Global Journal of Computer Science and Technology: D
Neural & Artificial Intelligence
Volume 19 Issue 2 Version 1.0 Year 2019
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

Recognition of Handwritten Digit using Convolutional Neural Network (CNN)

By Md. Anwar Hossain & Md. Mohon Ali

Pabna University of Science & Technology

Abstract- Humans can see and visually sense the world around them by using their eyes and brains. Computer vision works on enabling computers to see and process images in the same way that human vision does. Several algorithms developed in the area of computer vision to recognize images. The goal of our work will be to create a model that will be able to identify and determine the handwritten digit from its image with better accuracy. We aim to complete this by using the concepts of Convolutional Neural Network and MNIST dataset. We will also show how MatConvNet can be used to implement our model with CPU training as well as less training time. Though the goal is to create a model which can recognize the digits, we can extend it for letters and then a person's handwriting. Through this work, we aim to learn and practically apply the concepts of Convolutional Neural Networks.

Keywords: convolutional neural network; MNIST dataset; MatConvNet; ReLu; softmax.

GJCST-D Classification: I.2.6

Strictly as per the compliance and regulations of:

Recognition of Handwritten Digit using Convolutional Neural Network (CNN)

Md. Anwar Hossain[α] & Md. Mohon Ali[σ]

Abstract- Humans can see and visually sense the world around them by using their eyes and brains. Computer vision works on enabling computers to see and process images in the same way that human vision does. Several algorithms developed in the area of computer vision to recognize images. The goal of our work will be to create a model that will be able to identify and determine the handwritten digit from its image with better accuracy. We aim to complete this by using the concepts of Convolutional Neural Network and MNIST dataset. We will also show how MatConvNet can be used to implement our model with CPU training as well as less training time. Though the goal is to create a model which can recognize the digits, we can extend it for letters and then a person's handwriting. Through this work, we aim to learn and practically apply the concepts of Convolutional Neural Networks.

Keywords: convolutional neural network; MNIST dataset; MatConvNet; ReLu; softmax.

I. Introduction

Recently Convolutional Neural Networks (CNN) becomes one of the most appealing approaches and has been an ultimate factor in a variety of recent success and challenging machine learning applications such as challenge ImageNet object detection image segmentation and face recognition. Therefore, we choose CNN for our challenging tasks of image classification. We can use it for handwriting digits recognition which is one of high academic and business transactions. There are many applications of handwriting digit recognition in our real life purposes. Precisely, we can use it in banks for reading checks, post offices for sorting letter, and many other related works.

a) MNIST database

The MNIST database (Modified National Institute of Standards and Technology database) is a handwritten digits dataset. We can use it for training various image processing systems [11]. The database is also widely used for training and testing in the field of machine learning. It has 60,000 training and 10,000 testing examples.

Each image has fixed size. The images are of size 28*28 pixels. It is a database for people who want to try learning techniques and pattern recognition methods on real-world data while spending minimal efforts on preprocessing and formatting. We will use this database in our experiment.

b) Convolutional Neural Networks

Convolutional neural networks are deep artificial neural networks. We can use it to classify images (e.g., name what they see), cluster them by similarity (photo search) and perform object recognition within scenes. It can be used to identify faces, individuals, street signs, tumors, platypuses and many other aspects of visual data. The convolutional layer is the core building block of a CNN. The layer's parameters consist of a set of learnable filters (or kernels) which have a small receptive field but extend through the full depth of the input volume. During the forward pass, each filter is convolved across the width and height of the input volume, computing the dot product, and producing a 2-dimensional activation map of that filter. As a result, the network learns when they see some specific type of feature at some spatial position in the input. Then the activation maps are fed into a down sampling layer, and like convolutions, this method is applied one patch at a time. CNN has also fully connected layer that classifies output with one label per node.

II. Related Work

Handwriting digit recognition has an active community of academics studying it. A lot of important work on convolutional neural networks happened for handwritten digit recognition [1,6,8,10]. There are many active areas of research such as Online Recognition, Offline recognition, Real-Time Handwriting Recognition, Signature Verification, Postal-Address Interpretation, Bank-Check Processing, Writer Recognition.

III. Methodology

Deep Learning has emerged as a central tool for self-perception problems like understanding images, a voice from humans, robots exploring the world. We aim to implement the concept of Convolutional Neural Network for digit recognition. Understanding CNN and applying it to the handwritten digit recognition system is the target of the proposed model. Convolutional Neural Network extracts the features maps from the 2D images.

Author α: Assistant Professor, Department of Information & Communication Engineering, Faculty of Engineering & Technology, Pabna University of Science & Technology, Pabna, Bangladesh.
e-mail: manwar.ice@gmail.com
Author σ: Student, Department of Information & Communication Engineering, Faculty of Engineering & Technology, Pabna University of Science & Technology, Pabna, Bangladesh.
e-mail: mohonali22@gmail.com

Then it can classify the images using the features maps. The convolutional neural network considers the mapping of image pixels with the neighborhood space rather than having a fully connected layer of neurons. The convolutional neural network is a powerful tool in signal and image processing. Even in the fields of computer vision such as handwriting recognition, natural object classification, and segmentation, CNN has been a much better tool compared to all other previously implemented tools. The broader aim may be to develop a machine learning model that could recognize people's handwriting.

a) The Architecture of the Proposed Model

When one starts learning deep learning with a neural network, he realizes that CNN is an important tool for image classification. Convolutional Neural Networks are a special kind of multi-layer neural networks designed to recognize visual patterns directly from pixel images with minimal preprocessing. Almost all CNN architectures follow the same general design principles of successively applying convolutional layers to the input, periodically downsampling (Max pooling) the spatial dimensions while increasing the number of feature maps. Moreover, there are also fully connected layers, activation functions and loss function (e.g., cross entropy or softmax). However, among all the operations of CNN, convolutional layers, pooling layers, and fully connected layers are the most important ones. Therefore, we will quickly introduce these layers before presenting our proposed model.

The convolutional layer is the first layer which can extract features from the images. Because pixels are only related to the adjacent and close pixels, convolution allows us to preserve the relationship between different parts of an image. Convolution is filtering the image with a smaller pixel filter to decrease the size of the image without losing the relationship between pixels. When we apply convolution to the 5x5 image by using a 3x3 filter with 1x1 stride (1-pixel shift at each step), we will end up having a 3x3 output (64% decrease in complexity).

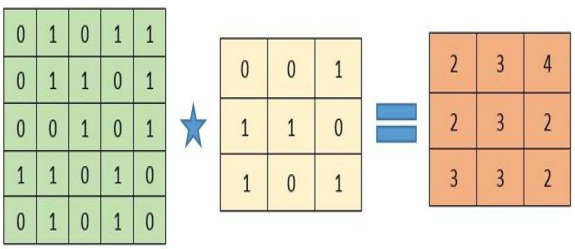

Fig. 1: Convolution operation

When constructing CNN, it is common to insert pooling layers after each convolution layer to reduce the spatial size of the features maps. Pooling layers also help with the overfitting problem. We select a pooling size to reduce the amount of the parameters by selecting the maximum, average, or sum values inside these pixels. Max Pooling, one of the most common pooling techniques, may be demonstrated as follows:

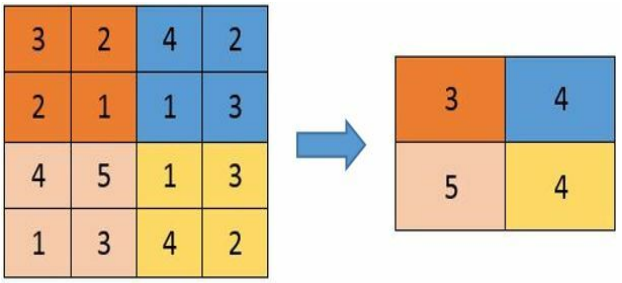

Fig. 2: Max pooling operation

A fully connected network is in any architecture where each parameter is linked to one another to determine the relation and effect of each parameter on the labels. Since convolution and pooling layers reduce time-space complexity, we can construct a fully connected network in the end to classify the images.

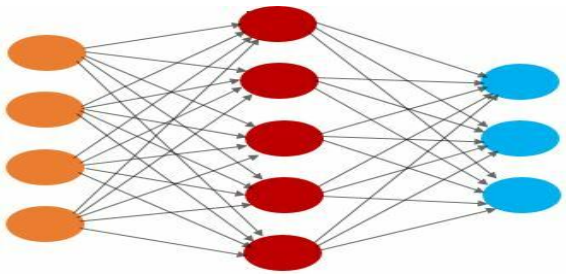

Fig. 3: Fully connected layers

Now we think, it is time to share an overview look of our proposed convolutional neural network. It has similarity with other handwritten digit recognition architectures [1,6,8,10,11] but has changed in a number of filters, neurons and activation functions for better performance. It has seven layers.

b) Explanation of the Model

A simple convolutional neural network is a sequence of layers, and every layer transforms one volume of activations to another through a differentiable function. We use three main types of layers to build the network. These are convolutional layers, pooling layers and fully connected layers. We will stack these layers to form our network architecture. We will go into more details below.

Fig.4 shows the architecture of our proposed CNN model. At first, we need some pre-processing on the images like resizing images, normalizing the pixel

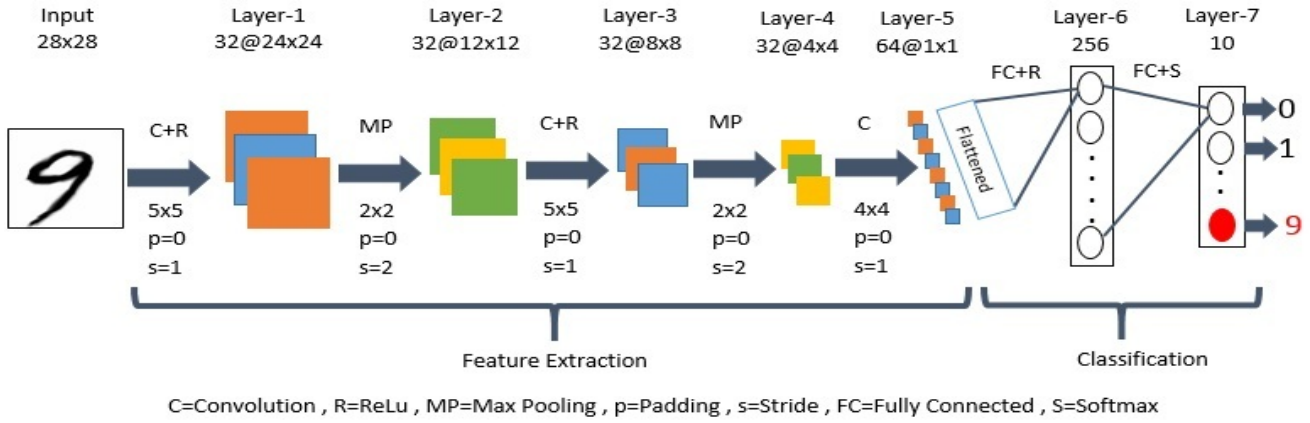

Fig. 4: The architecture of our proposed CNN

values, etc. After the necessary pre-processing, data is ready to be fed to the model.

Layer-1 consists of a convolutional layer with ReLu (Rectified Linear Unit) activation function. It is the first convolutional layer of our CNN architecture. This layer gets the pre-processed image as the input of size n*n=28*28. The convolution filter size (f*f) is 5*5; padding (p) is 0 (around all the sides of the image), stride (s) is 1 and the number of filters is 32. After this convolution operation, we get feature maps of size 32@24*24 where 32 is the number of feature maps which is equal to the number of filters used, and 24 comes from the formula ((n+2p-f)/s)+1=((28+2*0-5)/1)+1=24. Then the ReLu activation is done in each feature map.

Layer-2 is the max pooling layer. This layer gets the input of size 32@24*24 from the previous layer. The pooling size is 2*2; padding is 0 and stride is 2. After this max pooling operation, we get feature maps of size 32@12*12. Max pooling is done in each feature map independently, so we get same number feature maps as the previous layer, and 12 comes from the same formula ((n+2p-f)/s)+1. This layer has no activation function.

Layer-3 is the second convolutional layer with ReLu activation function. This layer gets the input of size 32@12*12 from the previous layer. The filter size is 5*5; padding is 0, the stride is 1 and the number of filters is 32. After this convolution operation, we get feature maps of size 32@8*8. Then ReLu activation is done in each feature map.

Layer-4 is the second max pooling layer. This layer gets the input of size 32@8*8 from the previous layer. The pooling size is 2*2; padding is 0 and stride is 2. After this max pooling operation, we get a feature map of size 32@4*4.

Layer-5 is the third convolutional layer without ReLu activation function. This layer gets the input of size 32@4*4 from the previous layer. The filter size is 4*4; padding is 0, the stride is 1 and the number of filters is 64. After this convolution operation, we get feature maps of size 64@1*1. This layer acts as a fully connected layer and produces a one-dimensional vector of size 64 by being flattened.

Layer-6 is the fully connected layer. This layer takes an input one-dimensional vector of size 64 and outputs a one-dimensional vector of size 256. It has ReLu activation function.

Layer-7 is the last layer of the network. It is also fully connected layer. This layer will compute the class scores, resulting in a vector of size 10, where each of the ten numbers corresponds to a class score, such as among the ten categories of MNIST dataset. It has softmax activation function for final outputs.

In this way, CNN transforms the original image layer by layer from the original pixel values to the final class scores. Note that some layers contain parameters and others don't. In particular, the convolution / fully connected layers perform transformations that are a function of not only the activations in the input volume but also of the parameters (the weights and biases of the neurons). On the other hand, the ReLu / pooling layers will implement a fixed function. The parameters in the convolutional / fully connected layers will be trained with stochastic gradient descent algorithm so that the class scores are consistent with the labels in training set for each image. The algorithm will prepare the trained model which will be used to classify the digits present in the test data. Thus, we can classify the digits presents in the images as: Class-0,1,2,3,4,5,6,7,8,9.

IV. Implementation

To implement our CNN architecture, we will use MatConvNet. MatConvNet is an implementation of Convolutional Neural Networks (CNN) for MATLAB [12]. It exposes the building blocks of CNN as easy-to-use MATLAB functions, providing routines for computing linear convolutions with filter banks, feature pooling and many more. In this manner, MatConvNet allows fast prototyping of new CNN architectures; at the same time, it supports efficient computation on CPU and GPU allowing to train complex models on large datasets such as Image Net ILSVRC.

Convolutional Neural Networks (CNN) are the current state-of-art architecture for the image classification task. Our proposed 2-D Convolutional Neural Network (CNN) model is designed using MatConvNet backened for the well known MNIST digit recognition task. The whole workflow can be to preparing the data, building and compiling of the model, training and evaluating the model and saving the model to disk to reuse.

Preparing the data is the first step of our approach. Before we build the network, we need to set up our training and testing data, combine data, combine labels and reshape into the appropriate size. We save the dataset of normalized data (single precision and zero mean), labels, and miscellaneous (meta) information.

Building and compiling of the model is the second step. To create the CNN, we must initialize MatConvNets DagNN (Directed acyclic graph neural network) network and then define important initialization parameters.

Batch size determines the number of samples for the training phase of the CNN. The CNN will process all the training data, but only in increments of the specified batch size. We use batch size for computational efficiency, and its value will be dependent on the user's available hardware. An epoch is a successful forward pass and a backward pass through the network. It's usually beneficial to set its value high. Then we can reduce the value once if one is satisfied with the convergence at a particular state (chosen epoch) in the network. Learning rate is a very sensitive parameter that pushes the model towards convergence. Finding its best value will be an empirical process unless one invokes more powerful techniques such as batch normalization. In our experiment, we use batch size 40, a number of epochs 8 and learning rate 0.001 for maximum accuracy.

Now we can build our CNN by creating each layer individually as shown in Fig.5. Afterward, we will invoke objective (log loss) and error (classification loss) layers that will provide a graphical visualization of the training and validation convergence after completing each epoch. After building the network layers, we initialize the weights. MatConvNet does this with Gaussian distribution.

```
% Layer-1 and Layer-2 ( Convolution + ReLu and Max Pooling )
net.addLayer('conv1', dagnn.Conv('size', [5 5 1 32], 'hasBias', true, 'stride', [1 1], 'pad', [0 0 0 0]), {'input'}, {'conv1'},  {'conv1f'  'conv1b'});
net.addLayer('relu1', dagnn.ReLU(), {'conv1'}, {'relu1'}, {});
net.addLayer('pool1', dagnn.Pooling('method', 'max', 'poolSize', [2 2], 'stride', [2 2], 'pad', [0 0 0 0]), {'relu1'}, {'pool1'}, {});

% Layer-3 and Layer-4 ( Convolution + ReLu and Max Pooling )
net.addLayer('conv2', dagnn.Conv('size', [5 5 32 32], 'hasBias', true, 'stride', [1 1], 'pad', [0 0 0 0]), {'pool1'}, {'conv2'},  {'conv2f'  'conv2b'});
net.addLayer('relu2', dagnn.ReLU(), {'conv2'}, {'relu2'}, {});
net.addLayer('pool2', dagnn.Pooling('method', 'max', 'poolSize', [2 2], 'stride', [2 2], 'pad', [0 0 0 0]), {'relu2'}, {'pool2'}, {});

% Layer-5 ( Convolution acts as Fully Connected and Flattened )
net.addLayer('conv3', dagnn.Conv('size', [4 4 32 64], 'hasBias', true, 'stride', [1 1], 'pad', [0 0 0 0]), {'pool2'}, {'conv3'},  {'conv3f'  'conv3b'});

% Layer-6 ( Fully Connected + ReLu )
net.addLayer('fc1', dagnn.Conv('size', [1 1 64 256], 'hasBias', true, 'stride', [1 1], 'pad', [0 0 0 0]), {'conv3'}, {'fc1'},  {'conv5f'  'conv5b'});
net.addLayer('relu5', dagnn.ReLU(), {'fc1'}, {'relu5'}, {});

% Layer-7 ( Fully Connected + Softmax )
net.addLayer('classifier', dagnn.Conv('size', [1 1 256 10], 'hasBias', true, 'stride', [1 1], 'pad', [0 0 0 0]), {'relu5'}, {'classifier'},  {'conv6f'  'conv6b'});
net.addLayer('prediction', dagnn.SoftMax(), {'classifier'}, {'prediction'}, {});

% Loss
net.addLayer('objective', dagnn.Loss('loss', 'log'), {'prediction', 'label'}, {'objective'}, {});
net.addLayer('error', dagnn.Loss('loss', 'classerror'), {'prediction','label'}, 'error') ;
```

Fig. 5: CNN layers in MatConvNet

The third step is the training and evaluating the model. Training a CNN requires computing the derivatives of the loss concerning the network parameters. We use back propagation algorithm for computing derivatives. It is a memory-efficient implementation of the chain rule for derivatives. We have use Stochastic Gradient Descent (SGD) training algorithm to adjust the weight of the connection between neurons so that the loss reaches a minimum value or stops after several epochs. Only log loss is used to adjust weights. We have used CPU training. It is important to note that GPU training will dramatically help training time for CNN.

Lastly, we can begin the training of CNN by supplying the training data, the constructed model and the current 'batch' of data. When training the CNN, only the data specified for training plays a role in minimizing error in the CNN. The algorithm uses training data for the forward and backward pass. The algorithm uses validation data to see how the CNN responds to new similar data, so it only is fed through the forward pass of the network. Afterward, we save the trained CNN and prepare for the testing phase.

During the training phase of the CNN, each epoch will produce up to two plots (error and objective) shown in Fig.6. For the DagNN network, MatConvNet minimizes the loss, while the error plot allows one to compute more statistical inference. The blue curve represents the training error and the orange represents the validation error of our CNN model during training.

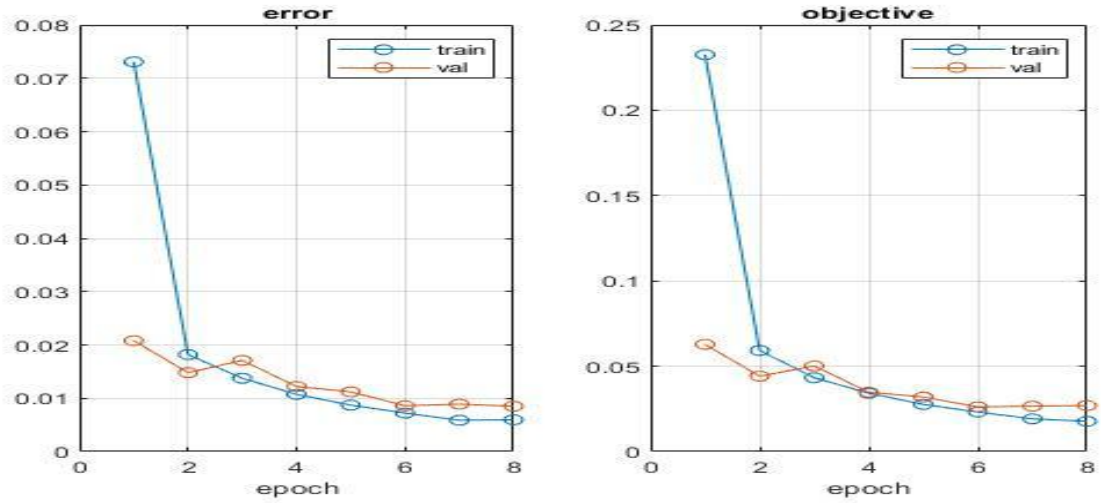

Fig. 6: Classification loss (error) and log loss (objective) during training

Finally, by using the testing data, we can evaluate our model. The following are example classification outputs from our model during testing.

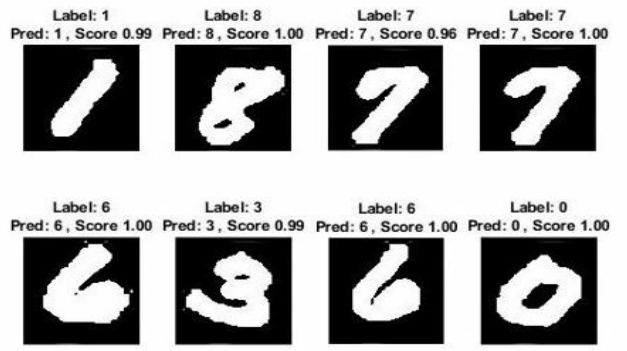

Fig. 7: Some correct recognized output

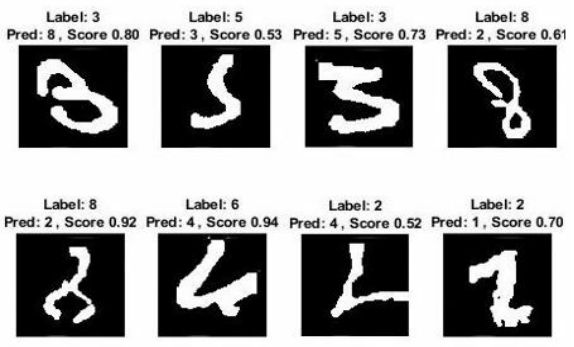

Fig. 8: Some wrong recognized output

We can find these test cases that show failed classifications. But we think most patterns can be solved by increasing the training set and increasing the standard number pattern. Moreover, missing pixels caused by image compression and image sharpness problems are also reasons for misclassification.

The fourth and final step is to save the model in the disk for reuse. We can store the trained model in a Matlab file format. Hence the saved model can be reused in later or easily ported to other environments too.

V. Results and Discussion

Among 10,000 test cases, our model misclassifies total 85 digits after eight epochs which correspond to 99.15% recognition rate shown in Table 1. The results are pretty good for such a simple model with CPU training and less training time (about 30 minutes).

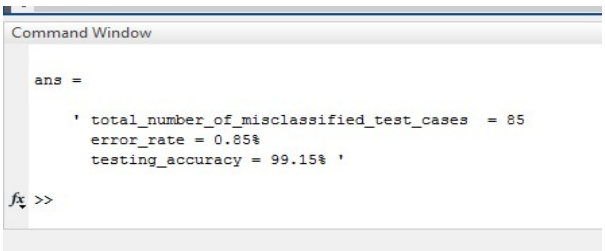

Fig. 9: Error rate and accuracy of our model

Although there are some digits which are not a good handwriting, our model will be able to classify them correctly. For example, our model classifies the following image as '2'.

Fig.10: Correct recognition of bad handwriting

Table 1: Summary of the experiment

Batch size	No. of epochs	Accuracy
80	15	98.80%
60	8	99.14%
40	8	99.15%

Testing accuracy 99.15% implies that the model is trained well for prediction. Training set size affects the accuracy and accuracy increases as the number of data increases. The more data in the training set, the smaller the impact of training error and test error, and ultimately the accuracy can be improved.

VI. Conclusion and Future Work

Here we demonstrate a model which can recognize handwritten digit. Later it can be extended for character recognition and real-time person's handwriting. Handwritten digit recognition is the first step to the vast field of Artificial Intelligence and Computer Vision. As seen from the results of the experiment, CNN proves to be far better than other classifiers. The results can be made more accurate with more convolution layers and more number of hidden neurons. It can completely abolish the need for typing. Digit recognition is an excellent prototype problem for learning about neural networks and it gives a great way to develop more advanced techniques of deep learning. In future, we are planning to develop a real-time handwritten digit recognition system.

References Références Referencias

1. Nimisha Jain, Kumar Rahul, Ipshita Khamaru. AnishKumar Jha, Anupam Ghosh (2017). "Hand Written Digit Recognition using Convolutional Neural Network (CNN)", International Journal of Innovations & Advancement in Computer Science, IJIACS,ISSN 2347 – 8616,Volume 6, Issue 5.
2. Dr.Kusumgupta2 ,"A Comprehensive Review On Handwritten Digit Recognition Using Various Neural Network Approaches", International Journal Of Enhanced Research In Management & Computer Applications, Vol. 5,No. 5, Pp. 22-25, 2016.
3. Saeed AL-Mansoori, "Intelligent Handwritten Digit Recognition using Artificial Neural Network", Int. Journal of Engineering Research and Applications, vol. 5, no. 5, pp. 46-51, 2015.
4. Nurul Ilmi, Tjokorda Agung Budi W and Kurniawan Nur R, "Handwriting Digit Recognation using Local Binary Pattern Varience and K-Nearest Neighbor," 2016 Fourth International Conference on Information and Communication Technologies (ICoICT).
5. Tobias Kutzner, Mario Dietze, Ingrid Bönninger, Carlos M. Travieso, Malay Kishore Dutta, and Anushikha Singh, "Online Handwriting Verification with Safe Password and Incresing Number of Features,"2016 3rd International Conference on Signal Processing and Integrated Networks (SPIN).
6. Haider A. Alwzwazy1, Hayder M. Albehadili2, Younes S. Alwan3, Naz E. Islam4, "Handwritten Digit Recognition using Convolutional Neural Networks", International Journal of Innovative Research in Computer and Communication Engineering, vol. 4, no. 2, pp. 1101-1106, 2016.
7. Kaiming, He and Xiangyu, Zhang and Shaoqing, Ren and Jian Sun " Spatial pyramid pooling in deep convolutional networks for visual recognitionll European", Conference on Computer Vision, arXiv:1406.4729v4 [cs.CV] 23 Apr 2015.
8. Kussul, Ernst; Tatiana Baidyk (2004). "Improved method of handwritten digit recognition tested on MNIST database". Image and Vision Computing, 22(12): 971–981.
9. "Handwritten Digit Recognition using various Neural Network Approaches", International Journal of Advanced Research in Computer and Communication Engineering, vol. 4, no. 2, pp. 78-80, 2015.
10. Huimin Wu. CNN-Based Recognition of Handwritten Digits in MNIST Database.
11. LeCun, Yann; Léon Bottou; Yoshua Bengio; Patrick Haffner (1998). "Gradient-Based Learning Applied to Document Recognition". Proceedings of the IEEE, 86(11): 2278–2324.
12. "MatConvNet Convolutional Neural Networks for MATLAB" Andrea Vedaldi, Karel Lenc, Ankush Gupta.
13. derektanderson.com/FuzzyLibrary/customnetmnst.html.
14. Ciresan, Dan Claudiu; Ueli Meier; Luca Maria Gambardella; Jürgen Schmidhuber (2011). Convolutional neural network committees for handwritten character classification 2011 International Conference on Document Analysis and Recognition (ICDAR). pp.1135–1139.
15. Youssouf Chherawala, Partha Pratim Roy and Mohamed Cheriet, "Feature Set Evaluation for Offline Handwriting Recognition Systems: Application to the Recurrent Neural Network," IEEE TRANSACTIONS ON CYBERNETICS, VOL. 46, NO. 12, DECEMBER 2016
16. Ishani Patel, ViragJagtap and Ompriya Kale. "A Survey on Feature Extraction Methods for Handwritten Digits Recognition", International Journal of Computer Applications, vol. 107, no. 12, pp. 11-17, 2014
17. Faisal Tehseen Shah, Kamran Yousaf, "Handwritten Digit Recognition Using Image Processing and Neural Networks", Proceedings of the World Congress on Engineering, vol., 2007.
18. Viragkumar N. Jagtap, Shailendra K. Mishra, "Fast Efficient Artificial Neural Network for Handwritten Digit Recognition", International Journal of Computer

Science and Information Technologies, vol. 52, no. 0975-9646, pp. 2302-2306, 2014.
19. Gil Levi and Tal Hassner, "OFFLINE HANDWRITTEN DIGIT RECOGNITION USING NEURAL NETWORK", International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, vol. 2, no. 9, pp. 4373 -4377, 2013.
20. XuanYang, Jing Pu , "MDig: Multi-digit Recognition using Convolutional Neural Network on Mobile", Stanford University.
21. Yuhao Zhang, "Deep Convolutional Network for Handwritten Chinese Character Recognition", Stanford University.
22. Jung Kim, Xiaohui Xie, "Handwritten Hangul recognition using deep convolutional neural networks", University of California.
23. Nada Basit, Yutong Zhang, Hao Wu , Haoran Liu, Jieming Bin, Yijun He, Abdeltawab M. Hendawi, "MapReduce-based Deep Learning with Handwritten Digit Recognition Case Study", University of Virginia, VA, USA.
24. Fabien Lauer, Ching Y. Suen, and G´erard Bloch "A trainable feature extractor for handwritten digit recognition", Journal Pattern Recognition, Elsevier, 40 (6), pp.1816-1824, 2007.
25. Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey. "ImageNet classification with deep convolutional neural networks". In Advances in Neural Information Processing Systems 25 (NIPS'2012). 2012.
26. Liu, C.L. et al.,." Handwritten digit recognition: Benchmarking of state-of-the-art techniques". Pattern Recognition 36, 2271–2285. 2003
27. M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional neural networks," arXiv: 1311.2901, 2013.