MNIST Digit Classification Machine Learning Project

1st Aizaz Omer
360569
SEECS, NUST
Islamabad, Pakistan
aomer.mscs21seecs@seecs.edu.pk

2nd Muhammad Qaisar
360662
SEECS, NUST
Islamabad, Pakistan
mqaisar.mscs21seecs@seecs.edu.pk

3rd Muhammad Salman Akhtar
398895
SEECS, NUST
Islamabad, Pakistan
aomer.mscs21seecs@seecs.edu.pk

***Abstract*—In this project, we have implemented a Convolutional Neural Network (CNN) based handwritten digit classifier. CNNs are known to be excellent feature extractors for structured data such as images. They exhibit the desired properties of invariance with respect to geometric and photo metric transformations. While, earlier layers of CNN detect basic features such as edges and corners; the deeper layers detect shapes and contours. Moreover, the back-propagation empowers the CNN to be end-to-end trainable thus simplifying the model building and training process. Keeping in view of the stated benefits of CNNs, we have adopted them into our classifier model. Additionally, we have also demonstrated that image augmentation can be used for making the model more robust and useful in real-life scenarios where it is common to encounter transformed images (rotated, shifted, varying brightness, zoom) rather than especially curated training images dataset.**

I. Introduction

The project is aimed at building a classification algorithm for identifying handwritten digits through a deep learning framework. The problem falls in the Computer Vision domain since handwritten digits in the form of images are required to be classified into corresponding numeric digits. The resultant algorithm can have useful applications since it can be augmented in a number of automation tasks such as number plate recognition and postal address identification.

The introduction of deep learning significantly revolutionized the approach of dividing the problem into independent modules. Deep learning encourages end-to-end learning, thus helps in simplifying and generalizing the task. For Computer Vision, deep learning has introduced a powerful tool in the form of Convolutional Neural Networks (CNN). The ordinary Multi-Layer Perceptron (MLP) architecture is not suitable for images due the structured and spatial relationship of nearby objects in the images. Therefore, a specialized neural network in the form of CNNs have been introduced which exhibit the highly desirable qualities of feature extractors that are capable of handling numerous transformations such as rotation, translation, scaling and brightness variation. CNNs are very good at detecting and combining local features thus constructing feature maps which can be used for any specialized tasks such as classification.

II. Literature Review

Traditionally, Computer Vision algorithms rely on elaborate and discrete pipeline for accomplishing domain specific tasks such as classification, segmentation or object tracking. This pipeline is usually composed of image pre-processing, feature extraction and task specific output modules. Each stage of the pipeline is designed to feed the next stage until the output is obtained. Every stage had its own set of algorithms which had to be tweaked manually according to the problem in hand. For example, specially curated algorithms for feature extraction were used such as SIFT [1], SURF [2] and HOG [3] etc. The extracted features were then passed to the next stage which has its own set of specialized algorithms such as SVM, decision trees and random forest etc. This approach served well to solve domain specific problems of that era but failed to generalize. The slight variation in the dataset requires significant changes in the algorithm pipeline.

In the past, a number of approaches have been demonstrated for classifying handwritten digits. [4] suggested structural feature extractors incorporating a statistical approach to accomplish the task. [5] have proposed the use of k Nearest-Neighbor (KNN) and Radial Basis Function Network (RBF) for classification. [6] showed that even the fully connected neural network with sufficient layers can achieve high accuracy in handwritten digits classification. [7] proposed a CNN-based neural network architecture which scaled very well for the image-based classification tasks. It was perhaps the first time that multi-layer CNN was applied on images with back-propagation which was later known as LeNet.

We have also implemented LeNet architecture for accomplishing the task of handwritten digit classification. However, we changed the training paradigm to make our model more robust and capable of handling real-life scenarios in a more efficient manner.

III. MNIST Dataset

Modified MNIST (National Institute of Standards and Technology) is a massive collection of computer vision datasets that is widely used for training and testing various systems. It was built using two NIST datasets containing binary pictures of handwritten numbers. It's a subset of the NIST's wider collection. In a fixed-size picture, the digits have been

size-normalized and centered. The training dataset includes handwritten numbers from 250 persons, with 50 percent of the training dataset coming from Census Bureau employees and the remaining from high school students. However, it is frequently credited as one of the first datasets among others to demonstrate the efficiency of neural networks.

The database contains 60,000 images for training and 10,000 images for testing, with just a portion of them suitable for cross-validation. The digits are all gray scale and are all the same size, with the intensity in the middle of the image at 28x28 pixels. Because all of the images are 28x28 pixels wide, the array may be flattened into a 28x28=784 dimensional vector. Each component of the vector is a binary value that represents the pixel's intensity. Chris Burges and Corinna Cortes chose and experimented with the digit images in the MNIST collection using bounding-box normalization and centering. The version supplied on this page by Yann LeCun employs centering by the center of mass within a larger window. Figure 1 depicts some of the sample handwritten digits from MNIST dataset.

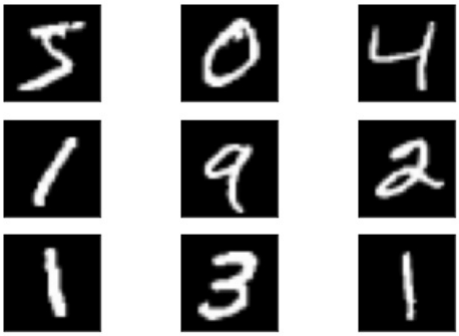


Fig. 1. Sample Digits

IV. Model Architecture

Although the MNIST dataset has been effectively solved, it might be a good way to start when building and practicing a methodology for tackling image classification tasks using convolutional neural networks. Building a model is crucial because it requires both constructing the infrastructure for the test harness so that any model we create can be assessed on the dataset and establishing a baseline in model performance on the problem against which any advances can be measured. The images in datasets only contain a hand-drawn digit with the same square size of 28×28 pixels.

A basic convolutional neural network with two convolution layers will be defined. We activate it with ReLU, then apply a max-pooling layer with a kernel size of 2 and a stride of 2 to it. The feature maps are downscaled to a dimension of 28 x 28 x 20 pixels. Each convolution layer with kernel size 5x5 is followed by a ReLU activation function and a max-pool layer, after which we have a Flatten layer leading to a dense layer with ReLU activation and another dense layer with a softmax activation function. Non-linearity is introduced via ReLU, and noise is reduced through max-pooling. Figure 2 shows the implemented architecture for handwritten digit classification task.

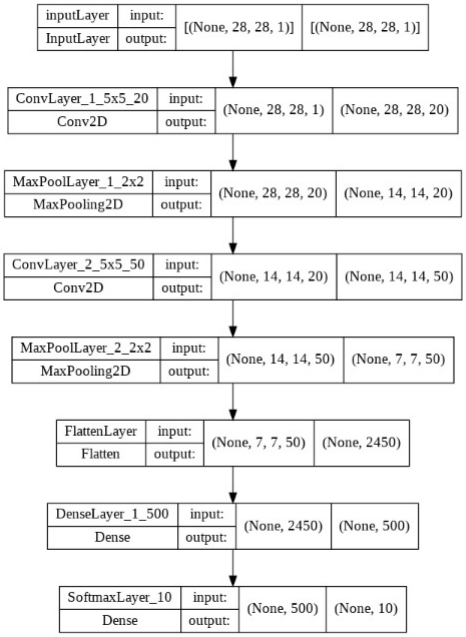


Fig. 2. Model Architecture

V. Support Functions and Model Evaluation

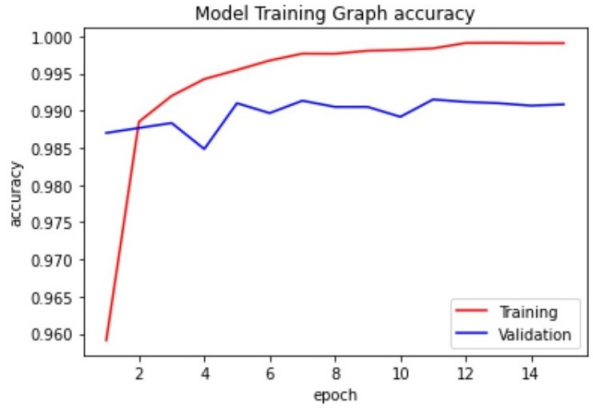
Due to multi-class classification we choose cross-entropy as our loss function, which combines a softmax function LogSoftmax() along with NLLLoss() loss function. As an optimizer, we have used RMSprop with a learning rate of 0.001 to optimize the model parameters. For each epoch, we use an enumerate function over the data loader to loop through batches of the training datasets and input the batch of image tensors into the model, which returns a tensor with predictions for that batch.

For validation we only used the forward propagation steps and compared the model's prediction to the actual labels for estimating the model's accuracy. The validation loss for the epoch is calculated by summing all of the losses for each batch iteration. As can be observed in figure 3, the epoch following 14 has the lowest validation loss of 0.005 and the highest accuracy of 0.995, after which the model begins to overfit and the validation loss starts to rise.

VI. Results and Discussion

The results of our proposed model trained and evaluated on MNIST dataset are analyzed in figure 4, figure 5 and figure 6. In figure 4, it is shown that an accuracy of 99 percent is achieved on MNIST dataset without any data augmentation.

But this technique is very fragile to even the smallest change in the data, e.g. resize, rotation and other transformations etc. When we slightly modified our test set, the classification



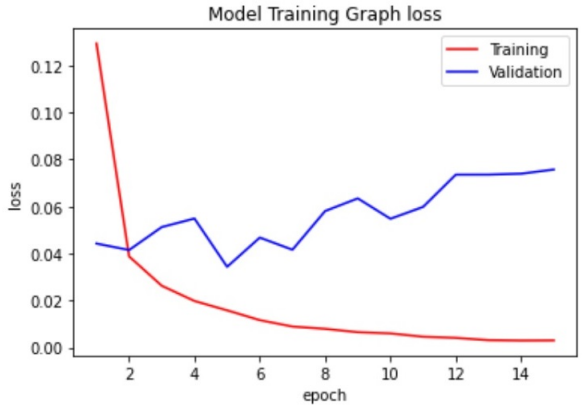


Fig. 3. Training Graph of Model

Accuracy of the trained model is 0.9921

	precision	recall	f1-score	support
0	0.99	1.00	1.00	980
1	0.99	1.00	1.00	1135
2	0.99	0.99	0.99	1032
3	0.99	0.99	0.99	1010
4	0.99	0.99	0.99	982
5	0.99	0.99	0.99	892
6	1.00	0.99	0.99	958
7	1.00	0.99	0.99	1028
8	0.99	0.99	0.99	974
9	0.98	0.99	0.99	1009
accuracy			0.99	10000
macro avg	0.99	0.99	0.99	10000
weighted avg	0.99	0.99	0.99	10000

Fig. 4. Classification Report (test data not modified)

Accuracy of the trained model is 0.2776

	precision	recall	f1-score	support
0	0.34	0.18	0.24	980
1	0.37	0.26	0.30	1135
2	0.21	0.35	0.26	1032
3	0.42	0.31	0.36	1010
4	0.30	0.28	0.29	982
5	0.27	0.35	0.30	892
6	0.22	0.32	0.26	958
7	0.24	0.38	0.29	1028
8	0.30	0.18	0.22	974
9	0.31	0.18	0.23	1009
accuracy			0.28	10000
macro avg	0.30	0.28	0.28	10000
weighted avg	0.30	0.28	0.28	10000

Fig. 5. Classification Report (test data modified)

Accuracy of the trained model is 0.937

	precision	recall	f1-score	support
0	0.95	0.95	0.95	980
1	0.95	0.99	0.97	1135
2	0.94	0.92	0.93	1032
3	0.96	0.92	0.94	1010
4	0.93	0.95	0.94	982
5	0.90	0.94	0.92	892
6	0.91	0.96	0.93	958
7	0.93	0.91	0.92	1028
8	0.97	0.91	0.94	974
9	0.93	0.91	0.92	1009
accuracy			0.94	10000
macro avg	0.94	0.94	0.94	10000
weighted avg	0.94	0.94	0.94	10000

Fig. 6. Classification Report (training data augmented)

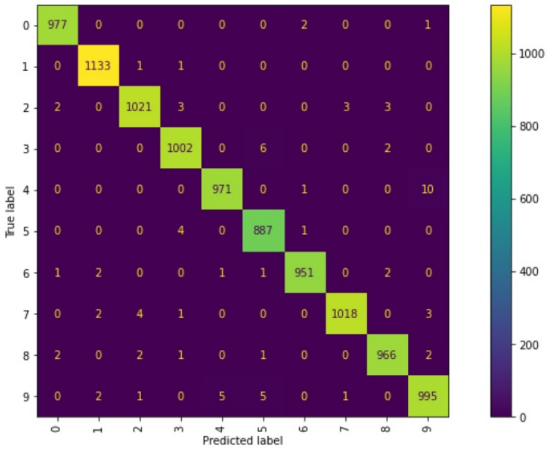


Fig. 7. Confusion Matrix (test data not modified)

accuracy dropped drastically and reduced to 27 percent as shown in figure 5.

If we augment MINIST dataset before training, then we can clearly see our network is now able to classify modified data with high accuracy. As shown in figure 6, we achieved an accuracy of 93.47 percent on modified data using this technique.

Figure 7, figure 8 and figure 9 show the confusion matrices of the results. Confusion matrix summarizes the prediction results and are helpful to identify which particular class is not being classified correctly.

VII. Conclusion

With the aim of improving the performance of handwritten digit recognition, we evaluated convolutional neural network based architecture . In contrast to fully connected networks, CNN exhibits a sparsely connected network with shared parameters which not only reduces the model complexity but also enhances the model's capability to extract spatially related features. By using an MNIST dataset, it can be observed that the traditional training approach works well when the test dataset is not exposed to simple transformations. However,

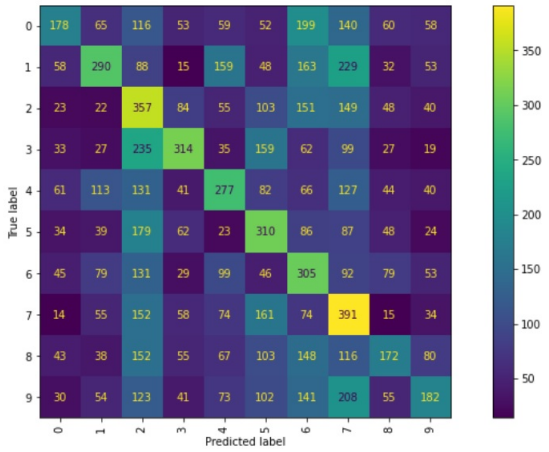


Fig. 8. Confusion Matrix (test data modified)

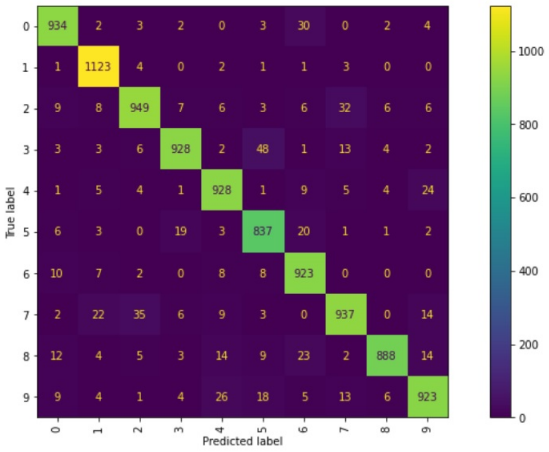


Fig. 9. Confusion Matrix (training data augmented)

with the slight modification of the test dataset, the accuracy of the model dropped and thus miss-classifications exploded. In order to enhance the robustness of our model, we have augmented the training dataset with transformations (re-scale, rotation, shift, brightness change) that can be commonly observed in real-life scenarios. We can see our classification accuracy improved and our model can recognize transformed data easily and accurately with this approach.

VIII. How to run the code

The code for this classification task has been written in python using Google Colab environment. It can be accessed from this link. The notebook has been commented on for easy understanding and running the code.

References

[1] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[2] H. Bay, T. Tuytelaars, and L. V. Gool, "Surf: Speeded up robust features," in *European conference on computer vision*. Springer, 2006, pp. 404–417.

[3] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, vol. 1. Ieee, 2005, pp. 886–893.

[4] L. Heutte, T. Paquet, J.-V. Moreau, Y. Lecourtier, and C. Olivier, "A structural/statistical feature based vector for handwritten character recognition," *Pattern recognition letters*, vol. 19, no. 7, pp. 629–641, 1998.

[5] Y. Lee, "Handwritten digit recognition using k nearest-neighbor, radial-basis function, and backpropagation neural networks," *Neural computation*, vol. 3, no. 3, pp. 440–449, 1991.

[6] G. Martin and J. Pittman, "Recognizing hand-printed letters and digits," *Advances in neural information processing systems*, vol. 2, 1989.

[7] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel, "Handwritten digit recognition with a back-propagation network," *Advances in neural information processing systems*, vol. 2, 1989.