IJARIIT INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

ISSN: 2454-132X
Impact Factor: 6.078
(Volume 7, Issue 6 - V7I6-1212)
Available online at: https://www.ijariit.com

Handwritten Digit Classification using CNN

V. Serisha
serisha071999@gmail.com
Bharath University, Chennai, Tamil Nadu

Beeraka Sridevi
sridevibeeraka999@gmail.com
Bharath University, Chennai, Tamil Nadu

Shriya Singh
shriyabanno29@gmail.com
Bharath University, Chennai, Tamil Nadu

Dr. B. Karthik
karthik.ece@bharathuniv.ac.in
Bharath University, Chennai, Tamil Nadu

Abstract— Since people's writing styles vary in form and orientation, handwritten digit identification is a difficult job. Two steps are essential for the production of effective handwritten digit recognition. The extraction of discriminating features from handwritten images is the first step, and the classification of new digit images is the second. Convolutional Neural Networks (CNN) are at the heart of spectacular developments in deep learning that combine Artificial Neural Networks (ANN) and cutting-edge deep learning techniques. Pattern recognition, sentence classification, speech recognition, face recognition, text categorization, document interpretation, scene recognition, and handwritten digit recognition are only a few of the applications. Our project's aim is to improve detection accuracy by expanding the model depth. This is accomplished by increasing the number of layers and the number of filters. The Modified National Institute of Standards and Technology (MNIST) datasets were used for this performance assessment using CNN.

Keywords— *Convolutional Neural Network, handwritten digit classification, MNIST datasets, Artificial Neural Network, deep learning*

1. INTRODUCTION

Computer vision aims to give machines the ability to see and interpret information in the same way as humans do. In the field of computer vision, several algorithms for image recognition have been developed. Computer vision aims to give machines the ability to see and interpret information in the same way as humans do. In the field of computer vision, several algorithms for image recognition have been developed. In general, a seven-layered convolutional neural network is designed with one input layer, five hidden layers, and one output layer. By adding more convolutional and pooling layers with the same sized filter while increasing the number of filters, we can increase the depth of the function extractor part of the model. In our case, we'll use a double convolutional layer with 64 filters on each side, followed by a max-pooling layer.
This will help to increase the system's accuracy by 99 percent, which is an improvement over the original system's performance. The aim of this study is to look at all of the CNN architecture parameters that provide the best recognition accuracy for the MNIST dataset.

Some researchers used a collection of CNN network architectures for the same dataset to boost recognition accuracy at the expense of increased computational cost and testing complexity, but with similar accuracy as the current work. This research is unique in that it examines all of the parameters of CNN architecture that offer the best recognition accuracy for an MNIST dataset while minimizing the extra cost. This experiment clearly shows the impact of increasing the number of convolutional layers in CNN architecture on the efficiency of handwritten digit recognition.

Our main goal is to achieve comparable accuracy without using ensemble architecture by using a pure CNN architecture. Running the machine across different epochs was used to vary the accuracy. One loop through the entire training dataset is referred to as an epoch. Training a neural network typically takes many epochs.

2. LITERATURE REVIEW

CNN is significant in a variety of fields, including image processing. It has a major influence in a variety of fields. CNN is used for fault identification and classification in nanotechnologies such as semiconductor manufacturing. The identification of handwritten digits has piqued researchers' interest. This subject has resulted in a large number of papers and articles being written in recent years. In contrast to the most commonly used machine learning algorithms like SVM, KNN, and RFC, research shows that Deep Learning algorithms like multilayer CNN using Keras with Theano and Tensorflow have the highest accuracy. Convolutional Neural Network (CNN) is widely used in image recognition, video processing, and other applications due to its high accuracy. Many researchers are attempting to recognize emotion in a phrase.

CNN is being used in natural language processing and sentiment recognition by varying different parameters. It is pretty challenging to get a good performance as more parameters are needed for the large-scale neural network. Many researchers are trying to increase the accuracy with less error in CNN. In another research, they have shown that deep nets perform better when they are trained by simple back-propagation. Their architecture results in the lowest error rate on MNIST compare to NORB and CIFAR10. Researchers are working on this issue to reduce the error rate as much as possible in handwriting recognition. In one research, an error rate of 1.19% is achieved using 3-NN trained and tested on MNIST. Deep CNN can be adjustable with the input image noise. Coherence recurrent convolutional network (CRCN) is a multimodal neural architecture. It is being used in recovering sentences in an image.

Some researchers are working on new approaches to overcome the limitations of conventional convolutional layers. It has a 99 percent accuracy rate and can handle vast amounts of data. CNN's applications are the by the day, and many different types of research are being conducted. Researchers are working hard to reduce the number of errors. Error rates are measured using MNIST datasets and CIFAR. CNN is being used to cover up blurry videos. A new model was proposed using the MNIST dataset for this reason. This method has a 98 percent accuracy rate and a failure range of 0.1 percent to 8.5 percent. A CNN traffic sign recognition model has been proposed in Germany. With 99.65% accuracy, it proposed a faster result. A loss function was created that can be used with light-weighted 1D and 2D CNN. The accuracy in this case was 93 percent and 91 percent, respectively.

Multiple researches in the area of fault detection and classification (FDC) for semiconductor manufacturing processes have been performed on the prediction of manufacturing results using sensor signals. However, fault diagnosis, which is used to find answers to root causes, has remained a difficult task. Specifically, despite its high classification accuracy, process monitoring using neural networks was only used to a limited degree because it is a black-box model, making the relationships between input data and output results difficult to interpret in real manufacturing settings.

3. CONVOLUTIONAL NEURAL NETWORK MODELING FOR HANDWRITTEN DIGITS CLASSIFICATION

The cost function can be used to alter the efficiency of the proposed model. The following equation is used to express the function:

$$C(w,b) = \frac{1}{2n}\sum_{x}\left[y(x) - a^{2}\right]^{2} \tag{1}$$

Where, w denotes the total number of training inputs,
b denotes the bias, and
n denotes the total number of weights in the network.
The actual production is denoted by the letter a.

$$w^{new} = w^{old} - \frac{\eta}{m}\frac{\partial C_{xj}}{\partial w^{old}}$$

$$b^{new} = b^{old} - \frac{\eta}{m}\frac{\partial C_{xj}}{\partial w^{old}}$$

(2) & (3)

The performance of the network can be expressed as follows using the equations (2) and (3):

$$a = f(z) = f(wa + b) \tag{4}$$

The back propagation method was used to calculate the effective weight that contributes to the total error of the network. It keeps changing the neural network's weights until an accurate and efficient result is obtained.

3.1 Developing Baseline Model: This is important because it both necessitates the development of the test harness infrastructure so that any model we create can be tested on the dataset, as well as establishing a benchmark in model performance on the problem against which all changes can be measured.

The test harness is modular in nature, allowing us to create separate functions for each component. This helps one to modify or replace a specific feature of the test harness without affecting the rest.
This test harness is made up of five main components. They are the loading of the dataset, the preparation of the dataset, the model description, the model evaluation, and the results presentation.

3.2 Loading the dataset

The images are indeed pre-aligned (e.g., each image only contains a single hand-drawn digit), which means they are all grayscale and have the same square size of 2828 pixels.

As a result, we reshaped the data arrays to provide a single colour channel and loaded the files.

3.3 Preparing the Pixel Data

The pixel values for and image in the dataset are unsigned integers between 0 and 255, or black and white. The pixel values of grayscale images were then normalised, or rescaled, to the range [0,1]. The pixel values were separated by the maximum value after converting the data form from unsigned integers to floats.

3.4 Defining the Model

The feature extraction front end, which consists of convolutional and pooling layers, and the classifier backend, which will allow a prediction, are the two main aspects of this model.

We can start with a single convolutional layer with a small filter size (3, 3) and a small number of filters (32) for the convolutional front-end, followed by a max-pooling layer. The filter maps can then be flattened to give the classifier features.

To predict the probability distribution of an image belonging to each of the 10 groups, we needed an output layer with 10 nodes. A softmax activation function is also needed for this. We introduced a dense layer between the feature extractor and the output layer to interpret the features, in this case with 100 nodes, between the feature extractor and the output layer.

The ReLU activation function and the weight initialization scheme were used by all layers, which are both best practises.
For the stochastic gradient descent optimizer, we used a conservative configuration with a learning rate of 0.01 and a momentum of 0.9. The categorical cross-entropy loss function was designed for multi-class classification, and the classification accuracy metric was controlled, which was reasonable considering that each of the 10 classes had the same number of instances.

3.5 Evaluating the model

Five-fold cross-validation was used to test the model. The value of k=5 was chosen to serve as a benchmark for both repeated evaluation and to avoid requiring a long run time. Each test set consisted of 20% of the training dataset, or approximately 12,000 instances, which was similar to the size of the actual test set for this issue.

With a default batch size of 32 instances, we trained the baseline model for a modest 10 training epochs. The learning curves were assessed using the test set for each fold.

3.6 Improved model

We experimented with two aspects of model configuration that often resulted in an improvement: changing the learning algorithm and increasing the depth of the model.

- **Changing the learning algorithm**

Batch normalisation is a technique for rapidly speeding up the learning of a model and resulting in significant performance gains. The impact of batch normalisation on our built baseline model was investigated.

After the convolutional and completely linked layers, batch normalisation was used. It has the effect of altering the layer's output distribution, specifically by standardising the outputs. The learning process is stabilised and accelerated as a result of this.

For the convolutional and dense layers of our baseline model, we revised the model specification to use batch normalisation after the activation function.

- **Increasing the model depth**

We increased the depth of the function extractor portion of the model by following a VGG-like trend of raising the number of filters while adding more convolutional and pooling layers with the same sized filter. We used a double convolutional layer with 64 filters on each side, followed by another max-pooling layer in this case.

The loading of the dataset, the preparation of the dataset, the description of the model, the evaluation of the model, and the presentation of results are all key elements of all the enhanced models. Between the baseline model and the improved methods, we found a strange difference in accuracy. Although the improved model due to batch normalisation reduced the baseline model's accuracy, the depth model's accuracy increased.

The proposed model consists of various CNN layer combinations. It has been used to classify MNIST datasets. The input layer consists of a (28, 28, 1) input matrix, suggesting that the network's input data consists of 784 neurons. For dataset generation, 0 represents white pixels and 1 represents black pixels. The activation function ReLU (Rectified Linear Unit) is used to extract features from the input data after the convolution process is completed with 32 (fixed) filters. The locality of the filters is determined by the kernel size in this case. In this segment, the kernel size used is (3, 3).

The 2D Max pooling operation is performed in the following session with a kernel size of (2, 2) for sub sampling the dimension of the function map. The primary goal of using the activation function ReLU after each convolutional operation is to improve the model's output. The proposed model uses the pooling layer to reduce the output information, the number of parameters, and the computational complexity of the convolutional layers.

The convolution layer 02 and convolution layer 03 are the two major experimental layers. The total number of filters is denoted by "N," which is a variable digit. Although experimenting, the value of N has been modified for individual approaches. The kernel size of convolution layer 02 and convolutional layer 03 is the same (3,3), and they are both accompanied by a ReLU layer. Further, before entering the completely connected layer in the flattened section, the 2D Max pooling operation is used to improve efficiency.

After the pooling layer, the flatten layer transforms the 2D feature map matrix into a 1D feature vector, allowing the completely connected layer to handle the output properly. There are 100 neurons in the totally connected layer, which is followed by another ReLU layer. At the completely connected layer, the dropout regularisation approach is used to minimise model overfitting. While practising, the method switches off some neurons at random in order to improve the network's efficiency. It aids in better generalisation and reduces the amount of time spent assembling the training dataset.

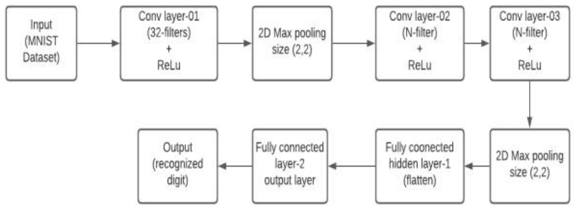

Input (MNIST Dataset)
Conv layer-01 (32-filters) + ReLu
2D Max pooling size (2,2)
Conv layer-02 (N-filter) + ReLu
Conv layer-03 (N-filter) + ReLu
2D Max pooling size (2,2)
Fully cooneсted hidden layer-1 (flatten)
Fully connected layer-2 output layer
Output (recognized digit)

Fig.1. Proposed system block diagram

4. MNIST DATASET

The MNIST database (Modified National Institute of Standards and Technology) is a large database of handwritten digits that is widely used to train image processing systems. This classic dataset of handwritten images has been used to test classification algorithms since its release in 1999. In the world of machine learning, it's still commonly used for training and research. It was made by "re-mixing" samples from the original datasets of NIST (National Institute of Standards and Technology). Since NIST's training dataset was derived from American Census Bureau employees, while the research dataset was derived from American high school students, the inventors believed it was unsuitable for machine learning experiments. The digits' black and white images were normalised and anti-aliased to fit into a 28x28 pixel bounding frame, which included grayscale levels.

The MNIST database contains 70,000 hand-written digits that have been numbered (60,000 training images and 10,000 research images). A set of training images is a collection of images for which the desired result is defined. When we enter them into a programme, it analyses their features and runs them through a classification routine to decide the best weights to apply to the features in order to achieve the desired result. A typical test image is a digital image file that is used to test image processing and image confining algorithms through multiple institutions. Different labs can visually and quantitatively compare findings by using similar standard test photos.

There are ten different grades (one for each of the 10 digits). The aim is to train a model with 60,000 training images and then test its classification accuracy against 10,000 test images. Each image has a height of 28 pixels and a width of 28 pixels, for a total of 784 pixels. Each pixel has a single pixel-value that indicates its lightness or darkness, with higher numbers indicating darker pixels. This pixel value is an integer that ranges from 0 to 255 inclusively. Since all of the images are 2828 pixels wide, they form an array that can be flattened into a vector with dimensions of 28*28=784.

Each pixel's intensity is defined by a binary value in each part of the vector. Because of the availability of reported results with various classifiers, the MNIST database was used in this analysis. The database is also widely used as a benchmark database in comparison studies of various handwritten digit recognition experiments for different regional and foreign languages.

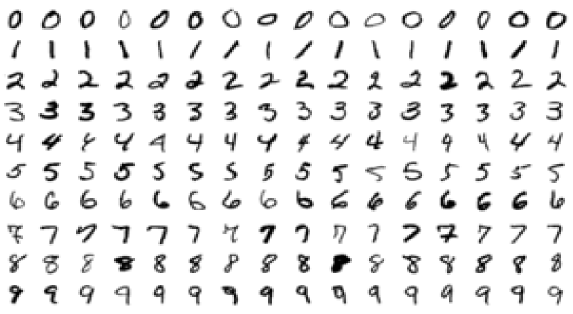
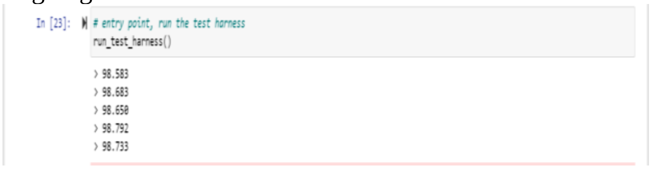


Fig.2. MNIST database sample images

5. RESULTS AND ANALYSIS

We got the graph for the classification accuracy for each fold of the cross-validation phase after running the baseline model. This was useful for getting a sense of how the model evaluation was going.



```
In [23]:  # entry point, run the test harness
           run_test_harness()

           > 98.583
           > 98.683
           > 98.650
           > 98.792
           > 98.733
```

Fig .3. Baseline model accuracies for each fold

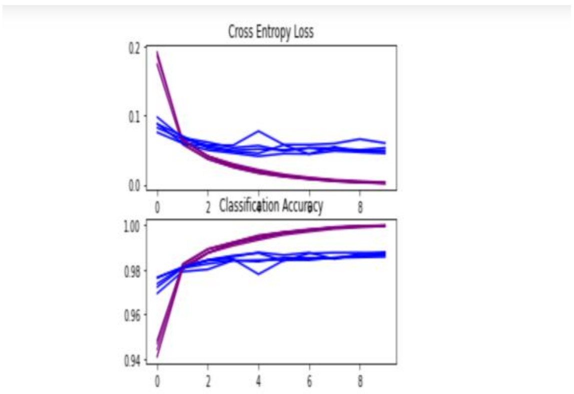


Fig.4. Loss and Accuracy Learning Curves for the Baseline Model during k-Fold Cross-Validation

The model achieved a good fit in general, with the train and test learning curves converging, as seen in the diagnostic curve above. There are no obvious signs of overfitting or underfitting. Model performance on the training dataset is indicated by purple lines, while model performance on the holdout test dataset is indicated by blue lines.

The model's approximate skill in this case is about 98.688 percent, which is fair.

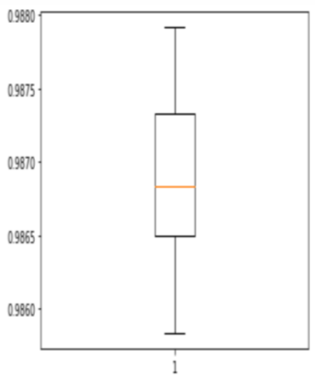
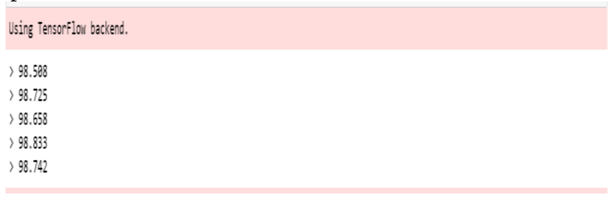


Fig.5. Box and Whisker Plot of Accuracy Scores for the Baseline Model Evaluated Using k-Fold Cross-Validation

Across the cross-validation folds, we saw a slight decrease in model output for the batch normalisation technique as compared to the baseline.



```
Using TensorFlow backend.

> 98.508
> 98.725
> 98.658
> 98.833
> 98.742
```

Fig.6. Batch normalization accuracies for each fold

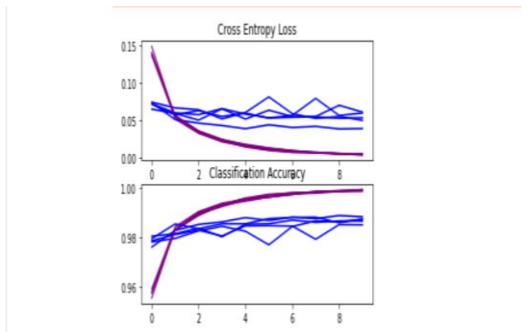
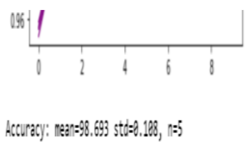


Fig.7. Loss and Accuracy Learning Curves for the Batch Normalization during k-Fold Cross-Validation

The learning curves are plotted, showing that the speed of learning (improvement over epochs) does not appear to vary from the baseline model in this case.

The graphs indicate that batch normalisation, at least as applied in this case, is not beneficial.



```
Accuracy: mean=98.693 std=0.108, n=5
```

Fig.8. Summary of batch normalization model performance

We can see that the difference between the baseline model and batch normalisation systems is very small.

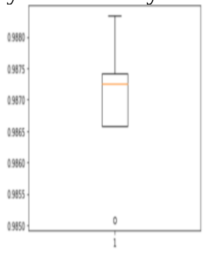
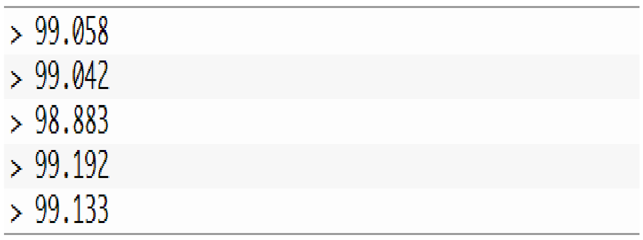


Fig.9. Box and Whisker Plot of Accuracy Scores for the Baseline Model Evaluated Using k-Fold Cross-Validation

The per-fold scores for the model depth method indicated some progress over the baseline.



```
> 99.058
> 99.042
> 98.883
> 99.192
> 99.133
```

Fig.10. Model depth accuracies for each fold

The learning curves are plotted, which in this case shows that the models are still well-fitting to the problem, with no obvious signs of overfitting. The plots may also indicate that further training epochs are beneficial.

Model performance on the training dataset is indicated by purple lines, while model performance on the holdout test dataset is indicated by blue lines.

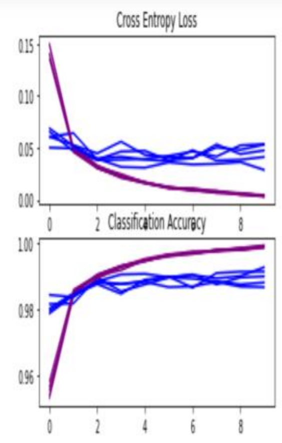
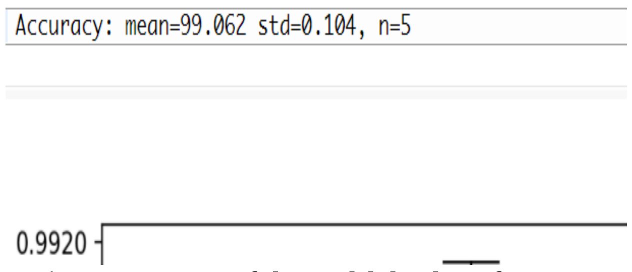


Fig.11. Loss and Accuracy Learning Curves for the model depth during k-Fold Cross-Validation

The model's estimated performance is seen, with a slight increase in performance from 98.688 percent to 99.062 percent relative to the baseline, as well as a small decrease in the standard deviation.



```
Accuracy: mean=99.062 std=0.104, n=5
```

Fig.12. Summary of the model depth performance

The batch normalisation model's standard deviation was 104. However, we saw an increase in the model depth model, with the standard deviation dropping to 104.

All three systems were subjected to a five-fold cross-validation procedure. This is due to the fact that the software takes a long time to run on the server. As a result, increasing the k fold cross validation lengthens the code's run time, making it take even longer to show the results and diagnostic curve.

Finally, to summarise the distribution of accuracy scores, a box and whisker plot is developed.

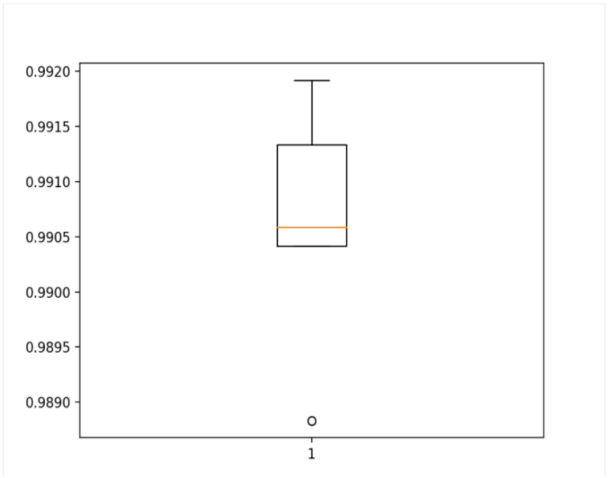


Fig.13. Accuracy Scores for the Model Depth Evaluated Using k-Fold Cross-Validation in a Box and Whisker Plot

6. CONCLUSION

The overall performance of the CNN model was analyzed in relation to the number of filters used in the convolutional layer. The study found that increasing the number of filters applied in the CNN model's different layers or increasing the number of feature extractors would improve the CNN model's output for recognizing handwritten digits. The accuracy and standard deviation were used to assess the results.

When comparing the baseline model, batch normalisation, and the model depth system, it is clear that the model depth system had the highest accuracy, with the least amount of error and over-fitting.

We tested variants of a convolutional neural network to avoid complex pre-processing, costly feature extraction, and a complex ensemble (classifier combination) method of a conventional recognition system in order to boost the efficiency of handwritten digit recognition. The current study suggests the function of various hyper-parameters based on an extensive evaluation using an MNIST dataset.

We also found that fine-tuning hyper-parameters is critical for improving CNN architecture efficiency. For the MNIST database, we obtained a recognition rate of 99 percent, which is higher than all previous reports.

Experiments show how increasing the number of convolutional layers in CNN architecture affects the efficiency of handwritten digit recognition.

7. FUTURE SCOPE

Handwritten digit recognition is a large concern for learning about neural networks and improving more advanced deep learning techniques.

The future of this system is vast. The future work consists of a deeper analytical process for the proposed CNN model to ensure better accuracy concerning the computational time, sensitivity, and specificity. By varying the number of secret layers and batch size, we can see how the overall classification accuracy varies.

Different CNN architectures, such as hybrid CNN, such as CNN-RNN and CNN-HMM models, and domain-specific recognition systems, may be studied in the future. CNN learning parameters, such as the number of layers, learning rate, and kernel sizes of convolutional filters, can be optimised using evolutionary algorithms.

It can be extended in the future to include character recognition, real-time handwriting recognition, reading computerised bank check numbers, signature authentication, and postal address interpretation, among other things.

The proposed model can be enhanced to recognise handwritten characters in a number of languages, including French, English, Hindi, Bengali, and others. To improve classification efficiency, some optimization techniques can be investigated.

REFERENCES

[1] Larochelle H., Bengio Y., Louradour J. and Lamblin P. (2009) “Exploring Strategies for Training Deep Neural Networks.” Journal of Machine Learning Research 10:1-40.
[2] Wang, L., Wang, Y. and Chang, Q. (2016) “Feature selection methods for big data bioinformatics: A survey from the search perspective.” Methods 111:21-31.
[3] Goltsev, A. and Gritsenko, V. (2012), “Investigation of efficient features for image recognition by neural networks”, Neural Networks 28:15- 23.
[4] Kang, M., Palmer-Brown and D. (2008) “A modal learning adaptive function neural network applied to handwritten digit recognition.” Information Sciences—Informatics and Computer Science, Intelligent Systems, Applications 178(20) :3802-3812.
[5] Lauer F., Suen Y. C. and Bloch G. (2007) “A trainable feature extractor for handwritten digit recognition.” Pattern Recognition 40(6):1816- 1824, 2007.
[6] Niu, X. X. and Suen, C. Y. (2012) “A novel hybrid CNN–SVM classifier for recognizing handwritten digits”, Pattern Recognition 45(4):1318– 1325.
[7] Choudhary, A., Rishi, R., and Ahlawat, S. (2012). “Unconstrained Handwritten Digit OCR Using Projection Profile and Neural Network Approach.” In: Proceedings of International Conference on Information System Design and Intelligent Applications (INDIA 2012), Springer, Heidelberg, 132, 119-126.
[8] Choudhary, A., Rishi, R. and Ahlawat, S. (2013) “A New Approach to Detect and Extract Characters from Off-Line Printed Images and Text.” In: Proceedings of International Conference on Information Technology and Quantitative Management (ITQM), Procedia Computer Science, 17, 434–440.
[9] Choudhary, A. and Rishi, R. (2011) “Improving the Character Recognition Efficiency of Feed Forward BP Neural Network.” International Journal of Computer Science & Information Technology (IJCSIT), 3(1), 85-96.
[10] Graves A. and Schmidhuber J. (2009) “Offline handwriting recognition with multidimensional recurrent neural networks.” in: Advances in neural information processing systems 545–552.
[11] Sayre (1973) “Machine recognition of handwritten words: A project report.” Pattern recognition 5 (3):213–228.
[12] Plamondon, R. and Srihari S.N. (2000) “Online and off-line handwriting recognition: a comprehensive survey.” IEEE Transactions on pattern analysis and machine intelligence 22 (1):63–84.
[13] Yuan, A., Bai, G., Jiao, L. and Liu, Y. (2012) “Offline handwritten English character recognition based on convolutional neural network” in: Document Analysis Systems (DAS), 2012 10th IAPR International Workshop on, IEEE, 125–129.
[14] Manisha, C. N., Reddy, E. S. and Krishna, Y. S. (2016) “Role of offline handwritten character recognition system in various applications” International Journal of Computer Applications 135 (2):30–33.
[15] Sa´nchez, J. A., Bosch, V., Romero, V. and K. Depuydt, J. de Does (2014) “Handwritten text recognition for historical documents in the transcriptorium project, in: Proceedings of the First International Conference on Digital Access to Textual Cultural Heritage, ACM 111–117.
[16] Pl¨otz, T. and Fink, G.A. (2009) “Markov models for offline handwriting recognition: A survey.” Int. J. Doc. Anal. Recognit. 12 (4):269–298.
[17] Choudhary A., Ahlawat, S. and Rishi R. (2014) “A Binarization Feature Extraction Approach to OCR: MLP vs. RBF.” In: Proceedings of the International Conference on Distributed Computing and Technology ICDCIT, Springer, 341-346.
[18] Choudhary, A., Rishi, R. and Ahlawat, S. (2013) “Off-Line Handwritten Character Recognition using Features Extracted from Binarization Technique.” In: Proceedings of International Conference on Intelligent Systems and Control, AASRI Procedia, 4, 306-312.
[19] Choudhary A. and Rishi R. (2014) “A Fused Feature Extraction Approach to OCR: MLP vs. RBF.” In: Proceedings of the 48th Annual Convention of Computer Society of India- Vol I. Advances in Intelligent Systems and Computing, Springer, Cham, 248, 159-166.
[20] Choudhary, A. (2014) “A Review of various Character Segmentation Techniques for Cursive Handwritten Words Recognition.” International Journal of Information & Computation Technology (IJICT) 4 (6), 559-564.
[21] Cortes, C. and Vapnik V. (1995) “Support vector networks”, Machine Learning, 20
[22] Pontil, M. and Verri, A. (1998) “Support Vector Machines for 3-D Object Recognition.” IEEE Trans. Patt. Anal. Machine Intell. 20:637-646.
[23] Osuna, E., Freund, R. and Girosi, F. (1997) “Training Support Vector Machines: An Application to Face Detection.” Proc. of CVPR, Puerto Rico. IEEE 130-136.
[24] Malon, C., Uchida, S. and Suzuki, M. (2008) “Mathematical symbol recognition with support vector machines.” Pattern Recognition Letter, 29(9):1326–32.
[25] Mahto, M.K., Bhatia, K. and Sharma, R. K. (2015) “Combined horizontal and vertical projection feature extraction technique for Gurmukhi handwritten character recognition.” 2015 International Conference on Advances in Computer Engineering and Applications. 59–65.