach on Dig-MNIST dataset.*

Dig.	0	1	2	3	4	5	6	7	8	9	**Avg**
Acc.	94.12	91.80	84.62	91.16	86.21	93.08	86.79	89.31	88.37	88.18	89.44

Table 5. *Accuracy of the PSO optimized CNN on Dig-MNIST dataset.*

Dig.	0	1	2	3	4	5	6	7	8	9	**Avg**
Acc.	87.79	92.79	95.87	93.40	88.24	97.24	83.41	94.3	92.75	92.48	91.75

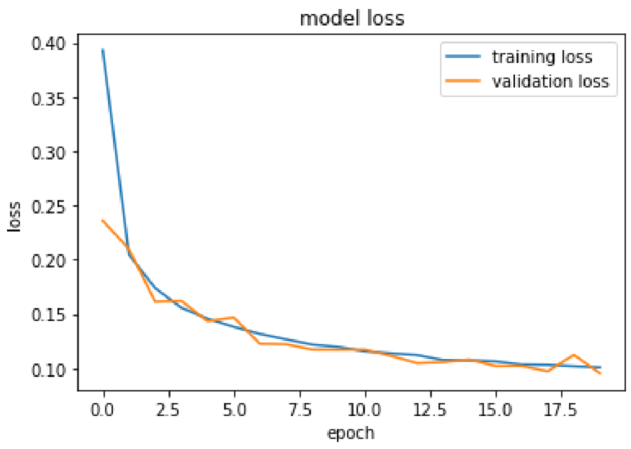


Fig. 6. *Variation in CAE model loss over epochs for Dig-MNIST dataset.*

Table 8. *Results of the proposed approaches across multiple metrics on the Dig-MNIST dataset.*

Approach	CAE+CNN	CNN+PSO
Accuracy (%)	89.44	**91.75**
Precision	0.893	**0.919**
Recall	0.893	**0.917**
F1 score	0.894	**0.917**

RESULTS ON OUR CREATED DATASET

Similar to the previous dataset, the performance of the proposed approaches is calculated and analyzed on all the aforementioned parameters. We start off by tabulating the class-wise accuracies and comparisons in Table 9, 10, and 11 respectively. Variation in CAE model loss over epochs for our dataset is represented in Fig. 7. A sample reconstructed input obtained from the CAE on our created dataset is shown in Fig. 8.

It can be seen that we have achieved the best possible results on both the datasets across all three splits of data as compared to all of the previous approaches. Further, we also see how out of the two proposed approaches in this paper, the second one turns out to be the best in terms of performance on all metrics as shown in Table 8 and 13. Performance of the proposed CAECNN and CNNPSO on both datasets is represented in Table 14, Table 15 respectively.

Table 11. *Comparison of CNNPSO with various methods on our dataset.*

Classifier	**Accuracy(%)**
Decision tree	35.10
Multilayer Perceptron	50.22
Linear SVM	33.18
Non linear SVM	36.64
LeNet CNN	73.26
CNN (random initialization)	88.06
CAECNN (Proposed)	89.56
CNNPSO (Proposed)	**91.66**

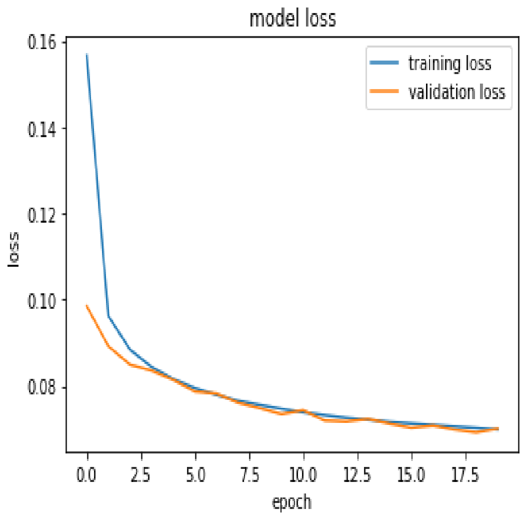


Fig. 7. *Variation in CAE model loss over epochs for our dataset.*

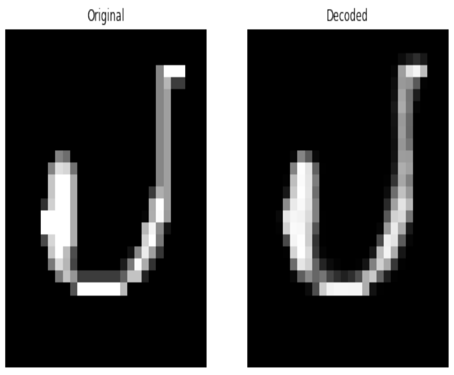


Fig. 8. *Sample input and reconstructed output of CAE for our dataset.*

Table 9. *Accuracy of the CAECNN on our dataset.*

Dig.	0	1	2	3	4	5	6	7	8	9	**Avg**
Acc.	89.47	90.08	89.72	89.17	87.04	93.35	85.68	89.79	91.52	89.96	89.56

Table 10. *Accuracy of the PSO optimized CNN on our dataset.*

Dig.	0	1	2	3	4	5	6	7	8	9	**Avg**
Acc.	94.71	90.93	90.68	92.84	91.37	91.08	92.18	92.7	89.75	90.46	91.66

Table 12. *Variation in 5-fold training accuracy of CNNPSO on our dataset.*

Number of fold during training	**Accuracy**
1	91.78
2	92.33
3	96.81
4	89.50
5	93.14
Final	92.94

Table 13. *Comparison of the two proposed approaches across all metrics on our dataset.*

Approach	CAECNN	CNNPSO
Accuracy (%)	89.56	**91.66**
Precision	0.895	**0.917**
Recall	0.896	**0.916**
F1 score	0.897	**0.915**

Table 14. *Performance of the proposed CAECNN on both datasets.*

Accuracy	**Created dataset**	**Dig-MNIST dataset**
Training accuracy	90.07	90.23
Validation accuracy	88.60	89.72
Test accuracy	89.56	89.44

Table 15. *Performance of the proposed CNNPSO on both datasets.*

Accuracy	**Created dataset**	**Dig-MNIST dataset**
Training accuracy	92.94	92.53
Validation accuracy	91.85	90.59
Test accuracy	91.66	91.75

CONCLUSION

In this paper, multiple new approaches were proposed for the task of handwritten Kannada numeral recognition. These two approaches were inspired by the need to ameliorate issues in model initialization and configuration optimization strategies. We propose the unsupervised convolutional autoencoder as an for the initialization of the CNN architecture. We apply the natural algorithm of particle swarm optimization that focuses on particle space search for deriving the ideal configuration of the CNN architecture. The performance of these proposed systems was compared with the standard techniques used traditionally in the domain. We found the PSO optimized CNN to deliver the best overall results with above 91% accuracy while also generalizing well across all class labels. In the future, transfer learning could be applied as a technique to extend further the initialization of the CNN architecture. There has been limited work on context transfer for Kannada numerals. A new membership function or strategy of configuration space search could be proposed to work upon PSO such that the ideal configuration is obtained even faster. Also, the interpretability of these systems could be improved so that the domain users are provided with more useful explanations for the predictions beyond classifier function values and probabilities.

REFERENCES

Chen M, Shi X, Zhang Y, Wu D, Guizani M (2017). Deep features learning for medical image analysis with convolutional autoencoder neural network. IEEE Trans Big Data 7:750–8.

D. T. Mane UVK (2018). Pattern recognition of iris flower using neural network based particle swarm optimization. International Journal of Computer Sciences and Engineering 6:916–20.

Dhandra B, Mukarambi G, Hangarge M (2011). Zone based features for handwritten and printed mixed kannada digits recognition. In: IJCA Proceedings on International Conference on Vlsi Communications and Instrumentation.

Eberhart R, Kennedy J (1995). A new optimizer using particle swarm theory. In: Proceedings of the Sixth International Symposium on Micro Machine and Human Science.

El-Sawy A, Loey M, El-Bakry H (2017). Arabic handwritten characters recognition using convolutional neural network. WSEAS Trans Comp Res 5:11–9.

Erhan D, Courville A, Bengio Y, Vincent P (2010). Why does unsupervised pre-training help deep learning? In: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics.

Ganesh A, Jadhav AR, Pragadeesh KC (2016). Deep learning approach for recognition of handwritten kannada numerals. Adv Intell Syst :294–303.

Gurudath K, Ravi D (2016). Isolated digits recognition in kannada language. Int J Comput Appl 140(10):23–9.

Hallur VC, Hegadi R (2013). Kannada handwritten digits recognition: neural network approach. Int J Sci Res :417–9.

Hallur VC, Hegadi R (2014). Offline kannada handwritten numeral recognition: holistic approach. In: Proceeding of Second International Conference on Emerging Research in Computing, Information, Communication and Applications.

Hu (2020). Evaluation of deep learning models for kannada handwritten digit recognition. In: International Conference on Computing and Data Science.

Karthik S, Murthy KS (2015). 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.

Kavya T, Pratibha V, Priyadarshini B, Vijaya Bharathi M, Vijayalakshmi G (2016). Kannada characters and numerical recognition system using hybrid zone-wise feature extraction and fused classifier. Int J Eng Res Technol 5:506–10.

Kennedy J, Eberhart R (1995). Particle swarm optimization. In: Proceedings of ICNN'95-International Conference on Neural Networks, vol. 4.

Killedar ea (2015). Kannada handwritten numerals recognition and translation using template matching. International Journal on Recent Technologies in Mechanical and Electrical Engineering 2:77–80.

Mamatha H, Srirangaprasad S, Srikantamurthy K (2013). Data fusion based framework for the recognition of isolated handwritten kannada numerals. Int J Adv Comput Sci Appl 4:174–82.

Mane D, Kulkarni UV (2020). A survey on supervised convolutional neural network and its major applications. In: Deep Learning and Neural Networks: Concepts, Methodologies, Tools, and Applications. IGI Global, 1058–71.

Mukarambi G, Dhandrab V, Hangarge M (2011). Recognition system for handwritten and printed kannada numerals and vowels. Int J Mach Intell 3:259–62.

Nair V, Hinton GE (2010). Rectified linear units improve restricted boltzmann machines. In: ICML.

Prabhu VU (2019). Kannada-mnist: A new handwritten digits dataset for the kannada language. arXiv preprint arXiv190801242 .

Prasanna Kumar K (2013). Algorithm to identify kannada vowels using minimum features extraction method. International Journal of Innovative Technology and Exploring Engineering 5:79–84.

Ragha LR, Sasikumar M (2010). Adapting moments for handwritten kannada kagunita recognition. In: Second IEEE International Conference on Machine Learning and Computing.

Rajput G, Horakeri R, Chandrakant S (2010). Printed and handwritten kannada numeral recognition using crack codes and fourier descriptors plate. International Journal of Computer Application IJCA On Recent Trends in Image Processing and Pattern Recognition RTIPPR :53–8.

Ratadiya P, Asawa K, Nikhal O (2020). A decentralized aggregation mechanism for training deep learning models using smart contract system for bank loan prediction. ARXIV PREPRINT ARXIV201110981 .

Ratadiya P, Mishra D (2019). An attention ensemble based approach for multilabel profanity detection. In: IEEE International Conference on Data Mining Workshops.

Roy A, Dutta D, Choudhury K (2013). Training artificial neural network using particle swarm optimization algorithm. Int J Adv Res Comput Sci Softw Eng 3.

Saini A, Daniel S, Saini S, Mittal A (2021). Effkannadares-next: An efficient residual network for kannada numeral recognition. Multimed Tools Appl 3.

Sheshadri K, Ambekar PKT, Prasad DP, Kumar RP (2010). An ocr system for printed kannada using k-means clustering. In: IEEE International Conference On Industrial Technology.

Shettar S, Basavaprasad B, Bhagya H (2015). Recognition of printed kannada numerals by nearest neighbor method. In: Proceedings of the International Conference on Computational Systems for Health Sustainability.

Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014). Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 15:1929–58.