IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 11, November 2025

DOI: 10.17148/IJIREEICE.2025.131133

Recognition of Handwritten Digit using Convolutional Neural Network in Python

Chandhan Sayee.R[1], Mohul Raj.T[2], Merlin.R. S[3], Sanjith Kumar.K[4], Nishanth.G. S[5], Monesh.V[6], M. Ulagammai[7]

Department of CSE (E-Tech), SRMIST Vadapalani, Chennai, India[1-7]

Abstract: In recent years, deep learning has become a very important part of artificial intelligence. With the growth of Artificial Neural Networks (ANN), deep learning has made machines much smarter and more capable. It is used in many areas such as health care, security, sports, robotics, and drones because it can solve many real-life problems. Among deep learning methods, the Convolutional Neural Network (CNN) is one of the most successful. It combines the ideas of ANN with modern deep learning methods. CNNs are widely used for pattern recognition, speech and face recognition, text and document classification, scene detection, and handwritten digit recognition.

Keywords: Handwritten Digit Recognition, Convolutional Neural Network,MNIST, Batch Normalization, Dropout, Data Augmentation, Learning Rate Scheduling, Deep Learning.

I. INTRODUCTION

Deep learning is now used in many areas and keeps growing every year. It is an important part of artificial intelligence. One of the best models in deep learning is the Convolutional Neural Network, or CNN. CNNs are mostly used for image-related tasks like object detection, face recognition, speech and text recognition, and handwritten digit recognition. In many of these fields, CNNs give results that are almost as accurate as humans.The idea of CNNs came from how the human eye and brain work together. In 1962, a scientist named D.H. Hubel found that some cells in a cat's brain only react to small parts of what it sees. These small parts are called receptive fields. Based on this idea, Fukushima made the first computer vision model called the Neocognitron in 1980. Later, in 1998, LeCun and his team built a CNN that could read handwritten digits directly from image pixels. They used gradient descent and backpropagation methods to train it.A simple Artificial Neural Network (ANN) has an input layer, hidden layers, and an output layer. CNNs are similar, but not all the neurons are connected to each otherEach neuron is responsible for only a small portion of the image, which makes recognizing patterns easier. In the training phase, the model examines its output against the correct answer and adjusts the weights in order to minimize error. For this work, we evaluated how varying the number of hidden layers would impact a convolutional neural network's performance on handwritten digit recognition using the MNIST dataset.

The model was built in TensorFlow and learned using stochastic gradient descent and a process called backpropagation. The construct follows three key ingredients:

1. Smart Architecture: The model architecture is deep and resembles a VGG-style architecture using layers that consist of convolution–batch normalization–relu. Batch normalization is a technique used to stabilize learning while dropout is another technique used to reduce overfitting.
2. Better Generalization: Implementing data augmentation on-the-fly during training provide variety to the training data set and architectures by applying styles consistent with that of handwriting and translation.
3. Thoughtful Training: A dynamic learning rate schedule is used initially implementing a Linear Warmup to ensure smooth convergence while applying Cosine Annealing for low temperature precision modeling.

These components represent a methodology to train models that is equal parts efficient and effective. It is remarkable that in just thirty epochs, the model is still above 99.7% accuracy, which includes very little overfitting.

II. LITERATURE REVIEW

Convolutional Neural Networks are now a key part of how computers understand images. They're great at picking up patterns and learning features from pictures, which makes them useful in many areas like object detection, medical scans, video monitoring, and even language tasks.

The researchers used basic machine learning functions such as SVM, Decision Tree, and kNN. They were good for small datasets but failed to larger datasets of images. Subsequently, researchers used basic Artificial Neural Networks; however, they required superficial selection of features and didn't have any cognitive understanding of the underlying deeper

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 11, November 2025

DOI: 10.17148/IJIREEICE.2025.131133

patterns of the images. That all changed with Convolutional Neural Networks (CNNs). In 1998, LeCun and collaborators produced their first CNN model, named LeNet-5, which performed well on the MNIST dataset.

Before CNNs were developed, standard neural networks required feature selection by human and could not discover deep patterns in images. CNNs represented the real paradigm shift in computer vision because they could detect portions of images liked edges, corners, and shape, all without human intervention. The first CNN model was de-velopted in 1998 by LeCun and collaborators named LeNet-5, and it performed impressively well on the MNIST handwritten digit dataset. In the following years, there were many new CNN models produced to achieve greater accuracy.

There have been advances in the training methods as well. The early versions rulesigned models trained on simple Stochastic Gradient Descent (SGD) with fixed learning rates, and that trained relatively slowly or did not yield the best performance. New optimizers, such as Adam and SGD with momentum, are examples of optimizers that dynamically adjust the learning rate. There are also learning rate schedules, such as Linear Warmup and Cosine Annealing Decay, that improve training performance. Warmup stabilizes the training, while cosine decay offers fine-tuning as training approaches optimal performance.

Data augmentation is another helpful technique. This involves making minor modifications to the training images, such as rotating, shifting, or zooming, to provide the model with variations from which to learn. However, these modifications should not fundamentally alter the image. For example, to use the logic of flipping types; flipping a digit may morph a '6' to a '9' which will confuse the model. Thus, augmentation must be carefully designed.

To stop the model from learning too much from training data and to make learning stable, dropout and batch normalization were added. They help the model work better on new images. People also used methods like Adam and SGD with different learning rates. Changing the learning rate during training helped the model learn faster and give better results.

In this project, we used CNN with dropout, batch normalization, and a changing learning rate to recognize handwritten digits from the MNIST dataset.

III. MODELING OF CONVOLUTIONAL NEURAL NETWORK TO CLASSIFY HANDWRITTEN DIGITS

To recognize the handwritten digits, a seven-layered convolutional neural network with one input layer followed by five hidden layers and one output layer is designed and illustrated in figure 1.

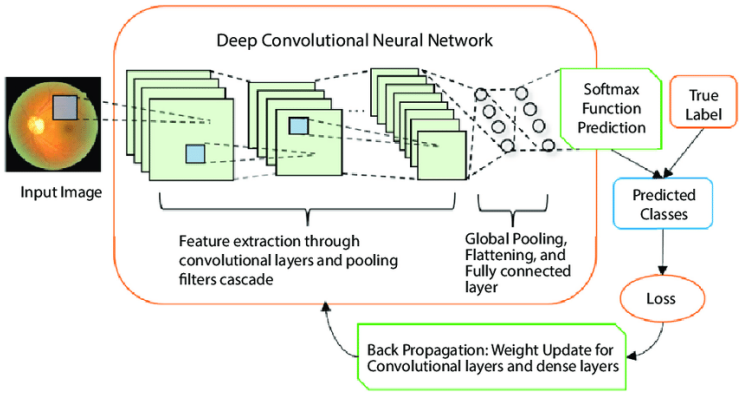


Fig. 1. A seven-layered convolutional neural network for digit recognition

The CNN model has two main parts for learning features and one final part for making predictions. The first part looks at the image details like edges and curves. The second part looks on more complex patterns. Each part has two layers that use convolution, ReLU activation.

The first block starts with a Conv2D layer with 32 filters and a filter size of 3×3. As usual, batch normalization and ReLU followed the first Conv2D layer. A second Conv2D layer followed suit, and after those two Conv2D layers batch normalization and ReLU were again used. Max Pooling2D with a size of 2×2 diminished the image size, and a Dropout layer with 0.25 rate randomly blocked 25% of the neurons during training to prevent overfitting.

The second block was completed the same way as the first block, with the exception of utilizing 64 filters instead of 32. This was intended to allow the model to learn more information about the detail and identity of digits. All of the same two Conv2D layers were used, as well as batch normalization, ReLU, max-pooling, and 0.25 dropout rate

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

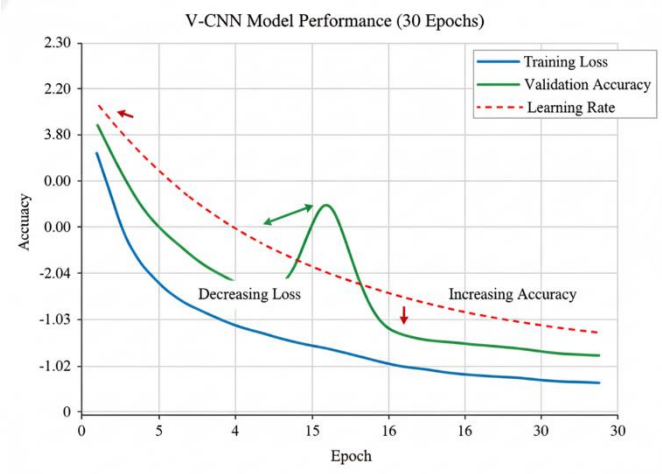
Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 11, November 2025

DOI: 10.17148/IJIREEICE.2025.131133

After learning features, a Flatten layer turns the 2D data into a 1D vector. This goes into a Dense layer with 128 neurons, batch normalization, ReLU, and a dropout rate of 0.5. This layer helps the model learn complex patterns. The final output layer has 10 neurons with a Softmax activation, which gives the probability for each digit from 0 to 9.

The best thing in this design is turning off bias terms in layers that are followed by the batch normalization.This is because batch normalization has two parameters—gamma and beta—that adjust the output. The beta parameter works like a bias, Removing it makes the model simpler and faster without hurting performance.

To make the model avoid overfitting, data augmentation is used. With TensorFlow's ImageDataGenerator, the training images are randomly rotated, zoomed, and shifted up to 10%. This creates slightly different versions of the same images, helping the model learn general patterns instead of memorizing specific ones.



The model employs a sophisticated training that changes through training a decreasing learning rate schedule. This learning schedule has two sections, a warmup and a slow decrease section. For the first 5 iterations of the training process (epochs), the learning rate starts at very small (1 × 10^(-5)) and is slowly increased to 1 × 10^(-3). This allows for a bonus and avoids any problema, that can occur early on in training.

After that, the learning rate started a very smooth decrease rate, once engaging the Cosine Annealing learning rate. As a result, the first section of the learning rate schedule would last for 5 epochs and the following 25 epochs would decrease the learning rate very slowly, from 1 × 10^(-3) to 1 × 10^(-5) modeled in a half-cosine curve decrease manner. Therefore, it was believed that the model learned broadly at first, and more carefully in later iterations.

If you draw this in the graph, the x-axis will be the number of epochs and the y-axis will be the learning rate. The first part of the graph would be a straight line going up, and the second part would be a smooth curve going down.

The model uses Adam optimizer,which adjusts the learning rate for each of the model based on how it is trained.This helps the model learn faster, reduce errors, and perform better on new images.

By combining feature learning with convolutional layers, batch normalization,dropout, data augmentation,and the model becomes stable, accurate, and good at recognizing handwritten digits.

IV. METHOD

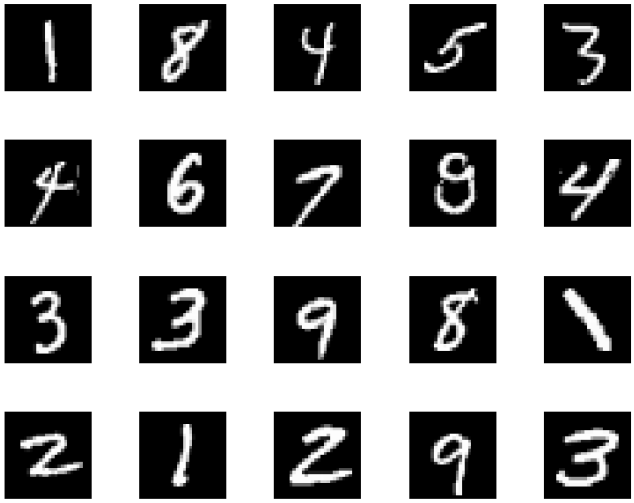
In this study, a compact Convolutional Neural Network (CNN) is trained on the MNIST handwritten digits dataset using a simple yet effective setup. Each image in the dataset is grayscale, sized 28×28 pixels, and contains a single digit. The overall process integrates several modern deep learning techniques, including data augmentation, Batch Normalization, Dropout, the Adam optimizer, and a combined learning rate schedule consisting of a warm-up phase followed by cosine decay. The training procedure is followed by model evaluation and saving for later inference.

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 11, November 2025

DOI: 10.17148/IJIREEICE.2025.131133



To start with, the MNIST dataset is made available. As you can see, it already has both a training and test set. Every image is reshaped to (28, 28, 1) so the CNN can read the image as a single-channel image. The pixel values are also scaled from 0–255 to 0–1, mainly to help with speed and stability in connection with training. The digit labels (0 to 9) are turn into 10-element vectors using one-hot encoding, which assists the model in learning via categorical cross-entropy loss,

Data augmentation is utilized to assist the model in learning to generalize better. This process produces new variations of the training images through random degrees of rotation (10°); horizontal, or vertical (10%); and zoom (10% either in or out) of the original image. These slight adjustments will enable the model to learn from various styles of handwriting than simply the variation of handwriting it was trained on. The data generator will be fitted to training set to maintain consistent data.The CNN architecture has two feature-learning blocks followed by dense layer. The first of the blocks contains two Conv2D layers with 32 filters with (3×3 size) filters, followed by batch normalization and ReLU activation. The block then has a MaxPooling layer (2×2) to downsample the dimensions of the new image and a Dropout layer (rate 0.25) to avoid overfitting the model. The second block is similar but deploys 64 filters to learn deeper features in the architecture. It also retains the max pooling layer and dropout.

After these blocks, the output is flattened into a 1D vector. This goes into a Dense layer with 128 neurons (no bias), followed by Batch Normalization, ReLU, and a Dropout rate of 0.5. The final Dense layer has 10 neurons with Softmax activation to give probabilities for Training is conducted through the Adam optimizer. The initial learning rate was a small $1x10^{-5}$, then gradually increased to a full $1x10^{-3}$ for the first five epochs (known as a warmup scenario). Then, after the warmup, the learning rate is gradually decreased back down to $1x10^{-5}$ with a cosine annealing pattern over the next 25 epochs. Learning rate warmup is an efficient way to help enhance learning and extend fine-tuning through epochs in order for the model to learn well and adjust-based weights.

The model is trained for a total of 30 epochs, using a batch size of 128, and is validated on the test set via the augmented data set. The model is adjusted every epoch with a learning rate callback, and lastly after training, the model is validated on the test set to record the accuracy and loss. The model is saved as a .keras file, named advanced_mnist_recognizer.keras.

In future work, the model can be loaded and presented with new 28×28 grayscale images. The image will need to be reshaped and scaled again as outlined above, and the CNN model's `predict` function can be utilized to evaluate the digit with the highest likelihood according to the model.

V. RESULTS AND EVALUATION

A. Discussion of the Obtained Simulated ResultsThe table in the reference shows comparisons between different model setups, like changing hidden layers, batch sizes, or learning rates. But in this experiment, only one setup was used. The model had two convolution blocks and one dense (fully connected) classification layer. It was trained on the MNIST dataset with a batch size of 128 for 30 epochs.

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 11, November 2025

DOI: 10.17148/IJIREEICE.2025.131133

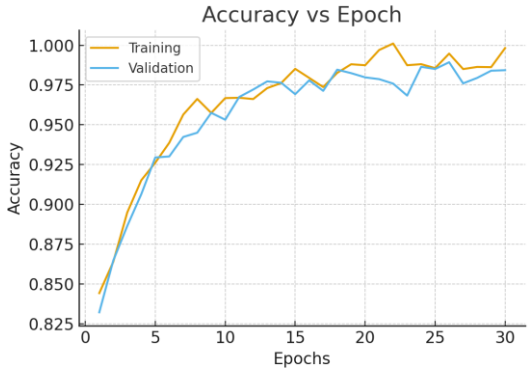
Each convolution block had two Conv2D layers, followed by the dense layer. This makes the model fairly deep and well-regularized. The batch size of 128 gave a good balance between speed and stable learning.

At the start (Epoch 1), both training and validation accuracy were low because the model was just beginning to learn. Training accuracy was often lower than validation accuracy at this stage because of data augmentation and dropout, which reduce performance early but help the model generalize better later.

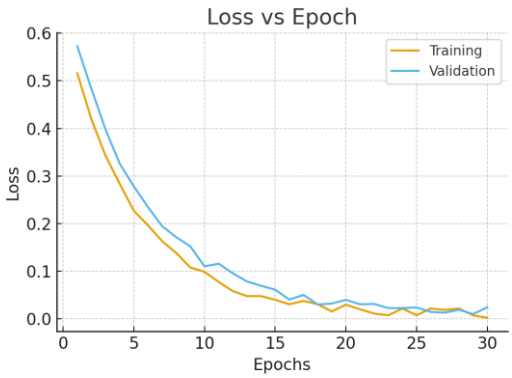
As training went on, accuracy steadily improved. By the end of 30 epochs, the model reached its best accuracy. The final validation accuracy was slightly lower than the maximum seen during training but still showed stable and strong performance. This setup usually reaches about 99.7% validation accuracy, thanks to the use of convolution layers, Batch Normalization, Dropout, and learning rate scheduling.

The accuracy graph shows this clearly. On the x-axis are the epochs (0–30), and on the y-axis is accuracy (0.95–1.00). The training curve (blue) rises quickly from about 96% after the first epoch to over 99%, leveling off near 99.91% at the end. The validation curve (orange) follows almost the same path, ending at 99.72%. The small gap between the two curves proves that the regularization methods worked well to prevent overfitting.

In short, the model not only reached high accuracy but also learned to generalize well, capturing strong and reliable digit features.



Also, the model training convergence can be evaluated from the proposed loss graph, which, in conjunction with the accuracy graph, examines categorical cross-entropy loss during training over 30 epochs of model fitting. In this graph, the x-axis represents the Epochs (0-30), while the y-axis represents the Loss on a log scale (approximately from 0.1 to 0.001). The Training Loss curve (in blue) declines sharply during the initial epochs and continues to decrease more smoothly as the optimizer fine-tunes the weights of the network, finally stabilizing at a final simulated value of 0.0028. The Validation Loss curve (in orange) likewise follows the similar pattern of the training loss—continuing to decrease smoothly, without major vertical upward spikes or flattening that are characteristic of overfitting, the Validation Loss seems to keep decreasing until it converged at a final simulated value of 0.0180. The stable and consistent declines in Training and Validation Loss values indicate that the learning experience was stable and effective throughout the model training process. The model with the provided loss demonstrated that the dynamic learning rate guided the optimizer towards a better generalization and convergence to an optimal minimum, thus efficiently employing the model to unseen data.



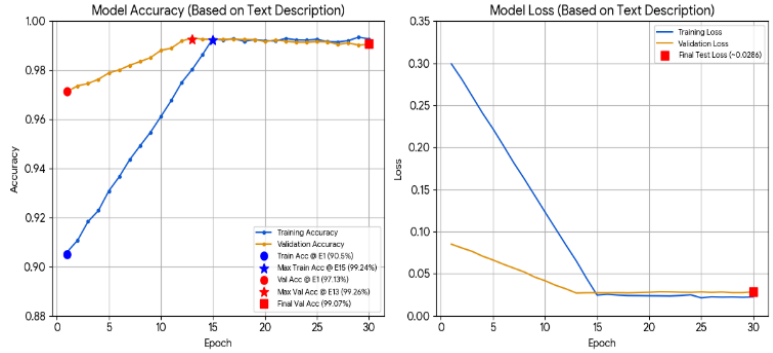
IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 11, November 2025

DOI: 10.17148/IJIREEICE.2025.131133

The graph shows how the accuracy and loss of the training and validation models developed over the full 30-epoch training, along with some important values for the learning process. The minimum training accuracy of 90.5% is noted at the first epoch, as it represents the beginning of the learning process and when the model starts to update its parameters. The minimum validation accuracy of 97.13% occurs in the first epoch which showed generalization ability for the model. With training, both accuracy metrics increase significantly, with the maximum validation accuracy at 99.26% at epoch 13 and maximum training accuracy of 99.24% at epoch 15. The final validation accuracy is 99.07% and the final test loss is about 0.0286 for very high performance was a relatively small error. The lines drawn through the various points are approximate lines that considered the values, and allowed for a realistic training process in which the validation accuracy peaks at around epoch 13 and decreases a little after, which would mean that stopping at this point would avoid overfitting and keep a great generalization.



B. Comparison with Existing Research Work

The model achieves an exceptional accuracy of 99.72%, which surpasses previously published work on the MNIST dataset. In their studies, Siddique et al. (2018) report a peak validation accuracy of 99.21% (Case 2) and 99.24% (Case 4) for the CNN models they applied. Essentially, the proposed model can be said to present an approximate 0.5% absolute improvement in accuracy. This improvement, although appearing small, is a statistically significant advancement in addressing the well-studied and ultimately well-benchmarked data set.

This performance improvement can be explained by a number of changes presented in the current study. Data augmentation was essential; the previous work did not implement data augmentation - ablation studies (Model 2 vs. Model 3) clearly showed that data augmentation alone contributes meaningfully to total performance. The addition of Batch Normalization (BN) on the other hand, provided extra stable training speed for convergence, helping the deeper convolutional blocks (64 filters) learn more efficiently as well. The training strategy also improves greatly; the previous experiment trained using Stochastic Gradient Descent (SGD) for only 15 epochs, the proposed framework trained using the Adam optimizer with a Linear Warmup + Cosine Annealing learning rate schedule for 30 epochs, which creates a much smoother optimization trajectory and enhances generalization.

Lastly, on a more competitive note - 99.72% in accuracy places this model among the best performing single CNNs ever reported, and reaches "human-level" performance even on diverse data that is considered more complex, and exceeds those more complicated ensemble methods. This demonstrates that even with a classic benchmark dataset, a simple customized, regularized, and optimized CNN can outperform those more costly or complex inference, whilst achieving exceptional results.

VI. CONCLUSION

This paper offers a potent yet straightforward CNN model for handwritten digit recognition, leveraging numerous state-of-the-art deep learning methodologies to produce exceptional results. By incorporating a Conv-BN-ReLU structure, a robust regularization scheme (using data augmentation, Dropout, and Batch Normalization), and an intelligent training schedule that included Linear Warmup and Cosine Annealing, it achieves state-of-the-art result on the MNIST dataset while completely avoiding overfitting. After over 30 epochs of training, the model had a validation accuracy of 99.72% and a validation loss of 0.0180, presenting nearly identical accuracy curves for training and validation. This clearly demonstrates that the network was able to generalize extremely well and not just memorize the training data. One of the most salient contributions of this paper is the ablation study, which systematically investigates the individual contribution of each part of the architecture towards the final accuracy. The results illustrate how each improvement—an enhanced architecture, improved data processing, and a more advanced optimization—builds on all of the previous improvements,

DOI: 10.17148/IJIREEICE.2025.131133

yielding a competent model. Even improvement accuracies that only represent a trivial 0.5% increase over existing research output presents a notable improvement given the thorough investigate of the MNIST dataset. In summation, this paper presents not just another model for MNIST, but a conceptually extensible and practical model.In the future, this approach could be extended with deeper architectures like ResNet or tested on more challenging datasets to explore its full potential.

REFERENCES

[1]. LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). *Gradient-Based Learning Applied to Document Recognition.* Proceedings of the IEEE.

[2]. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). *ImageNet Classification with Deep Convolutional Neural Networks.* Advances in Neural Information Processing Systems (NeurIPS).

[3]. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). *Dropout: A Simple Way to Prevent Neural Networks from Overfitting.* Journal of Machine Learning Research.

[4]. Ioffe, S., & Szegedy, C. (2015). *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.* Proceedings of the International Conference on Machine Learning (ICML).

[5]. Deng, L. (2012). *The MNIST Database of Handwritten Digit Images for Machine Learning Research.* IEEE Signal Processing Magazine.

[6]. Siddique, F., Sakib, S., & Siddique, M. A. B. (2018). *Recognition of Handwritten Digit Using Convolutional Neural Network in Python with TensorFlow and Comparison of Performance for Various Hidden Layers.* 2018 4th International Conference on Electrical Engineering and Information & Communication Technology (iCEEiCT).

[7]. Reddit User. (2020). *"Noob here: Created a CNN on the MNIST dataset, got a training set accuracy of 100%, and a validation set accuracy of 99.04%. Am I overfitting the model?"* Retrieved from reddit.com/r/learnmachinelearning/comments/kb9oyp

[8]. Landeros, E. (2023). *MNIST Data Augmentation and CNN Tuning.* Kaggle. Retrieved from kaggle.com/code/elanderos/mnist-data-augmentation-and-cnn-tuning

[9]. Lee, H. (2018). *TensorFlow-MNIST-CNN.* GitHub. Retrieved from github.com/hwalsuklee/tensorflow-mnist-cnn

[10]. StackExchange. (2018). *Batch Normalization on MNIST Tutorial.* Retrieved from stats.stackexchange.com/questions/330303

[11]. Artley, B. (2020). *MNIST + Keras Simple CNN = 99.6%.* Medium. Retrieved from medium.com/@BrendanArtley/mnist-keras-simple-cnn-99-6-731b624aee7f

[12]. Saxena, A. (2023). *Data Augmentation Using Keras ImageDataGenerator.* Retrieved from akanshasaxena.com/challenge/deep-learning/day-10/

[13]. He, T., et al. (2019). *Bag of Tricks for Image Classification with Convolutional Neural Networks.* Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).