Le Net-like for Handwriting Recognition on MNIST Dataset

Anusha. D
Department of Industrial Engineering
University of south Florida

Abstract

Image recognition methods and applications have become widely used and are constantly being developed around the world with the help of Machine Learning techniques. The MNIST database is a globally available dataset that is used to compare the performance and accuracy rates of various models to ensure continuous progress as a result of global competition among developers.
The MNIST dataset contains 60,000 training images and 10,000 test images of handwritten digits that have been normalized and anti-aliased to allow Machine Learning developers to compare the performance of their application to that of others.

This explains the theoretical and practical methods of a straightforward solution to this problem using the Convolutional Neural Network, a suitable model for image recognition. Once the data has been preprocessed, the model may be applied using the user-friendly Keras package function. To achieve the optimal outcome, the model's several layers should be designed in a sensible order.

1.Introduction

Nowadays, image recognition is becoming an essential component in practically every branch of study. In biology, it is used for gray-level transformation, binarization, image filtering, segmentation, visual object tracking, optical flow, and picture registration.[3] Furthermore, in microbiology, recognition of microscopic pictures is used to identify, count, categorize, or predict the form, type, or number of microorganisms. This has allowed for new types of microscopy investigations that can generate enormous image files. [4].

For instance, it is employed in automated protein crystal recognition computer-assisted morphological sperm analysis, and bacteria picture recognition [5] Furthermore, take note of the fact that "the majority of automated image processing systems are trained for particular kinds of microscopy, contrast techniques, probes, and even cell types. This places heavy limitations on experimental design, restricting their use to the specific subset of imaging techniques for which they were intended.

One of the most recent and widely used models is Convolutional Neural Network (CNN, or ConvNet), a feed-forward deep neural network that has been effectively applied to the analysis of visual data. However, while fully connected neural networks can be used to learn features and categorize data, they are not practical for image processing [1] Even with a non-deep architecture, many neurons are required, as are many filters for different locations for the same pattern, which are not optimized and make no sense to employ.

Convolutional Neural Networks are intended to detect visual patterns directly from pixel pictures, with minimal preparation. They can detect a wide range of patterns, including handwritten characters, which we will use to analyze the MNIST dataset.
Convolutional Neural Networks were inspired by biological processes based on the connectivity pattern between neurons in the animal visual cortex, with different neurons' receptive fields partially overlapping to encompass the full vision field. hence they appear to be better suited for image recognition challenges.

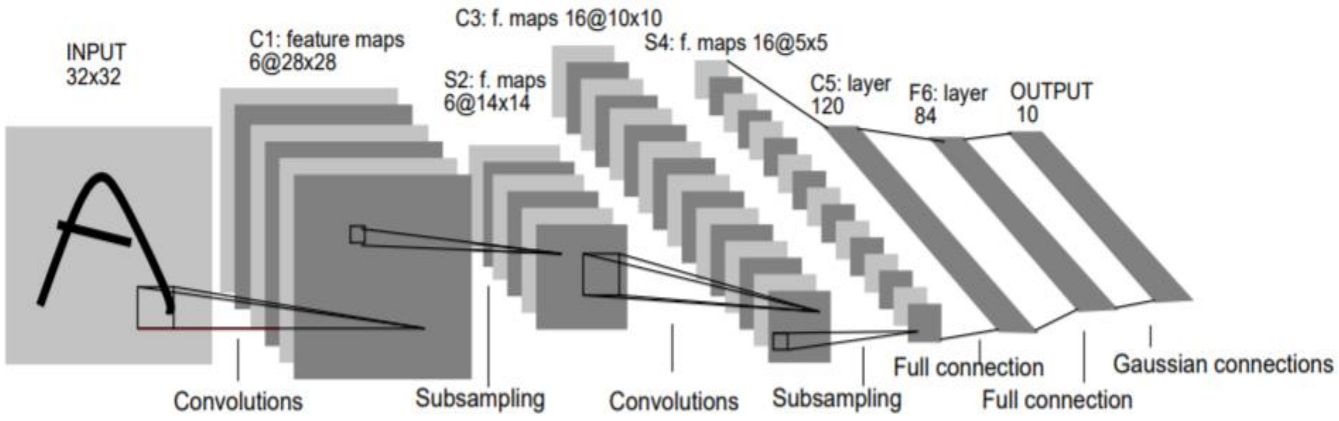


Figure. 1. The architecture of LeNet-5, CNN. The planes are the feature map consist of units of same weight but identical to other plane

2.Related Work

The architecture consists of convolutional layers, pooling layers, and fully connected layers, with the original activation function being tanh (later replaced by ReLU in modern implementations). LeNet became a benchmark model for the MNIST dataset, a widely used dataset for evaluating machine learning models on handwritten digit recognition.

Many subsequent architectures were influenced by LeNet, including AlexNet, VGG, and ResNet, which extended the principles of convolutional networks to larger-scale image recognition tasks Researchers have kept experimenting with LeNet variations, adjusting the number of convolutional layers, filter sizes, and pooling strategies. Furthermore, methods like dropout have been applied to lessen overfitting, guaranteeing that the network is resilient across various datasets.

Handwriting recognition has significantly improved in recent years due to major advancements in CNN technology. Advancements such as attention mechanisms enable networks to concentrate on particular areas of an image, and deeper networks have proven to be more accurate on more difficult recognition tasks. Building on LeNet's pioneering work in handwriting recognition, recurrent neural networks (RNNs), including Long Short-Term Memory (LSTM) units.

These developments show how early innovations in CNNs have informed and driven subsequent advances in deep learning and computer vision, highlighting the long-lasting influence of LeNet's architecture.

Why LeNet-like Architecture: The LeNet-like architecture offers simplicity and efficiency, making it a suitable choice for the MNIST dataset and handwriting recognition tasks.

In this code, a LeNet-like convolutional neural network (CNN) with dropout is implemented for handwritten digit recognition on the MNIST dataset. The key contributions of this implementation are as follows:

Model Architecture: The LeNet With Dropout class defines a CNN architecture consisting of two convolutional layers followed by max-pooling layers and three fully connected layers. Dropout regularization is applied to the fully connected layers to prevent overfitting.

Data Augmentation: The training dataset is augmented with random rotation transformations to improve the generalization of the model. This helps the model to learn invariant features and reduces the risk of overfitting.

3.Methods

3.1. Convolution

Convolution is a mathematical process that results in a third function, which is often displayed as a modified version of one of the previous functions, and can be used as a general matrix multiplication in at least one layer.

$$s(t) = (x * w)(t) \quad (1)$$

Generally, in CNNs, “x” which is the first argument is the input and the second argument “w” is the Kernel.

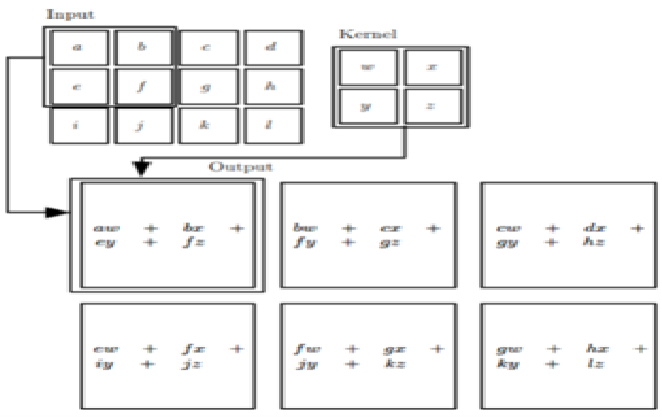


Figure 2. Representation of how convolution network works

3.2 Characteristics of CNN

Aside from the convolution process, CNN has some important specifications that distinguish it from other neural network approaches:

1. Sparse interactions: In contrast to typical NNs, not every input interacts with every output. CNN can be achieved by significantly reducing the size of the kernel relative to the input.
2. Parameter sharing: Throughout the input image, parameters have the same number as the kernel (m*m).
3. Pooling: This method essentially summarizes certain output locations statistically. For instance, max pooling is frequently used to increase pattern contrast by returning the maximum output within a specific rectangular neighborhood.
4. Weight: The hidden layers' weights should be moved in space but should be the same as those of their neighbors. and all other weights are 0, except for the tiny receptive field.

4.Experiments

4.1 Dataset

The Modified National Institute of Standards and Technology database, or MNIST database, is a collection of handwritten digits that was initially created to identify US post codes. This gave machine learning developers access to broad samples for image recognition techniques so they could assess how well various machine learning models performed.
These samples are essentially divided into two sets: 60,000 examples for training and 10,000 examples for testing. The digits were size-normalized, centered, and anti-aliased in a fixed-size image with a 20x20 pixel bounding box, resulting in grayscale levels. It is a fantastic database for folks who want to test learning techniques and pattern recognition algorithms on real-world data with minimal preparation and formatting. The data has already been divided into training, validation, and test sets, making them relatively simple to use.

4.2 Packages and Preprocessing

The code is written in Google Collab using python, and the relevant packages in Python for our purposes are used such as: Keras the most popular Python deep learning library. Keras can run on top of TensorFlow, CNTK, and Theano, which are all open-source software libraries for numerical computation (Keras-users, 2017); however, TensorFlow was chosen because of its special GPU support feature, which makes it much faster to compute as parallel processing. Scikit-Learn is a set of simple and efficient tools for data mining and analysis.
Other packages include Matplotlib for plotting and NumPy, which is the foundational package for scientific computing in Python. First, the MNIST data can be obtained as a URL request, which can then be separated into train, validation, and test data using the pickle function. To show how these handwritten numerals look like, Matplotlib is utilized.

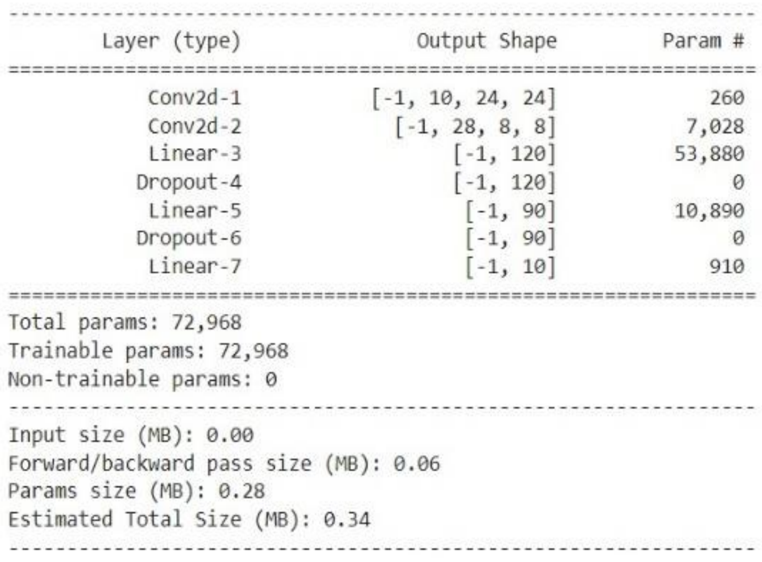
4.3 The Model

The model used is the classic LeNet-5 architecture, originally proposed by Yann LeCun et al. for handwritten digit recognition tasks. This modified LeNet variant incorporates dropout layers, aiming to enhance the model's generalization capabilities and mitigate overfitting.

The architecture consists of two convolutional layers followed by max-pooling layers, two fully connected layers, and an output layer. The first convolutional layer takes input grayscale images of size 28x28 and applies 10 filters with a kernel size of 5x5. The second convolutional layer employs 28 filters with the same kernel size. Both convolutional layers are activated using the Rectified Linear Unit (ReLU) activation function.

Max-pooling layers with a kernel size of 2x2 follow each convolutional layer, reducing the spatial dimensions of the feature maps while retaining the most salient features. The output of the second max-pooling layer is flattened into a vector and passed through two fully connected layers, each followed by a ReLU activation function and a dropout layer with a dropout probability of 0.5.

The first fully connected layer has 120 units, and the second has 90 units, gradually reducing the dimensionality of the feature representation. Finally, the output layer consists of 10 units, corresponding to the 10 classes of digits in the MNIST dataset. A SoftMax function is typically applied to the output layer to obtain class probabilities.
This model architecture aims to strike a balance between model complexity and capacity, effectively capturing the hierarchical features present in handwritten digits while incorporating dropout regularization to prevent overfitting during training.



```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 10, 24, 24]             260
            Conv2d-2             [-1, 28, 8, 8]           7,028
            Linear-3                  [-1, 120]          53,880
           Dropout-4                  [-1, 120]               0
            Linear-5                   [-1, 90]          10,890
           Dropout-6                   [-1, 90]               0
            Linear-7                   [-1, 10]             910
================================================================
Total params: 72,968
Trainable params: 72,968
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 0.06
Params size (MB): 0.28
Estimated Total Size (MB): 0.34
----------------------------------------------------------------
```

Figure 3: Model Summary

4.4 Training

1. Data Augmentation: Data augmentation techniques like random rotation are employed during training to artificially increase the diversity of the training set, helping the model generalize better to unseen data.
2. Normalization: Both training and test datasets are normalized using the mean and standard deviation of the MNIST dataset. Normalization helps in stabilizing the training process by bringing features to a similar scale.
3. Model Architecture: The model architecture used is a modified version of the LeNet-5 architecture, with added dropout layers. Dropout layers help in reducing overfitting by randomly dropping a fraction of the units during training.
4. Loss Function and Optimizer: The Cross-Entropy Loss function is used, which is suitable for multi-class classification problems. The Adam optimizer is employed with a learning rate of 0.0005.
5. Learning Rate Scheduler: A learning rate scheduler is utilized to adjust the learning rate during training. In this case, the learning rate is decreased by a factor of 0.5 every 10 epochs.
6. Training Loop: The training loop iterates over the specified number of epochs. In each epoch, the model is trained using batches of data from the training set. The optimizer is used to update the model parameters based on the computed loss.
7. Evaluation: After each epoch, the model is evaluated on the test set to monitor its performance. The accuracy on both the training and test sets is calculated and printed.
8. Confusion Matrix: At the end of training, a confusion matrix is generated using the model's predictions on the test set. This matrix provides insights into the model's performance across different classes.
9. Visualization: The code includes visualization of the training and test accuracies over epochs, providing a graphical representation of the model's learning progress.
10. Visualizing Predictions: Lastly, the code visualizes the predictions made by the model on a batch of test images, displaying the predicted labels alongside the ground truth labels for comparison.

4.5 Metrics

We quantitatively evaluate our model’s performance using the following method.

1. After processing all batches in the training and test sets, the total number of correct predictions and the total number of samples in each set are known.
2. The overall accuracy for each set (training and test) is calculated as the ratio of the total number of correct predictions to the total number of samples in the respective set.
3. These overall accuracies are typically expressed as percentages, providing a measure of how well the model performs on the entire dataset.

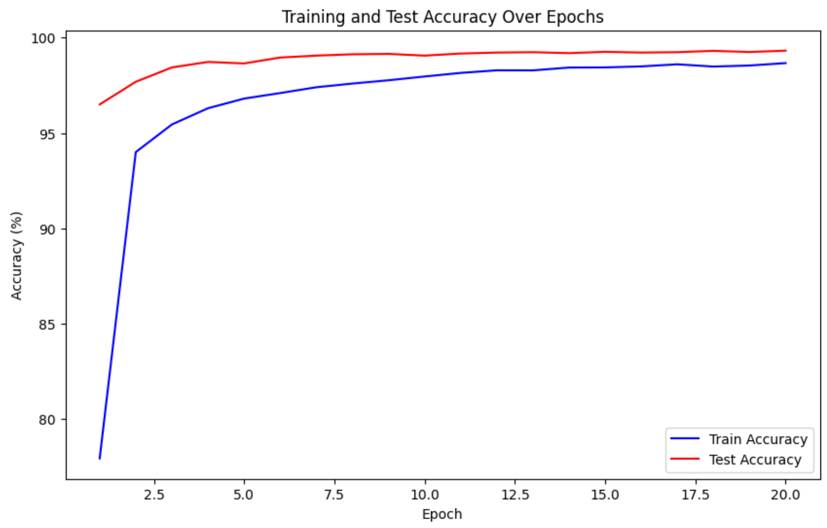


Figure 4: Graphical representation of Training and Test accuracy

4.6 Results

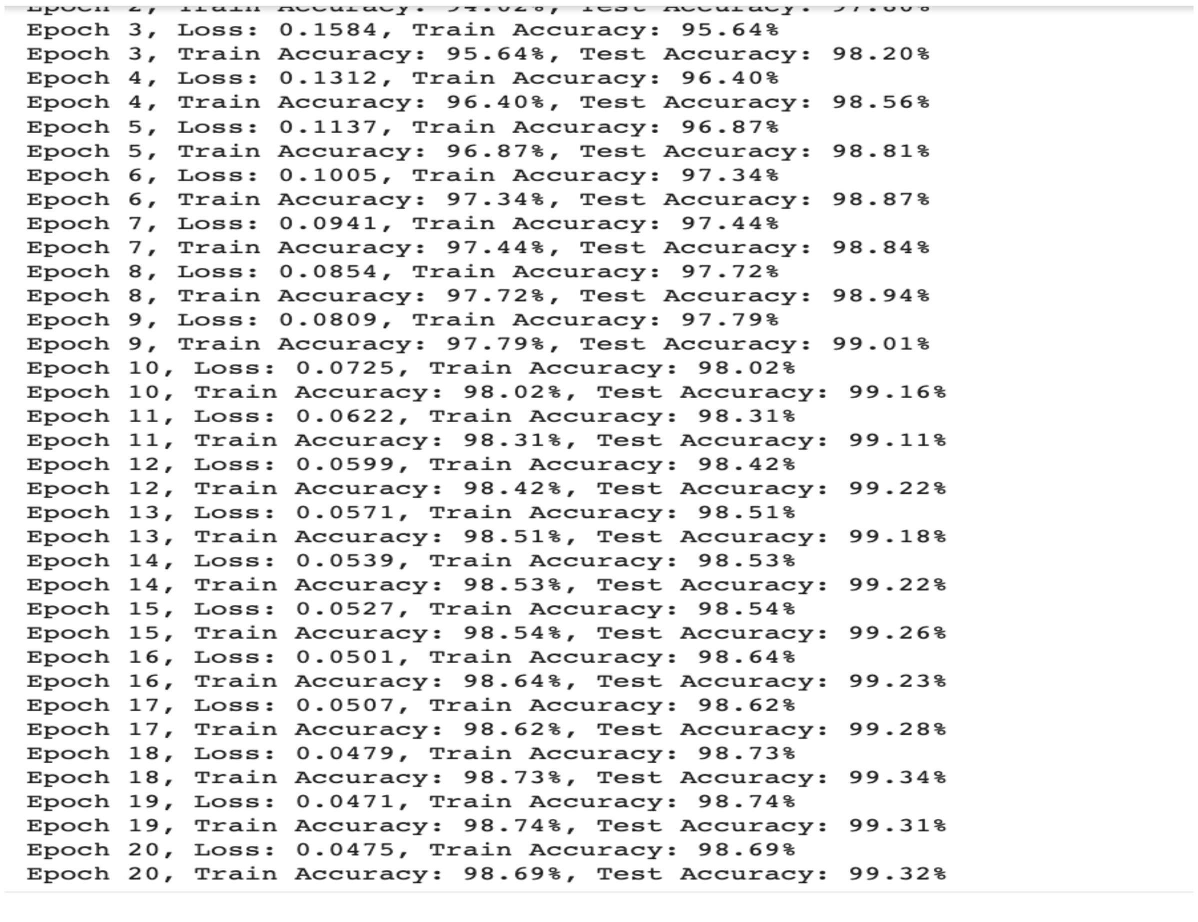
The accuracy of the training set after 20 iteration is 0.9867 and validation accuracy hits 0.9932 which is a very good result on the training and validation set. And the loss value is expected to be very low which is 0.0475.

When we apply the model on the test set, we get a promising result using the evaluation function of Keras. we achieved the accuracy of 0.9932.

The accuracy is a fraction of properly predicted cases; thus the fraction of misclassified cases is equal to 1 - Accuracy which is the error (rate) here.

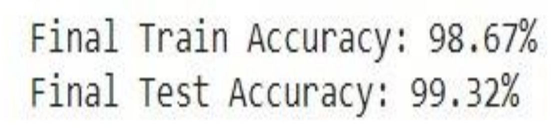
Accuracy rate = 1 - Error rate

The error rate = 1 - 0.9932 = 0.0097 * 100 = 0.97 percent of error rate.



```
Epoch 3, Loss: 0.1584, Train Accuracy: 95.64%
Epoch 3, Train Accuracy: 95.64%, Test Accuracy: 98.20%
Epoch 4, Loss: 0.1312, Train Accuracy: 96.40%
Epoch 4, Train Accuracy: 96.40%, Test Accuracy: 98.56%
Epoch 5, Loss: 0.1137, Train Accuracy: 96.87%
Epoch 5, Train Accuracy: 96.87%, Test Accuracy: 98.81%
Epoch 6, Loss: 0.1005, Train Accuracy: 97.34%
Epoch 6, Train Accuracy: 97.34%, Test Accuracy: 98.87%
Epoch 7, Loss: 0.0941, Train Accuracy: 97.44%
Epoch 7, Train Accuracy: 97.44%, Test Accuracy: 98.84%
Epoch 8, Loss: 0.0854, Train Accuracy: 97.72%
Epoch 8, Train Accuracy: 97.72%, Test Accuracy: 98.94%
Epoch 9, Loss: 0.0809, Train Accuracy: 97.79%
Epoch 9, Train Accuracy: 97.79%, Test Accuracy: 99.01%
Epoch 10, Loss: 0.0725, Train Accuracy: 98.02%
Epoch 10, Train Accuracy: 98.02%, Test Accuracy: 99.16%
Epoch 11, Loss: 0.0622, Train Accuracy: 98.31%
Epoch 11, Train Accuracy: 98.31%, Test Accuracy: 99.11%
Epoch 12, Loss: 0.0599, Train Accuracy: 98.42%
Epoch 12, Train Accuracy: 98.42%, Test Accuracy: 99.22%
Epoch 13, Loss: 0.0571, Train Accuracy: 98.51%
Epoch 13, Train Accuracy: 98.51%, Test Accuracy: 99.18%
Epoch 14, Loss: 0.0539, Train Accuracy: 98.53%
Epoch 14, Train Accuracy: 98.53%, Test Accuracy: 99.22%
Epoch 15, Loss: 0.0527, Train Accuracy: 98.54%
Epoch 15, Train Accuracy: 98.54%, Test Accuracy: 99.26%
Epoch 16, Loss: 0.0501, Train Accuracy: 98.64%
Epoch 16, Train Accuracy: 98.64%, Test Accuracy: 99.23%
Epoch 17, Loss: 0.0507, Train Accuracy: 98.62%
Epoch 17, Train Accuracy: 98.62%, Test Accuracy: 99.28%
Epoch 18, Loss: 0.0479, Train Accuracy: 98.73%
Epoch 18, Train Accuracy: 98.73%, Test Accuracy: 99.34%
Epoch 19, Loss: 0.0471, Train Accuracy: 98.74%
Epoch 19, Train Accuracy: 98.74%, Test Accuracy: 99.31%
Epoch 20, Loss: 0.0475, Train Accuracy: 98.69%
Epoch 20, Train Accuracy: 98.69%, Test Accuracy: 99.32%
```

Figure 5: Epochs run during training the model



```
Final Train Accuracy: 98.67%
Final Test Accuracy: 99.32%
```

Figure 6: Final Train and Test accuracies

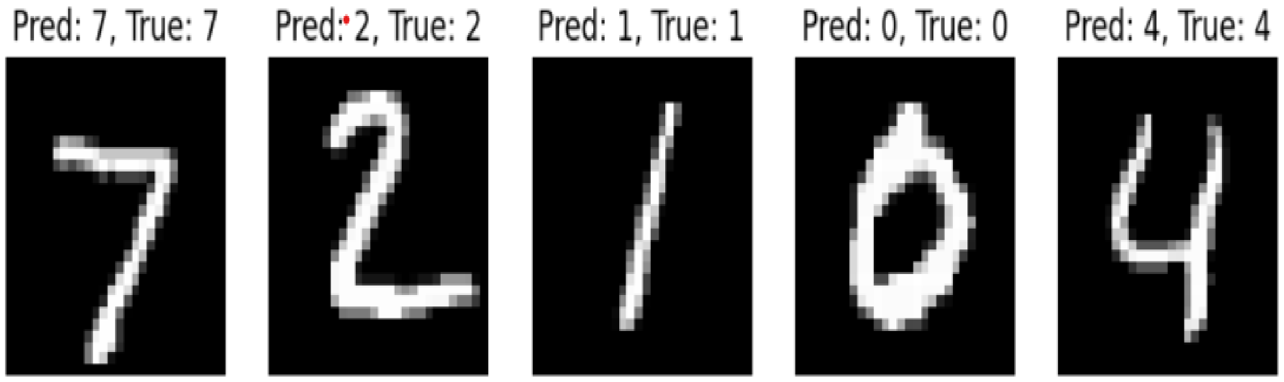


Figure 7: Predicted, True value and their corresponding Images.

5.Conclusion

In conclusion, the implementation of the LeNet-like convolutional neural network (CNN) with dropout regularization for handwritten digit recognition on the MNIST dataset demonstrates several key findings:

Performance Improvement: By applying dropout regularization and data augmentation techniques, the model's performance in terms of both training and test accuracies is enhanced. The dropout layers help prevent overfitting by randomly dropping units during training, while data augmentation introduces variability into the training data, leading to improved generalization.
Robustness: The model exhibits robust performance across multiple epochs, maintaining high accuracy on both the training and test sets. This robustness indicates that the model effectively learns discriminative features from the input images, enabling accurate predictions on unseen data.

References

[1] Aghdam, H. H. and Heravi, E. J. (2017*). Guide to convolutional neural networks: a practical application to traffic-sign detection and classification, volume 1.* Cham, Switzerland: Springer.

[2] LeCun, Y. *et al.* (1998b). The mnist database of handwritten digits. Marsico, A. (2017). Applied machine learning lecture in fu-berlin.

[3] Uchida, S. (2018). Image processing and recognition for biological images.

[4] Shamir, L. *et al.* (2010). Pattern Recognition Software and Techniques for Biological Image Analysis. *PLoS Computational Biology,* **6**, 1–10.

[5] wei Shi, H. *et al.* (2011). Study on method of bacteria image recognition. *IEEE,* pages 15–17.

[6] Marsico, A. (2017). Applied machine learning lecture in fu-berlin