International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 03 | March - 2025 SJIF Rating: 8.586 ISSN: 2582-3930

Digit Recognition Using Convolutional Neural Network

Ms. B. Veena
Deptartment of ECE
Institute of Aeronautical Engineering
Hyderabad, India
b.veena@iare.ac.in

Saketh Peetha
Deptartment of ECE
Institute of Aeronautical Engineering
Hyderabad, India
sakethpeetha@gmail.com

Kesari Likitha
Department of ECE
Institute of Aeronautical Engineering
Hyderabad, India
kesarilikithamutyaveni@gmail.com

Karanji abhinav
Deptartment of ECE
Institute of Aeronautical Engineering
Hyderabad, India
21951A0403@iare.ac.in

***Abstract*—Digit detection using convolutional neural networks works as an exciting area of computer vision and machine learning. Developed to process and analyze visual data, they comprise specialized kinds of neural networks that have a number of adaptive layers. CNNs thus manage to automatically learn and adapt to detecting edges, textures, and other shapes in different images. For the case of digit detection, training could involve characterization and further classification of handwritten or printed digits from images. It will start with a data collection and pre-processing task for a large dataset of digit images; for instance, taking the very famous MNIST dataset. The model goes through a training phase where it learns to identify patterns and other different features that are unique in each digital number via many layers of convolutional and pooling operations. Using Python, the text file was read and the digit recognised using CNN. The CNN model was trained on the MNIST dataset containing 50000 images of handwritten digits for an accuracy rate of 99.4% in the prediction of an unseen digit. Using Python, the predicted digit was displayed. Since it makes use of the CNN for model training, the proposed algorithm will be fast in processing. The existing works in this field make use of an image classification tree to recognize digits.**

***Index Terms*—Convolution Neural Network, digit detection, Image Processing, MNIST dataset.**

I. Introduction

Digit recognition is a fundamental problem in the field of computer vision and pattern recognition, which means to recognize numerical digits from images automatically [3]. This technology finds very important applications in real life, like automated check processing and reading of postal addresses, digitization of handwritten notes.

On the backbone of today's digit identification is the Convolutional Neural Network, a deep learning model designed primarily for the processing and analysis of visual data. CNNs are normally composed of multiple layers: convolutional, pooling, and fully connected, amongst which students collaborate to learn hierarchical representations of the input images automatically [1]. This will enable the CNN to learn effectively different shapes, curves, and patterns characterizing different digits, even in noisy and distorted instances.

Many blocks of recent popularity of Deep Learning Networks have been triggered by the fact that they act effectively to classify information in an image and detect an image's information with regard to certain object classes. On the other side, DLNs have been executed through CNNs in several computer vision tasks like: object tracking, pose estimation, action, recognition and Object Counting [6]. Examples range from yield predictions of fruits in vineyards to disease diagnosis, such as Parkinson's, using image data alone as in Heinrich et al. The increase in these applications of DLN results from enhanced computing powers and the need for faster training and inference by extensive RAM and GPU usage, respectively [9].

Beyond computer vision, CNNs have also been applied in speech recognition and natural language processing, where they outperformed the preceding algorithms mainly based on Hidden Markov models and Gaussian mixture models. Requirements for the execution of CNN architecture differ a lot depending on the domain of application. Generally, it aims at reducing inference times, while improving the times for prediction accuracy [8].

Currently, there is no standard guideline on how to effectively design CNN architectures for the solution of domain specific problems, many of which return apparently random input-output outcomes, like manufacturing fault detection through image classification. To some extent, this challenge arises due to inadequate insight into the nature and performance implications of CNN architectures [4]. Moreover, in many cases, DLNs are considered a black-box model that would make the result hard to predict, thus making the benchmark results and evaluations very necessary. We focus on giving a state-of-the-art re-view of the evolution of

architectures of DLN for image classification in this context, which delivers best-practice guidelines [9].

Since it is the most prevalent type of network used for this task, we particularly focus on CNNs. Our goals are to provide several evaluation results and metrics that outline the dynamics of characteristics of CNN architectures, their prediction performance, and computational requirements [13]. We also compare the different views among them over time to underline the technological trends in the design of the CNNs. We use this knowledge to formulate five guidelines to enhance the methodical selection of CNN designs and architectures [10].

Overall, problems of digit recognition constitute a window to the solution of more complex problems in handwritten pattern recognition and remain one of the vivid research areas that continues to drive improvement at both academia and industry.

II. Literature

A. Convolutional Neural Network

CNN stands for Convolutional Neural Network, a Deep Learning model designed particularly to work with data having grid topology, such as images [5]. Inherent strengths of CNNs in recognizing patterns and structures within input data make them very effective when applied for tasks like image classification, object detection, and segmentation.

These layers convolve the input with filters to detect a set of local patterns, such as edges, textures,and shapes. In doing so, these layers reduce the spatial dimensions of the data and down sample it, reducing computational complexity and improving robustness to spatial variations. These layers are identical to traditional neural network layers, wherein every neuron is connected to all neurons in the previous layer [2]. They are normally used at the end of a network to make final predictions. Other activation functions like ReLU introduce non-linearity into the model to have it pick more complex patterns.

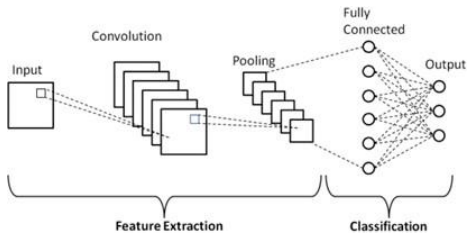

Fig. 1. Overview of CNN Architecture.

Images are high-dimensional data. Even a simple 256 x 256 RGB image has 196,608 values or pixels. Traditional neural networks struggle with handling input sizes this large. CNNs mitigate this problem by using convolutions which are local to receptive fields, greatly reducing the number of parameters and computational load [14]. CNNs can automatically learn to detect hierarchical features in images. Early layers capture simple patterns, including edges and textures, while deeper layers enable the extraction of more complex structures, including shapes and objects [15]. The hierarchical learning that occurs in this process plays a key role in image recognition, where it helps to understand different levels of abstractions.

By the very nature of convolution and pooling operations, a priori translational invariance relative to the input image is guaranteed in CNNs [4]. This means they are able to detect an object regardless of where it may be in the image, which makes them useful during object detection and image classification. Images have a spatial structure, where nearby pixels will tend to be more related than distant ones. CNNs exploit this property through local connections and shared weights, allowing them to effectively capture and process spatial hierarchies within images [7].

Image processing requires the extraction of meaningful features relating to edges, textures, and shapes. Convolutional layers of CNNs extract these features automatically during training; there is no need to engineer features manually [14]. Because CNNs are committed to weight sharing and local connectivity, they have fewer parameters compared to fully connected networks, and their computational power is reduced. This makes them more efficient for large images, reducing the possibility of over fitting.

B. Deep Neural Network

deep neural network is a type of Artificial Neural Network with multiple layers between input and output layers. Multiple layers in DNNs empower them to model complex patterns in data incorporating the learning of hierarchical representations [15].

Every layer in a DNN is composed of neurons, which are the basic processing units. The neurons in one layer are connected to the neurons in the next through weighted connections; these weights start the training and will aim to produce predictionswith minimal error.

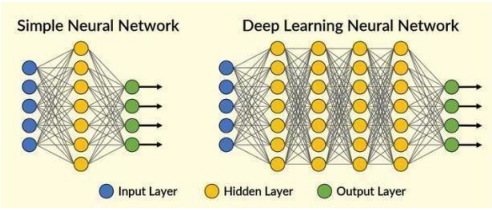

Fig. 2. Neural Network

Activation functions induce non-linearity into the network and thus allow it to learn more complex patterns. Common activation functions are the ReLU—rectified linear unit,the sigmoid and tanh [2]. The traditional training of DNNs involves backpropagation—a means of propagating an error backwards through the network as a means of updating the weights. It is normally combined with optimization algorithms such as stochastic gradient descent [14]. DNNs are very effective in applications like speech recognition, natural language process-

ing, and image recognition due to their capability of learning intricate patterns and representations in data [5].

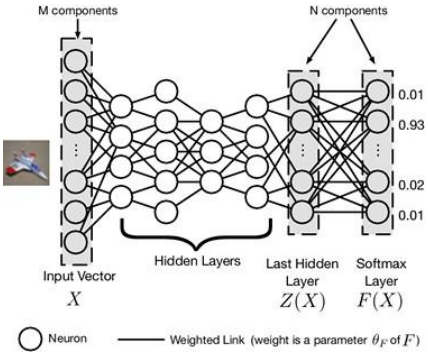

Fig. 3. Overview of DNN Architecture.

Training of DNNs, however, is compute-intensive, calls for large amounts of labeled data, and is model naive. Techniques such as transfer learning have shown great potential in reducing these challenges, whereby a model trained on some other task is fine-tuned for a new task [10]. Notably, also successfully applied in various fields including health care, for example, to support diagnosis from data, autonomous systems, where they enable tasks such as self-driving cars, are other endpoints of application. To avoid the over fitting of DNNs and improve their generalization ability on new data, most of the cases will have them with regularization techniques. This includes dropout and batch normalization [14].

C. MNIST Dataset

The MNIST database (Modified National Institute of Standards and Technology database) is a large database of handwritten digits that is commonly used for training various image processing systems. The database is also widely used for training and testing in the field of machine learning. It was created by ”re-mixing” the samples from MNIST’s original datasets. The creators felt that since MNIST’s training dataset was taken from American Census Bureau employees, while the testing dataset was taken from American high school students, it was not well-suited for machine learning experiments. Furthermore, the black and white images from MNIST were normalized to fit into a 28x28 pixel bounding box and anti-aliased, which introduced grayscale levels.

The MNIST database contains 60,000 training images and 10,000 testing images. Half of the training set and half of the test were taken from MNIST’s training dataset, while the other half of the training set and the other half of the test were taken from MNIST’s testing dataset. The original creators of the database keep a list of some of the methods tested on it.

III. Methodology And Technologies Used

As we know there is no readily available digit recognition model for recognizing digits, we made our digit recognition model by utilizing CNN. The methodology procedure is shown in diagram below.

Fig. 4. Sample images from MNIST test dataset

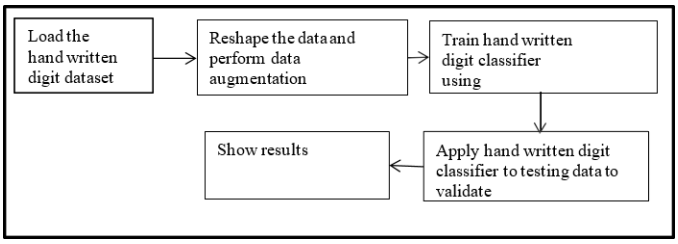

Fig. 5. Block diagram

From the block diagram above, it is clear that the first step involved is to load the training and testing dataset. After that, we build the model, which is the hand written digit classifier, where training dataset is used to teach the hand written digit classifier to recognize sign language. later, we will check results and will validate model using testing dataset. This is the overview of the methodology we did, which will be further explained in detailed as we proceed.

A. Activation Functions

1) ReLU: ReLU is a kind of piece wise linear function that returns ‘0’ if the input is negative and returns ‘1’ if it is positive. It became standard activation function to various NN’s because it’s easier for training and gives better results. To increase the image non-linearity, the rectifier function is applied. Because images are generally non-linear, This is something

we want to accomplish. The rectifier breaks up any linearity imposed on an image during the convolution operation even further to compensate for it. In an image like this, the rectifier function removes all dark portions, and those with positive value remain grey and white colors. Once we re-adjust image, one can see colors to change abruptly. Progressive transformation has ended. That means the linearity is gone.

2) Soft max Activation: It is a multi-dimensional variant of logistic function. It’s often employed as the last layer.

To work out relative likelihood, the Softmax initiation work is used. That is, the qualities of Z21, Z22, and Z23 define the ultimate probability value. Like the sigmoid activation function, the SoftMax function returns the likelihood of each class. The condition for the SoftMax actuation work is as per the following.

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

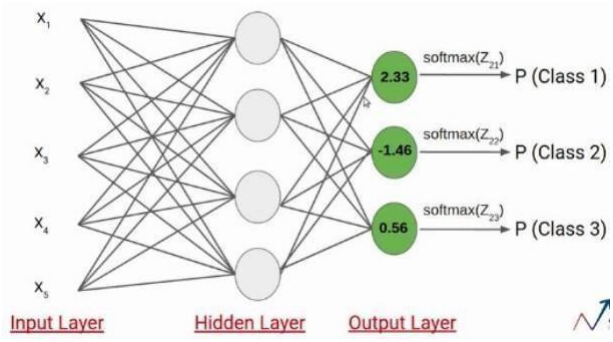

Fig. 6. Softmax classification

The Z represents output layer neurons values. Then the values will be normalized , later get converted to probabilities, which is done by dividing them via sum of exponential values. In other words, sigmoid is a Softmax variation. Let's analyze an example of how the softmax works. The neural network below is what we have.

B. Tensor flow

Rather than DL, TensorFlow was designed with large numerical computations in mind. TensorFlow showed to be advantageous to DL development, Hence Google made it open- source[6]. It is based on graphs that depict data flows and have nodes and edges. TensorFlow code may be spread over a cluster of computers using GPUs much more easily since execution mechanism is in mode of graphs.

C. Keras

Keras is the high level DL API, used during building NN's. Keras built utilizing Python, which is used to build NN implementation easier. It permits computation of various NN's in backend[6]. It is simple to comprehend. Because of this, Keras is slower compared to other DL frameworks, however it is also user-friendly. It permits one for swapping back- forth among various backends.

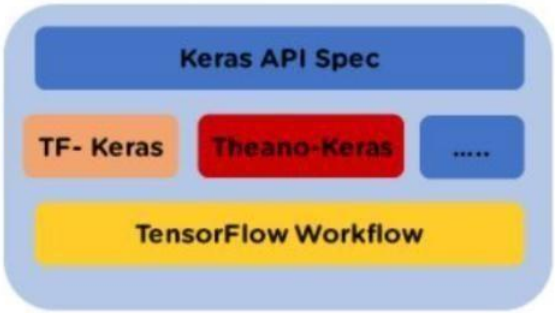

Fig. 7. Keras Architecture

TensorFlow has chose Keras as official high-level API, out of above frameworks. Keras is a type of TensorFlow module, which may be utilized to fastly run deep learning. At the same time, one can utilize Tensor flow Core API for creating custom computations including computation graphs, tensors, sessions, etc, allowing one all control on the application, permitting you to implement yours views in least time.

D. Numpy

NumPy is one of the Python library which supports massive, matrices, multi- dimensional arrays, including different high level algebraic functions, for manipulation of them. It is a python library for computing. It's an open-source module. NumPy is collaborative open-source project with a large number of collaborators. It's a tool for working with arrays and matrices in general.

Python is slower as compared to Fortran and other languages to perform looping. NumPy, which turns repetitive code into compiled form, is used to solve this problem. If you want to work on data analysis or machine learning projects, you'll need to know numpy. Because additional data analysis packages (like as pandas) are built on top of numpy, and because the scikit-learn package.

E. Pandas

Pandas is the data cleaning and a type of analysis ML tool. It provides features for data exploration, cleansing, transformation, and visualization. It gives swift, expressive, flexible data structures.

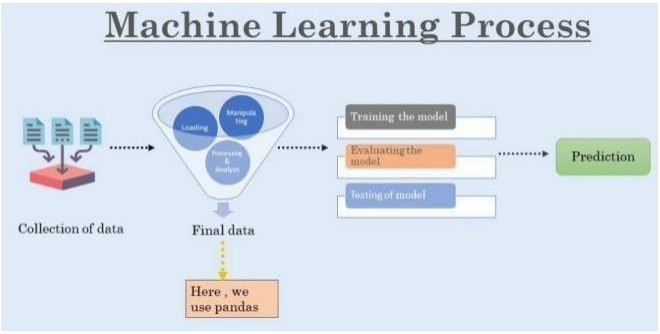

Fig. 8. Machine learning method steps

F. Matplotlib

Matplotlib, a Python package for data visualization, operates at a low level and offers a straightforward approach reminiscent of MATLAB's graphical representations. Built on NumPy arrays, this library boasts a wide array of plots, including line charts, bar charts, and histograms. While it provides considerable flexibility, leveraging its capabilities often requires writing additional code. Matplotlib is a 2003 open-source Python charting package. It's a large library with several charting functions that may be used in Python. It includes a variety of graphs like Line graph, Bar graph, Scatter graph, Histogram, so on, which permit us for visualizing different data.

G. Pyplot

Pyplot is a module of Matplotlib. Matplotlib made with intention of easy to use like MATLAB, however has its advantages of being open-source and free. It creates the plotting region in figure, creates plots with specific lines in plotting area, and then decorates plots with labels [9]. Line graph, Histogram graph, Scatter graph are just a few of the plotsavailable in Pyplot.

H. Seaborn

Seaborn is data visualisation programme in python on the basis of Matplotlib, which includes a high level user-interface that allows you to create visually stunning and practical statistics graphs. This article will show you how to use the Seaborn library to generate simple plots.

IV. Results and Discussions

Following that, the accuracy and loss curves will be shown. We are optimistic about our model's accuracy levels and accuracy/loss curves. The training accuracy and validation accuracy and their respective loss curves are shown below.

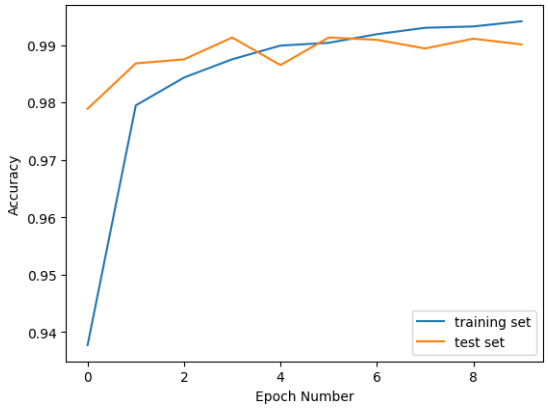

Fig. 9. Training and Validation accuracy

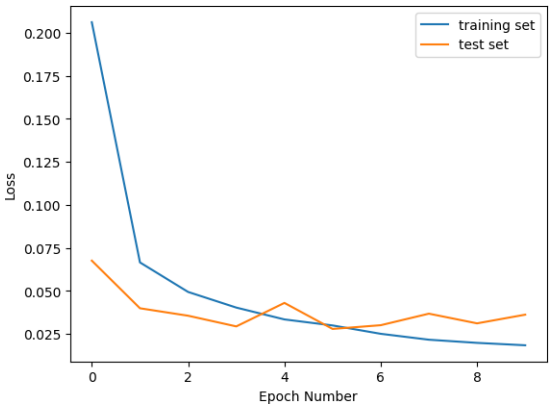

Fig. 10. Training and validation loss

Finally, we validated the model by checking a few outputs. Here, we uploaded a testing dataset. We can either upload a picture or upload a completely new data sector using open cv we can directly interact with the system, by using system local camera and get it done with the evaluation by checking predicted and actual results. So, what we did is we uploaded a dataset and wrote the piece of code to verify the output. That is, to see if the expected and actual outputs match. Fortunately, our model got enough training and from the above graphs earlier we can see how accurate our system is. Below is the sample output.

V. Conclusion

Compared with the other works, we were able to obtain a high accuracy over 99.4%. From earlier graphs (Figure 9 and 10), we can see we obtained an training accuracy of 99.4% in 10 epochs model training case and an valid accuracy of 99.4%. This concludes that the model is training and rectifying its errors as the epochs getting increased and at some point only little errors are incurred. Which means, the more training we provide to the model, and the wider the dataset, the lesser the errors and greater the accuracy. And since we are using convolution layer before the neural network, it is easier to classify the data. The trained CNN model can be improved to transmit sentences using a continuous stream of input of sign language.

VI. Future Scope

We can develop a system which can process continuous input flow of digits and convert them to their corresponding number. The proposed digit recognition instead of displaying digits, the system might be improved to recognize gestures and facial expressions in addition to letters, and instead of displaying letter labels, sentences could be displayed as a more accurate translation of language. This improves read-ability as well. It can be expanded to include translation of digits to speech. That is, The trained CNN model can be further developed to translate the written text output to voice translation. This model might be turned into a mobile app. The programme recognizes footage of speaker using neural networks and computer vision, and then utilizes sophisticated algorithms to convert it to speech. Affordably priced and always available services for any community

References

[1] Vishwanatha V, Ramachandra A C, Satya Dev Nalluri, sai Manoj Thota, & Aishwarya Thota (2023). "Hand Written Digit Recognition Using CNN"
[2] A. Krizhevsky , I. Sutskever, & G. E. Hinton (2012). ImageNet Classi-fication with Deep Convolutional Neural Networks. NIPS.
[3] D. C. Ciresan, U. Meier, L. M. Gambardella, & J. Schmidhuber (2010). Deep Big Simple Neural Nets Excel on Handwritten Digit Recognition. arXiv preprint arXiv:1003.0358.
[4] P. Y. Simard, D. Steinkraus, & J. C. Platt (2003). Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis. ICDAR.
[5] L. Wan, M. Zeiler, S. Zhang, Y. LeCun, & R. Fergus (2013). Regular-ization of Neural Networks using DropConnect. ICML.
[6] 6) M. D. Zeiler, & R. Fergus (2014). Visualizing and Understanding Convolutional Networks. ECCV.
[7] 7) I. J. Goodfellow, Y. Bulatov, J. Ibarz, S. Arnoud, & V. Shet (2013). Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks. arXiv preprint arXiv:1312.6082.

[8] H. Larochelle, D. Erhan, A. Courville, J. Bergstra, and Y. Ben-gio, 2007. “An Empirical Evaluation of Deep Architectures on Problems with Many Factors of Variation,” in Proceedings of the 24th International Conference on Machine Learning, ACM, pp. 473–480.
[9] J. Cheng, P. Wang, G. Li, Q. Hu, and H. Lu, 2018. “Recent Advances in Efficient Computation of Deep Convolutional Neural Networks,” Frontiers of Information Technology and Electronic Engineering (19:1), pp. 64–77.
[10] Y. LeCun, 1989. “Generalization and Network Design Strategies in Connectionism in Perspective”, R. Pfeifer, Z. Schreter, F. Fogelman, and L. Steels (eds.), Elsevier.
[11] D. Mishkin, N. Sergievskiy, and J. Matas, 2017. “Systematic Evaluation of CNN Advances on the ImageNet,” Computer Vision and Image Understanding (161),pp.11–19.
[12] S. Ji, W. Xu, M. Yang, K. Yu: 3D convolutional neural net-works for human action recognition. IEEE Trans. Pattern Anal. Mach. Intell. 35(1), 221–231 (2012)
[13] X. Cao, 2015. “A Practical Theory for Designing Very Deep Convolutional Neural Networks,” Technical Report.
[14] K. He, X. Zhang, S. Ren, S. Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp.770–778).
[15] I. Goodfellow, Y. Bengio, A. Courville, (2016). Deep Learning. MIT Press.
[16] K. Simonyan,A. Zisserman, (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv preprint arXiv:1409.1556.