Digit recognition using Logistic regression and CNN

Toker Ahmed

Introduction:

Digit recognition is the idea of recognizing correct digits from human handwritten images. This is a part of computer vision technique where a machine can correctly recognize human handwritten digits from pixel values of images. It is a very interesting and growing field because handwriting varies from person to person, and sometimes it is tough to recognize a digit from a distance such as reading a car's license plate number from a distance is tough for human eye. It can also help to read blurry handwritten sentences that can be easily identified by the digit recognition system, and it can also be used in postal mail sorting, bank check processing etc.

So, a model with high accuracy that can recognize digits, from images can help solve a lot of problems. However, if a model has low accuracy, it can lead to a lot of problems. For an automated bank check processing system, a machine with low accuracy will incorrectly recognize the digits from check. As a result, it will lead to incorrect bank transactions if it incorrectly recognizes the transaction amount. Thats why high accuracy is extremely important for digit recognition.

Moreover, choosing correct approach is extremely essential for this kind of problem. Choosing the correct approach will give the best result possible. For example, K nearest neighbor or clustering technique will not work for digit recognition because K nearest neighbor is an algorithm that is not ideal for learning from the training data. Instead, it memorizes the dataset. On the other hand, clustering is not ideal for digit recognition as it is an unsupervised learning method where it groups the data into clusters. Using them for digit recognition will require additional steps and would not give the best results. So, after complete research I chose CNN and logistic regression method for this problem.

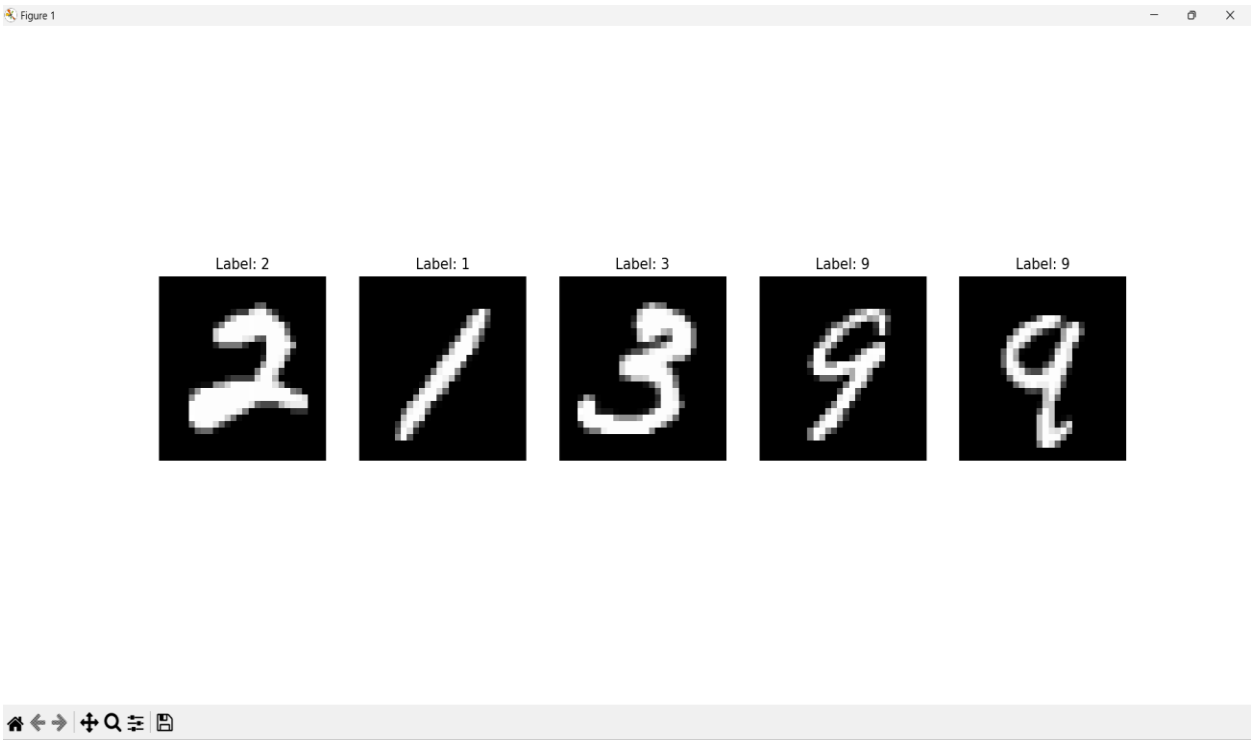
I followed a step-by-step technique to complete this project. I chose the MNIST dataset from Kaggle that I will include with this report along with the source code. The dataset was already very clean and organized and did not require much of the preprocessing other than normalization and reshaping and shuffling for logistic regression. Similarly, for CNN, reshaping, normalizing and one-hot encoding was needed. Then, I worked on training the model using CNN and multinomial logistic regression method. In terms of multinomial logistic regression, I used Softmax function as the loss function. After that, I finally evaluated the model using metrics like precision, recall, confusion matrix, accuracy to evaluate the performance of the model. A similar approach was used for CNN model. Finally, the result of both the model was evaluated and compared to determine which model gave us the best result.

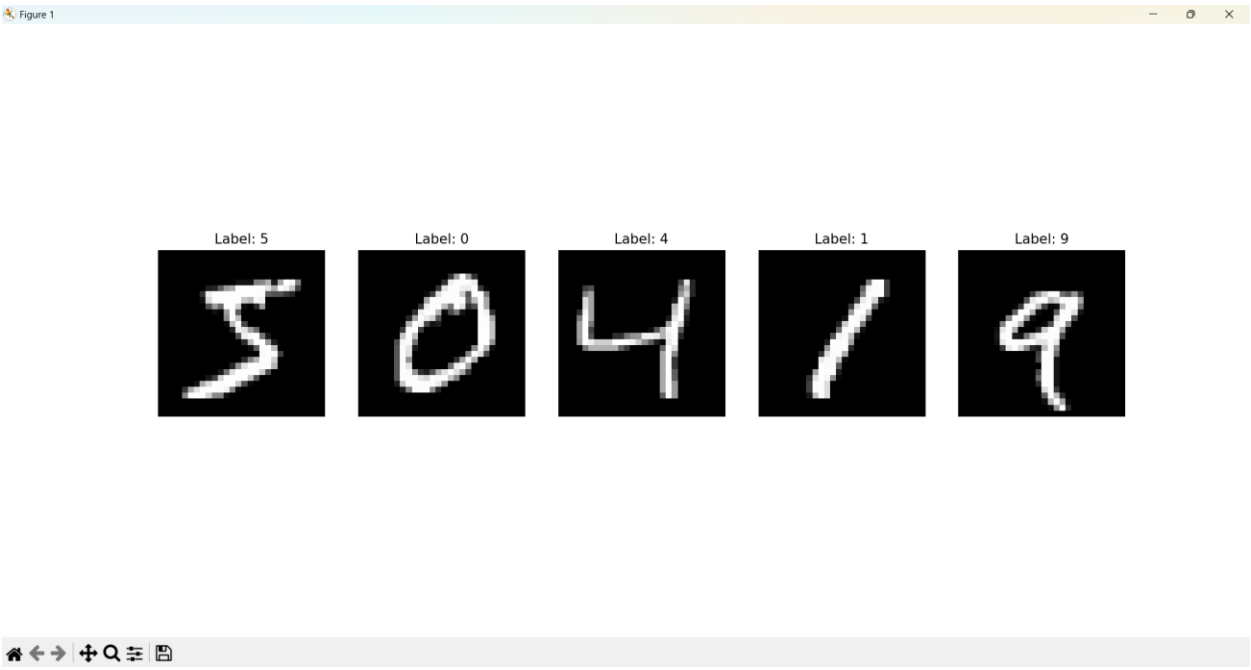
Approach:

The MNIST dataset was already splitted into testing and training set. The training dataset had 60000 rows and the testing dataset had 10000 rows. The dataset had 785 columns. The first column represents label from digits 0-9 for corresponding handwritten images. The rest of the 784 column represents the pixel value for that corresponding 28 by 28 pixel image.

Multinomial Logistic regression approach:

Multinomial Logistic regression approach was taken to classify handwritten digits into multiple classes from digit 0-9. Moreover, this approach was quite simple and easy to implement. It did not require complex parameter tuning and addition of regularization terms. I started with importing all the required libraries such as numpy, pandas, matplotlib, sklearn, and seaborn. Then, I loaded the dataset mnist_train.csv and mnist_test.csv using pandas. Then, I used NumPy library to convert the data into array format for simplicity. After that, I shuffled the data using shuffle() method. Then, I did the separation on training and testing data to prepare them for the model. After the separation, X_train contained the feature data or pixel values of the training set which excluded the first label column. Y_train contained the labels of the training set. X_test contained the feature data or pixel values of testing set which excluded the first label column. Finally, y_test contained the labels or digit classes of the testing set. Then, I used visualization technique using matplotlib library. This was done to check the datasets were correctly loaded or not. For both the training and testing set, first 5 digits were retrieved to visualize. Below figure is the image that was generated to visualize the first digits from training and testing dataset. Thats how the image looks like from the pixel values.





Both the figures represent first five digits from training data and testing data respectively. After the visualization of data, I normalized it by dividing each pixel value by 255 in the training and testing set. It was done to scale the pixel value in range 0 to 1 which would work better for feeding the data into the model. Then, a logistic regression model was initiated with three parameters. The parameters were multi_class= multinomial, solver='lbfgs', and max_iter=1000. Multinomial parameter ensures that it is a multiclass classification as it involves the classification into nine classes as digits 0-9. Lbfgs is considered an optimization algorithm for logistic regression and thus it was used, and the maximum number of iterations was used as 1000 for the algorithm to find the optimal solution. Then, I used fit() function to train the model. After training of the model was done, it was time for prediction. I used predict() function on the normalized data to compute the predictions for both the training and testing set.

Then, I evaluated the model's performance. I used sklearn.metrics library to use accuracy_score() , precision_score, and recall_score finction. After that, I used confusion_matrix function using same library. A heatmap visualization of the confusion matrix was done using seaborn library. It provided a clear presentation of how the model performed with the dataset. So, that was the overall approach using logistic regression.

CNN approach:

After the multinomial logistic regression approach, I chose CNN to solve this problem as from the research it seemed that neural network approach gave a good result for digit recognition problem. CNN uses three layered networks such as input layer, hidden layer, and output layer. Moreover, it requires very minimal data preprocessing. Thats why, I chose CNN method as my second approach. For this approach, I already imported all the libraries needed such as numpy, tensorflow, keras, sklearn, matplotlib and seaborn. The same MNIST dataset was used on this method too as it would help us compare the performance of both the methods. Firstly, I imported the MNIST dataset using TensorFlow keras dataset module. Then, the dataset was loaded into four variables such as X_train, Y_train, X_test,

Y_test. They represent training data with out label, training data with only the label of images, testing data without label, and testing data with only the label of images respectively. After that, the shape of the dataset was printed to make sure the dataset was correctly loaded, and ready. To visualize the data, a similar technique was used using plot to show the first five images from dataset. After that, a similar technique as before was used to normalize the data by dividing each pixel value by 255 in the training and testing set. Then, I used reshape() function to do reshaping of data, and then used to_categorcal function to do one-hot encoding on the data. These two steps were used to make the dataset ideal for multilayer model.

To build the CNN model, I used sequential model. At first, I added a 2D convolutional layer with 32 filters with a size of 3 by 3 pixel. Then, added a max-pooling layer of size 2 by 2 to reduce computational complexity and get optimization. Similarly, another layer was added with 64 filters. Then, another max-pooling layer was added. Finally, a third layer was added with 64 filters to build the CNN architecture model. In the hidden layers, relu function was used as the activation function, and at the output layer, SoftMax function was used because it ensures that the model outputs 10 neurons or classes for digits from 0 to 9. Then, compile() function was used to compile the model after it was built where as optimizer 'adam' was used to minimize the loss function during training. Categorical cross-entropy function was used as the loss function as it was for multi-class classification problem. For metrics, precision, recall and accuracy was used to evaluate the performance of the model. Then, finally, the model was trained using fit() function and with an epoch value of 10. Epoch value was assigned to 10 to make the model perform better and get a higher accuracy. Finally, the process ended with evaluating accuracy, loss, precision, and recall. I also plotted the confusion matrix for this approach to evaluate the models performance. Overall, the approach was quite similar in terms of data loading and preprocessing. However, building the model and compiling the model approach was different for both the methods.

Results:

The result that I found was quite interesting. I did not expect that I would get such a good result for multinomial logistic regression approach. I got an accuracy of 92.63% for multinomial logistic regression approach with a precision of 92.53% and recall of 92.52%, as shown below in the table. Even though, there was no parameter tuning and regularization done still the model gave a very good result in terms of accuracy.

On the other hand, the result from CNN approach was quite satisfying. I got an accuracy of 99.07% from this model with a precision of 99.11% and recall of 99.03%. As expected, the CNN method gave a very high accuracy and performed better than logistic regression approach.

Logistic Regression	CNN
Accuracy=92.63%	Accuracy=99.07%
Precision=92.53%	Precision=99.11%
Recall=92.52%	Recall=99.03%

Both the model's performance was evaluated based on the confusion matrix too. Below both the confusion matrix result was shown for logistic regression and CNN respectively. From the first confusion matrix, it is evident that logistic regression performed incorrectly for some digits. For example, from the figure1, digit 5 was incorrectly predicted as 3 for 35 times. On the other hand, for CNN it happened only once. The reason logistic regression model could not correctly detect the digit is because 5 and 3 almost looks similar and as a result incorrectly predicted 35 times. Similarly for digit 7, it was incorrectly labeled as 2 for 24 times from figure1, but for figure2, it happened for 7 times only. Although, digits from 0-9 were corrected predicted by both models for most of the time. For example, digit1 was predicted as 1 1111 times for regression approach and 1132 times for CNN approach. So, confusion matrix gave us an ideal scenario to compare both the models performance.

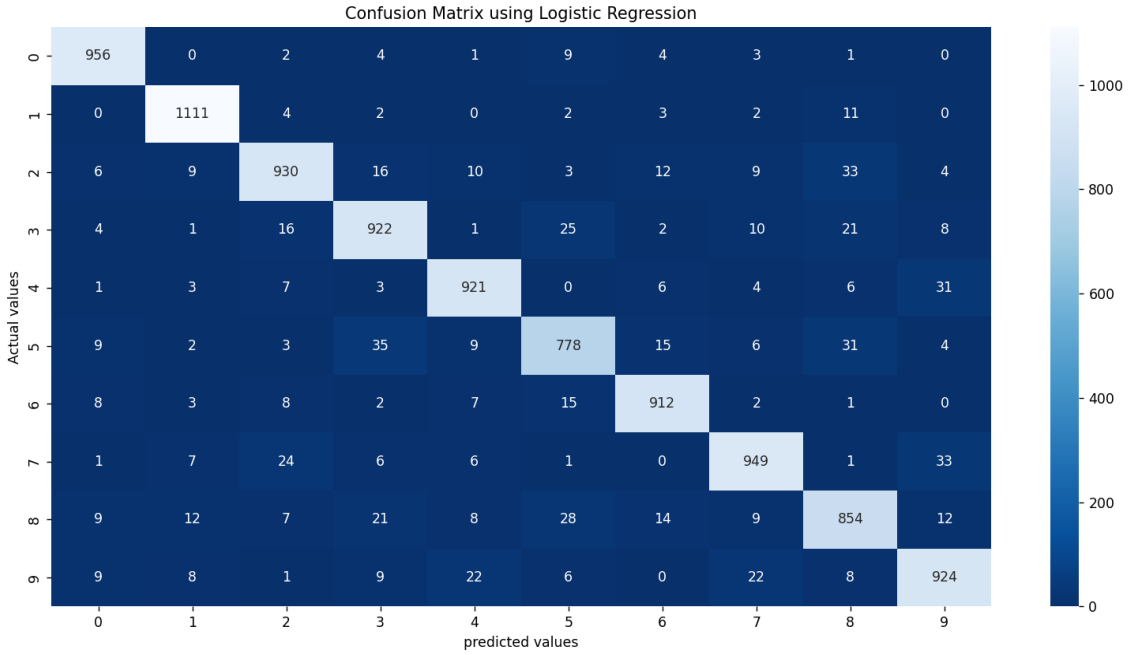


Figure1: Confusion matrix for Multinomial logistic regression

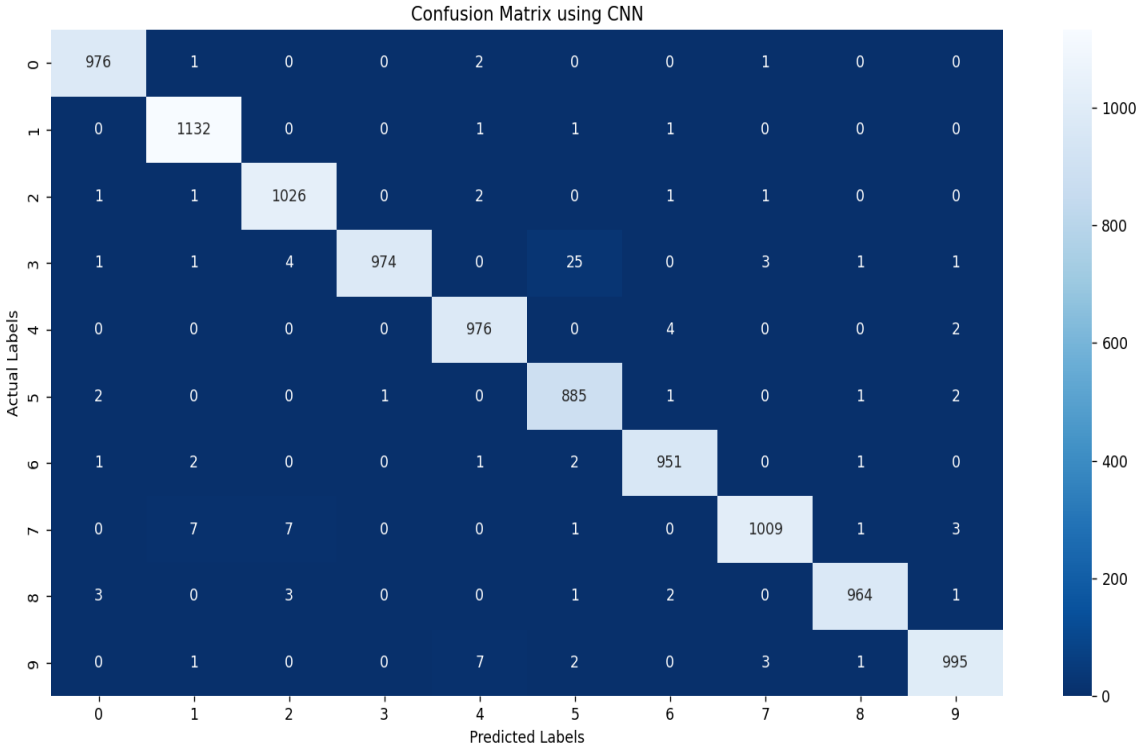


Figure2: Confusion matrix for CNN

Conclusion:

The project's goal was to build two models using multinomial logistic regression and CNN that can correctly recognize digits from handwritten images with a high accuracy. The dataset I used was MNIST dataset that was already splitted into training and testing dataset. I didn't have to do much of the preprocessing other than normalizing, reshaping, and shuffling it to make the dataset ready for training step. Then, for both the approach different technique was used to train the model. For logistic regression, it was quite simple, but for CNN I had to build the CNN model and then compiled the model, and finally trained the model. Finally, the result was computed using accuracy, precision, recall metrics along with confusion matrix for both the methods.

From the output, it is evident that CNN is a better approach for digit recognition than logistic regression because of its multilayered architecture. It gave us an accuracy of 99.07% compared to 92.63% accuracy that came from using logistic regression. It means the model will perform much better if we use CNN approach.

Acknowledgments:

Without the resources, I would not have been able to complete the project. I learned about CNN from sitepoint.com and a research paper that is listed below. I learned about applying logistic regression on digit recognition from the geeksforgeeks website. Also, going over other's project on Kaggle helped me a lot in terms of taking the correct approach, and implementing it. Here is the list of resources that helped me to complete the project:

1. https://www.geeksforgeeks.org/handwritten-digit-recognition-using-neural-network/

2. https://www.sitepoint.com/keras-digit-recognition-tutorial/

3. Dixit, Kushwah, Pashine. Handwritten Digit Recognition using Machine and Deep Learning Algorithms.

4. https://www.kaggle.com/code/rgarg1234/digit-recognizer-cnn/notebook