Design Of Cnn Based Model For Handwritten Digit Recognition Using Different Optimizer Techniques

Ayush Kumar Agrawal[1,a *],Vineet Kumar Awasthi[1,b] ,Kranti Kumar Dewangan[2,d],Deepesh Dewangan[3,e]and Sameera Khan[4,f]
[1]Department of IT & CS, Dr. C. V. Raman University, Chhattisgarh, India
[2]AAFT University, Chhattisgarh, India.
[3] Department of Computer Science, Shri Rawatpura Sarkar University, (C.G.), India.
[4]Department of Computer Science and Engineering, Amity University, (C.G.), India.
***corresponding author: ayushagrawal369@gmail.com,**
[b] vineet99kumar@gmail.com, [c]kranti.kumar@aaft.edu.in, [d]deepeshdewangan@sruraipur.ac.in,
[e]skhan@rpr.amity.edu

Article History: Received: 11 January 2021; Revised: 12 February 2021; Accepted: 27 March 2021; Published online: 10 May 2021

Abstract. Convolutional Neural Network (CNN) is the well-known technique for feature extraction capability. But the poor selection and setup of design parameters restricts its performance during the training. The necessity of effective CNN design is required in various fields such as banking, security and in digital documentation to recognize the specific handwritten pattern. In this direction, we design a custom CNN model to precisely recognize the handwritten digit using different set of optimizers. The behavior of the presented approach has been experimented on the public MNIST dataset. The results show the effectivity of the model outperforms several state-of-the-art techniques in the presented field.

Keywords: Convolutional Neural Network, Deep Learning, Handwritten Digit Recognition, Optimizers.

Introduction

Handwritten digit recognition portrays a noteworthy contribution in various user verification applications in the current scenario. Similarly, the pattern from the handwritten digits reveals the uniqueness of a person's identity and mostly utilized in authentication purposes. As the real time handwritten formations for character or digits do not always represent their uniform attributes in context of their style, size, shape and orientation. Hence, these challenges have to be tackled to which can be utilized in many digital applications too. The purpose of such a recognition scheme is to translate the handwritten characters into some readable format that a machine can understand. Some of the applications under this domain can be characterized using license-plate of vehicle's recognition, services governed by postal letter, Cheque transition system (CTS), and old documents which involve automation in bank and libraries. These areas have huge databases and henceforth claim high processing accuracy, and requires minimum complexity. Over the years, various techniques such as computer vision [1], [2], image processing [3]–[9] and deep learning based techniques [10], [11] has received a lot of attention from many researchers. Out of which CNN has been considered to identify some of the important visual patterns straight from the pixel level representations with nominal preprocessing, and are being proficient to spot the variable and different patterns. Emphasis is put on to obtain the significant feature capability for the target object. However, the task becomes challenging when the target object has a diverse variation in its representation. It is the intrinsic dissimilarity in writing which makes the problem more challenging at different instances. Hence, structuring a universal model capable enough to identify handwritten digits by different writers is not always practical. Still, obtaining the most significant features through the capability to advance the classification accuracy that too with the minimum complexity is one of the most vital issue in this problem domain. To provide an optimal model which not accurately recognize handwritten digits but also ensures the optimality using different optimizers, a CNN based handwritten digit recognition system has been proposed that uses seven different optimizers and built among best of them.

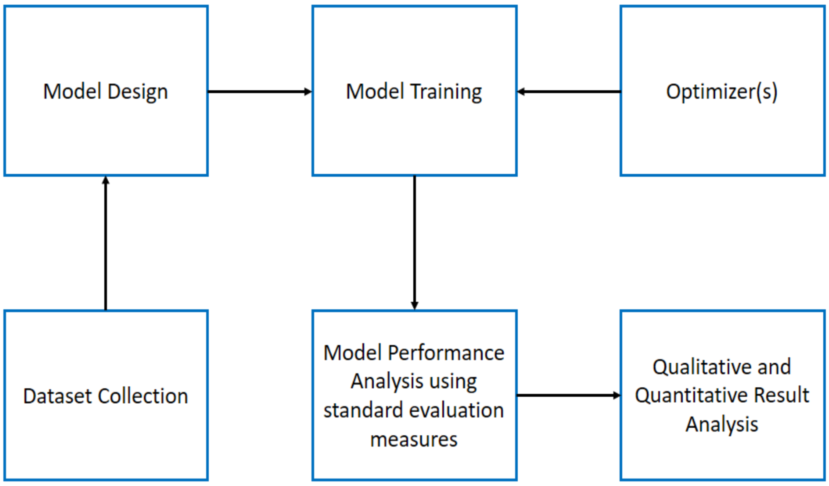


Figure 1: Process Flow of the offered handwritten digit recognition system.

Process flow for the proposed approach has been illustrated in figure 1. Organization for the rest of the paper is as follows: The section related works discusses the various state-of-the-art techniques in handwritten digit recognition domain. Proposed techniques along with the implementation details has been presented in the section material and methods. Further, the study of implementation, results and comparison with the standard techniques have been given in the discussion section. The concluding remarks and future direction has been mentioned in the last section.

Related Works

ShoponMd et al. proposed a model for recognition of the handwritten digit database [12]. Here author used blocky artifact augmentation technique with the deep convolutional neural network which increases the accuracy of the recognition of handwritten digits. In this paper, author used this blocky artifact technique with three types of database namely as MNIST Dataset, CMATERDB 3.1.1 Dataset, Indian Statistical Institute (ISI) Dataset and achieved excellent performance with it. With the blocky artifact author achieved accuracy 99.56% with MNIST, 99.83% with CMATERDDB and 99.35% with ISI dataset. But the accuracy achieved with MNIST is poor than the previous work without blocky artifact, so with blocky artifact author got excellent result with CMATERDDB and ISI dataset. BoukharoubaAbdelhak et al. presents the process of Persian numerals handwritten digit recognition by using the classification technique support (SVM) [13]. Here feature extraction and feature selection technique has been used with the SVM. Feature extraction technique does not require the normalization of digits. In this paper, chain code histogram (CCH) and white-black transition scheme has been used for feature extraction. Here the Farsi handwritten dataset has been used for the experiment named “HODA”. This dataset consists of ten digit classes from 0 to 9. Here author evaluated 80000 data and achieved (higher accuracy compared to other existed method) the excel result.

Wang Zi-Rui used the concept of deep neural network –hidden markov model (DNN-HMM) for proposing the new approach writer adaptation based DNN-HMM model for recognition of handwritten Chinese text datasets [14]. Here authors applied the new proposed writer adaptation DNN-HMM technique and also writer-independent DNN-HMM technique on mostly used ICDAR 2013 Chinese handwriting competition database and achieved the tremendous result, which shows the new writer-adaptation DNN-HMM achieved the better recognition rate compared to writer-independent DNN-HMM. Karimi Hossein et al. presents the recognition of Persian handwritten digits dataset by new proposed method [15]. Basically the new proposed method for Persian handwritten recognition contains three steps like pre-processing, feature extraction and classification where matlab coding has been used in pre-processing stage and ensemble classifier scheme and WEKA open source application has been used for classification. Experiment has been performed on Tarbiatmodares University (TMU) digits database and achieve the best recognition rate of Persian handwritten digits, was 95.280%. Zhang Xu-Yao et al. proposed a framework for recognizing the Chinese character dataset and also for writing the Chinese characters by using recurrent neural network [16]. This proposed model is the combination of two models like: discriminative model (for recognition) and generative model (for writing). With the LSTM and GRU author achieved excelled performance for recognition of the ICDAR-2013 competition database and with the discriminative model achieved high accuracy for drawing the characters. AshiquzzamanAkm et al. proposed a deep learning neural network model for recognition of handwritten Arabic characters [17]. Here author used the database for Arabic handwritten numerals are CMATERDB Arabic handwritten digit dataset. In the experiment author got excellent result 97.4 percent accuracy with CMATERDB Arabic handwritten digit dataset by proposed deep neural network model, which is the highest accuracy of recognition of CMATERDB Arabic handwritten digit dataset. Similarly, Aamir et al. used the contractive auto encoder CAE to obtain the significant and high-level representation without the backpropagation approach [18]. Afterwards, Softmax classifier was utilized in the last layer of the trained model for classification. Likewise, Gati et al. applied a custom model under skip CNN architecture, where the attained accuracy was mentioned as 96.8% on MNIST dataset [19]. The input had processing of $3x3$ filter and a fully connected layer with 128 neurons which utilizes the Softmax activation function. In this direction, Zaheer et al. had conducted the model training using different set of optimizers and attained the highest accuracy of 98.26% on MNIST dataset but observed a little instability in the training network [20]. Another CNN based approach had been marked for handwritten digit recognition where using tensor board platform was utilized and accuracy of 95.7% is attained during training [21]. Kaziha et al. had implemented long short term memory (LSTM) and quantized convolution based techniques and tested the performance on MNIST dataset with 99.45% accuracy [22]. However, chances of data overfitting is observed during training which was conducted for only 50 epochs. Garg et al. had experimented the performance of CNN using different learning rates and achieved 98.54% of accuracy [23].

Having various approaches in this direction, we observe that a steady design of CNN is required to extract the significant features of handwritten digits. In this direction, we design a convolutional neural network based model that uses seven different optimizers and obtains the desired accuracy with stable model training.

Material and Methods

There are various phases to be considered in handwritten digit recognition which have already been picturized in figure 1. These will be discussed in this section.

Dataset Collection. MNIST is well known database for handwritten digit recognition used by various researchers [24] and has a total number of 70,000 samples which is divided into a set of 60,000 training samples and a set of 10,000 test samples. The dataset is already preprocessed, hence requires minimum effort to arrange it according to specific application. In MNIST dataset example are in the form of magic number where magic number means each example starts form 00.

Model Design. The efficiency of the model depends on the feature extraction capability which corresponds to the high, middle and low level representation of the image data. To learn features of MNIST dataset, customized convolutional neural network based model has been considered which is developed and designed to classify the MNIST handwritten digits. For this, three steps have been involved which are convolution, activation and pooling followed by a fully connected layer that assist the model to classify among ten digits (0 to 9). Total of 4 convolutional layers have been considered, which has undergone through number of filters of the 3*3 size. These filters fetches and stores the boundary and texture information from the image. But not all the extracted information are significant. To provide the weightage to all the significant participating elements with minimized complexity, maximum pooling strategy has been adopted. Similarly, activation function decides which neurons has to be fired at a particular moment. In this direction, rectified linear unit is considered in the previous layers while softmax function is applied in the last dense layer where all the neurons are connected with each other. Detailed configuration of the proposed model is given in table 1.

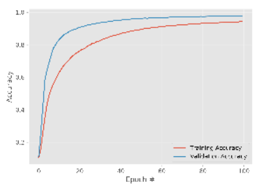
Model Training. Most of the CNN training requires processing to be done in forward and backward phase. During the first phase of forward stage, each considered layers will cache any data which will be needed for the backward phase. Similarly, during the second phase of backward stage, each of the layer will obtain and also yield a gradient. It will receive the loss of gradient against its corresponding output ($\partial L / \partial out$) and yield the loss of gradient against its input ($\partial L / \partial in$). In this stage, after loading the required data, the dataset is divided into two major parts training and testing. The entire dataset is randomly partitioned into 70% of training and 30% of testing data for the validation of our training. Once this step is complete then we will have the data representation for flatten the input data image whose dimensions is too converted into 1D (pixels in width x and pixels in height) followed by normalizing the input image pixel value. Afterwards one hot encoding is carried out for the categorical classification against each digit (0 to 9).For the training module, we have considered a batch size of 64, learning rate of .001, stride of unit movement in both horizontal and vertical direction, padding of similar dimension and epoch of 100 against every optimizer applied.

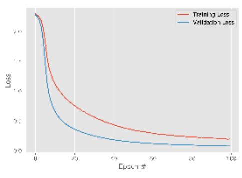
Table 1: Configuration detail of proposed CNN Model

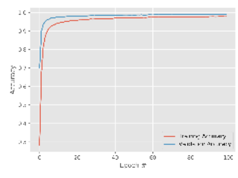
Layers	Output Shape	Parameters
Convolutional	(None, 26, 26, 32)	320

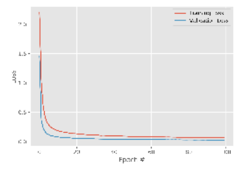
Activation	(None, 26, 26, 32)	0
Convolutional	(None, 24, 24, 32)	9248
Activation	(None, 24, 24, 32)	0
Max Pooling	(None, 12, 12, 32)	0
Convolutional	(None, 10, 10, 64)	18496
Activation	(None, 10, 10, 64)	0
Convolutional	(None, 8, 8, 64)	36928
Activation	(None, 8, 8, 64)	0
Max Pooling	(None, 4, 4, 64)	0
Flatten	(None, 1024)	0
Dense	(None, 512)	524800
Activation	(None, 512)	0
Dropout	(None, 512)	0
Dense	(None, 10)	5130
Activation	(None, 10)	0
Total Parameters		594922

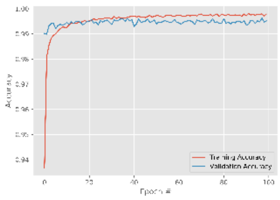
Optimizers. Throughout the model training, the error backpropagation assessments the error quantity against which the node weights in the network are accountable. As an alternative of apprising the weight, it is computed and scaled up by the rate of learning. Precisely, an average weighted which is exponentially growing or shrinking of the previous updates is considered to when the weights are continually updated. This change is said to be the “momentum” and increases the inertia to the update process in one direction. Moreover, effective execution of a feature learned model involves elevating numerous constraints by inspecting and regulating the rate of learning and decay rate in every mentioned layers. In this framework, seven different optimizers [11] have been applied and analyzed to obtain the optimal one for the proposed scenario.

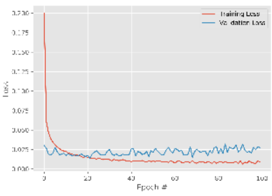

(a) Adadelta Accuracy

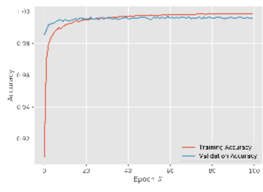

(b) Adadelta Loss

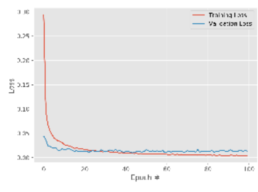

(c) Adam Accuracy

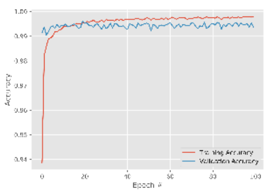

(d) Adam Loss

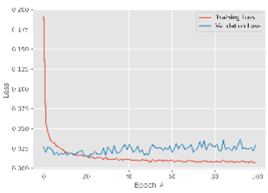

(e) Adagrad Accuracy

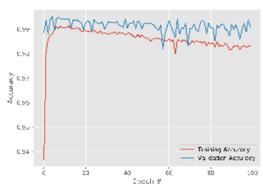

(f) Adagrad Loss

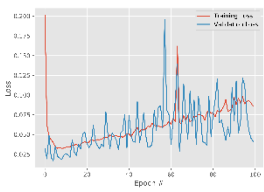

(g) Adamax Accuracy

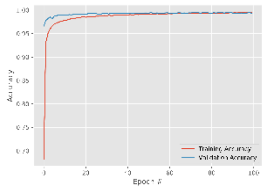

(h) Adamax Loss


(i) Nadam Accuracy

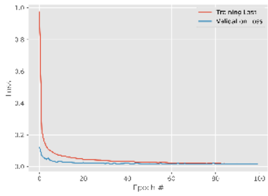

(j) Nadam Loss


(k) RMSProp Accuracy


(l) RMSprop Loss



(m) SGD Accuracy



(n) SGD Loss

Figure 2: Training performance of the proposed CNN model using different optimizers

Experimental Results and Discussion

For the handwritten digit recognition, the model has been setup under the google colab environment and the performance of model has been analyzed using GPU capability having 2496 CUDA cores , 12GB GDDR5 VRAM. The stopping criteria for the training is fixed on 100 epochs against the learning rate of .001. Along with the model training, the computation time has also been considered to analyze the impact of each optimizer on the model training.

Effect of different optimizers in the model training. The model learn noteworthy features during training and shows its effectivity. However, not all the mentioned optimizers attain the same performance and have marginal difference in their behavior and it can be observed from figure 2, where all the optimizers are represented with their training accuracy and loss graph. The highest level in the accuracy graph indicates the maximum accuracy attained during training. Similarly, the lower range reveals in the loss graph that the loss is gradually reducing. The fluctuation in the graph shows that the model has not been in the stable stage during training. Apart from these observations, the gap between training and loss in both accuracy and loss should be as close as possible to ignore the chances of overfitting and under fitting during training. With these close observation, it is found that behavior of Adam optimizer is found suitable as it has attained the maximum accuracy and that too in earlier epoch stages, and has low rate of loss and has been stable throughout the training period. Most of the other optimizers behaving in the similar fashion but some of them have instability such as RMSProp (figure 2 (k-l)). Also the problem of little overfitting has also been observed in case of Adagrad and Nadam optimizer (figure 2 (e-f) and (i-j)). Performance of SGD also looks promising and are important too, but in context of computation time, not all the optimizers are achieving the maximum accuracy within the stipulated time interval.However, to validate these qualitative performance, we have also incorporated the standard evaluation quantitative measures including accuracy with the computation time and has been represented in table 2. It is worth mentioning that the quantitative behavior in terms of confusion matrix for all the optimizer in the proposed approach has also been incorporated and shown in figure 3.

```
[ 970    0    0    0    1    2    2    2    2    1]
[   0 1123    4    0    0    0    1    1    6    0]
[   4    0 1005    2    1    0    1   13    6    0]
[   0    0    0  991    0    6    0    3   10    0]
[   1    0    1    0  955    1    2    1    3   18]
[   2    1    0    5    0  875    2    1    4    2]
[   7    3    0    0    2    3  941    0    2    0]
[   1    3   14    4    0    0    0  999    3    4]
[   4    0    3    1    2    6    0    3  951    4]
[   7    6    1    9    3    5    1    5    5  967]
```

(a) Adadleta

```
[ 978    0    0    0    0    0    0    1    1    0]
[   0 1130    1    1    0    0    1    1    1    0]
[   1    0 1023    0    1    0    0    4    3    0]
[   0    0    0 1007    0    1    0    0    2    0]
[   0    0    0    1  971    0    1    0    1    8]
[   0    0    0   12    0  877    1    1    1    0]
[   2    2    1    1    1    3  945    0    3    0]
[   0    1    4    2    0    0    0 1020    0    1]
[   1    0    2    4    1    3    0    2  959    2]
[   0    3    0    2    3    3    1    3    0  994]
```

(b) Adagrad

```
[ 977    0    0    0    0    0    2    1    0    0]
[   0 1133    0    0    0    0    1    0    0    1]
[   0    0 1025    2    0    0    0    5    0    0]
[   0    0    0 1007    0    2    0    1    0    0]
[   0    0    0    0  975    0    0    0    0    7]
[   0    0    0    5    0  886    1    0    0    0]
[   1    2    3    0    1    1  949    0    1    0]
[   0    2    0    0    0    0    0 1026    0    0]
[   0    0    0    2    0    1    0    0  970    1]
[   0    0    0    0    4    0    0    0    1 1004]
```

(c) Adam

```
[ 976    0    0    0    0    1    1    0    2    0]
[   0 1132    0    0    0    0    0    3    0    0]
[   1    1 1028    1    0    0    0    1    0    0]
[   0    0    1 1004    0    3    0    0    2    0]
[   0    0    0    0  979    0    0    0    1    2]
[   0    0    0    1    0  889    1    0    1    0]
[   1    1    0    0    0    0  954    0    2    0]
[   0    1    2    1    0    0    0 1024    0    0]
[   0    0    1    1    0    0    0    0  972    0]
[   0    0    0    1    6    2    0    0    0 1000]
```

(b) Adamax

```
[ 975    0    0    0    0    1    0    2    2    0]
[   0 1127    0    2    1    0    4    1    0    0]
[   1    0 1025    3    0    0    0    3    0    0]
[   0    0    0 1008    0    2    0    0    0    0]
[   0    0    0    0  973    0    1    0    0    8]
[   0    0    0    1    0  889    2    0    0    0]
[   2    1    0    0    1    1  951    0    2    0]
[   0    2    2    0    0    1    0 1022    0    1]
[   0    0    1    2    0    1    0    0  969    1]
[   0    0    0    0    3    4    0    3    1  998]
```

(e) Nadam

```
[ 968    0    3    0    0    2    1    1    3    2]
[   2 1122    1    0    0    0    4    4    2    0]
[   0    1 1023    0    0    0    0    7    1    0]
[   0    0    0 1006    0    1    0    2    1    0]
[   0    0    0    0  973    0    0    1    0    8]
[   0    0    0    3    0  884    4    1    0    0]
[   1    3    0    0    2    5  945    0    2    0]
[   0    2    2    1    0    0    0 1019    0    4]
[   0    1    1    1    0    0    1    0  970    0]
[   0    0    0    0    3    2    0    2    2 1000]
```

(f) RMSProp

```
[ 979    0    0    0    0    0    0    1    0    0]
[   0 1130    0    1    0    0    2    1    0    1]
[   0    0 1030    1    0    0    0    1    0    0]
[   0    0    1 1002    0    3    0    1    1    2]
[   0    0    0    0  976    0    0    1    0    5]
[   1    0    0    2    0  888    1    0    0    0]
[   3    0    1    0    1    2  950    0    1    0]
[   0    1    6    0    0    0    0 1021    0    0]
[   1    0    3    1    0    1    0    0  967    1]
[   0    1    0    0    5    4    0    2    0  997]
```

(g) SGD

Figure 3: Confusion matrix validation for all the optimizers used in the proposed model

Table-2: Performance evaluation of the proposed approach using different optimizers

Optimizers	Accuracy	Time (minutes)
Adam	99.60%	3544.15
RMSProp	99.10%	4112.70
SGD	99.40%	3568.10
Nadam	99.37%	5037.82
Adagrad	99.04%	5267.87
Adadelta	97.77%	4272.73
Adamax	99.58%	3587.38

Results of handwritten digit recognition during training of the proposed model. During the training module, performance of Adam optimizer has been identified optimal in both accuracy and computation time, so this is equipped with the final model and following figures showing some detection results which are attained through after training the model.

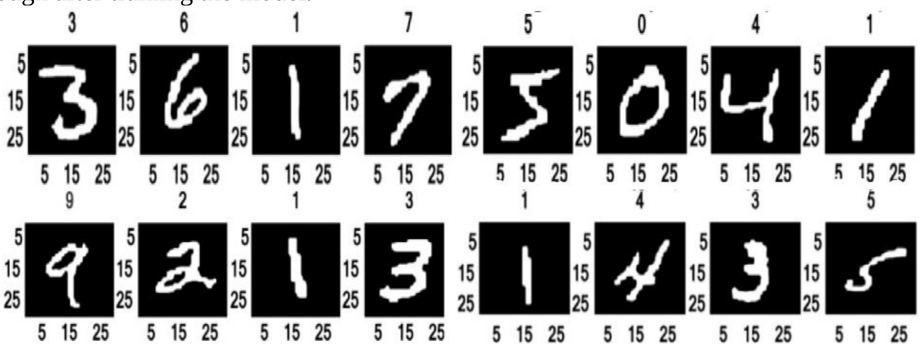


Figure 4: Detection result of handwritten digit model for the proposed approach using Adam optimizer.

Comparison of the proposed approach with state-of-the-art methods. The outcome using various optimizers have been considered in this study and the performance of the Adam optimizer of the proposed approach is observed optimal as given in table 2. Various state-of-the-art techniques using different approaches have been considered in this work. Gati et al. designed a customized model and used skip CNN design and achieved the accuracy of 96.8% on MNIST dataset [19]. It was mentioned that the input layer had handling of 3x3 kernel which utilizes the Softmax activation function. Likewise, Zaheer et al. had accompanied the model training by means of different set of optimizers and achieved the maximum accuracy of 98.26% on MNIST dataset but suffered from the network instability during training [20]. Similarly, another CNN based methodology had been spotted for handwritten digit recognition where using tensor board platform and get an accuracy of 95.7% [21]. Kaziha et al. had employed long short term memory (LSTM) and quantized convolution based techniques and tested the performance on MNIST dataset with 99.45% accuracy [22]. Though, overfitting during training was observed conducted for only 50 epochs. Garg et al. had experimented the performance of CNN using different learning rates and achieved 98.54% of accuracy [23]. In summary, table 3 represents the quantitative analysis and it is clearly found that our proposed approach outperforms the standard studies in the presented work and achieved accuracy of 99.60%.

Table- 3: Performance comparison of the of the proposed approach with state-of-the-art techniques

Methods	**Accuracy**
Gati et al [19]	96.80%
Zaheer et al [20]	98.26%
Ge et al [21]	95.70%
Kaziha et al [22]	99.45%
Garg et al [23]	98.54%
Proposed Model	99.60%

Conclusion

In this paper, we have developed a CNN model for handwritten digit recognition using 4 convolutional layers. Numerous convolution layers yield a various kind of features. Further, 32 filters with the size 3x3 in input layers and seven different optimizers have been considered to extract the optimal features and obtained the accuracy of 99.60% on standard data. In the future work, our aim is to build the hybrid CNN model to deal with noisy data and solving memory issues to make the model more robust.

References

1. D. K. Dewangan and S. P. Sahu, “Driving Behaviour Analysis of Intelligent Vehicle System for Lane Detection using Vision-Sensor,” IEEE Sens. J., vol. 21, no. 5, pp. 6367–6375, 2020, doi: 10.1109/JSEN.2020.3037340.
2. D. K. Dewangan and S. P. Sahu, “Real time object tracking for intelligent vehicle,” 2020 1st Int. Conf. Power, Control Comput. Technol. ICPC2T 2020, pp. 134–138, 2020, doi: 10.1109/ICPC2T48082.2020.9071478.
3. D. K. Dewangan and S. P. Sahu, “Deep Learning-Based Speed Bump Detection Model for Intelligent Vehicle System Using Raspberry Pi,” IEEE Sens. J., vol. 21, no. 3, pp. 3570–3578, 2021, doi: 10.1109/JSEN.2020.3027097.
4. P. Pandey, K. K. Dewangan, and D. K. Dewangan, “Enhancing the quality of satellite images by preprocessing and contrast enhancement,” in Proceedings of the 2017 IEEE International Conference on Communication and Signal Processing, ICCSP 2017, 2018, vol. 2018-Janua, doi: 10.1109/ICCSP.2017.8286525.
5. P. Pandey, K. K. Dewangan, and D. K. Dewangan, “Enhancing the quality of satellite images using fuzzy inference system,” in 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing, ICECDS 2017, 2018, doi: 10.1109/ICECDS.2017.8390024.
6. [6] P. Pandey, K. K. Dewangan, and D. K. Dewangan, “Satellite image enhancement techniques - A comparative study,” in 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing, ICECDS 2017, 2018, doi: 10.1109/ICECDS.2017.8389506.
7. Deepak Kumar Dewangan; Yogesh Rathore, “Image Quality estimation of Images using Full Reference and No Reference Method,” Int. J. Adv. Res. Comput. Sci., vol. 2, no. 5, pp. 323–326, 2011.
8. D. Dewangan and Y. K. Rathore, “Image Quality Costing of Compressed Image Using Full Reference Method,” vol. 1, no. February, pp. 68–71, 2016.
9. Deepak Kumar Dewangan; Yogesh Rathore, “Image Quality Evaluation of Images of Different Formats,” Int. J. Comput. Sci. Secur., vol. 1, no. 3, 2011.
10. D. K. Dewangan and S. P. Sahu, “PotNet: Pothole detection for autonomous vehicle system using convolutional neural network,” Electron. Lett., pp. 2–5, 2020, doi: 10.1049/ell2.12062.
11. Deepak Kumar Dewangan; Satya Prakash Sahu, “RCNet: road classification convolutional neural networks for intelligent vehicle system,” Intell. Serv. Robot., 2021, doi: https://doi.org/10.1007/s11370-020-00343-6.
12. M. Shopon, N. Mohammed, and M. A. Abedin, “Image augmentation by blocky artifact in Deep Convolutional Neural Network for handwritten digit recognition,” 2017 IEEE Int. Conf. Imaging, Vis.

Pattern Recognition, icIVPR 2017, 2017, doi: 10.1109/ICIVPR.2017.7890867.

13. A. Boukharouba and A. Bennia, "Novel feature extraction technique for the recognition of handwritten digits," Appl. Comput. Informatics, vol. 13, no. 1, pp. 19–26, 2017, doi: 10.1016/j.aci.2015.05.001.
14. Z. R. Wang and J. Du, "Writer code based adaptation of deep neural network for offline handwritten Chinese text recognition," Proc. Int. Conf. Front. Handwrit. Recognition, ICFHR, vol. 0, pp. 548–553, 2016, doi: 10.1109/ICFHR.2016.0106.
15. H. Karimi, A. Esfahanimehr, M. Mosleh, F. M. J. Ghadam, S. Salehpour, and O. Medhati, "Persian Handwritten Digit Recognition Using Ensemble Classifiers," Procedia Comput. Sci., vol. 73, no. Awict, pp. 416–425, 2015, doi: 10.1016/j.procs.2015.12.018.
16. X. Y. Zhang, F. Yin, Y. M. Zhang, C. L. Liu, and Y. Bengio, "Drawing and Recognizing Chinese Characters with Recurrent Neural Network," IEEE Trans. Pattern Anal. Mach. Intell., vol. 40, no. 4, pp. 849–862, 2018, doi: 10.1109/TPAMI.2017.2695539.
17. A. Ashiquzzaman and A. K. Tushar, "Handwritten Arabic numeral recognition using deep learning neural networks," 2017 IEEE Int. Conf. Imaging, Vis. Pattern Recognition, icIVPR 2017, pp. 1–4, 2017, doi: 10.1109/ICIVPR.2017.7890866.
18. M. Aamir, N. Mohd Nawi, F. Wahid, and H. Mahdin, "A deep contractive autoencoder for solving multiclass classification problems," Evol. Intell., no. 0123456789, 2020, doi: 10.1007/s12065-020-00424-6.
19. E. S. Gati, B. D. Nimo, and E. K. Asiamah, "Kannada-Mnist Classification Using Skip CNN," 2019 16th Int. Comput. Conf. Wavelet Act. Media Technol. Inf. Process. ICCWAMTIP 2019, pp. 245–248, 2019, doi: 10.1109/ICCWAMTIP47768.2019.9067521.
20. R. Zaheer and H. Shaziya, "A Study of the Optimization Algorithms in Deep Learning," Proc. 3rd Int. Conf. Inven. Syst. Control. ICISC 2019, no. Icisc, pp. 536–539, 2019, doi: 10.1109/ICISC44355.2019.9036442.
21. D. Y. Ge, X. F. Yao, W. J. Xiang, X. J. Wen, and E. C. Liu, "Design of high accuracy detector for MNIST handwritten digit recognition based on convolutional neural network," Proc. - 2019 12th Int. Conf. Intell. Comput. Technol. Autom. ICICTA 2019, pp. 658–662, 2019, doi: 10.1109/ICICTA49267.2019.00145.
22. O. Kaziha and T. Bonny, "A Comparison of Quantized Convolutional and LSTM Recurrent Neural Network Models Using MNIST," 2019 Int. Conf. Electr. Comput. Technol. Appl. ICECTA 2019, pp. 6–10, 2019, doi: 10.1109/ICECTA48151.2019.8959793.
23. A. Garg, Di. Gupta, S. Saxena, and P. P. Sahadev, "Validation of Random Dataset Using an Efficient CNN Model Trained on MNIST Handwritten Dataset," 2019 6th Int. Conf. Signal Process. Integr. Networks, SPIN 2019, pp. 602–606, 2019, doi: 10.1109/SPIN.2019.8711703.
24. Y. Lecun, "MNIST Handwritten digits recognition dataset," [Online]. Available: http://yann.lecun.com/exdb/mnist/.