MNIST Handwritten Digits Classification Using Deep Neural Networks

Son Tung Do
Department of Computer and Information Science
Fordham University
New York, NY
sdo3@fordham.edu

***Abstract*— The field of computer vision attempts to teach computer models to behave like the human visual system and gain a high-level understanding of visual inputs such as images and videos. One application of computer vision is handwriting recognition, where a computer attempts to read and interpret images of human handwritten text. This project aims to make a handwriting recognition model that can classify handwritten numbers 0-9. The dataset is taken from the MNSIT database of handwritten digits, and the model will be built using TensorFlow on Google Colab. Results show convolutional neural nets outperforming multi-layer neural nets, with the most complex model achieving 99.54% test accuracy by training only on raw data.**

Keywords—mnist, machine learning, deep learning, neural network.

I. Introduction

For thousands of years, humans have invented and used handwriting as a way to transmit and store information. Now, decades after the invention of the computer, we have multiple devices to quickly transmit and store data, but handwriting still remains a familiar way to deliver a message. What separates handwriting from typed characters on a computer is the unique individual elements the handwritten characters have. People all write in their own styles, while characters on a computer are just standardized strings of bits. This difference makes characters on a computer easily processed by a computer itself, as it can quickly read through the bit values to find the value of a character.

However, handwritten characters are hard to read for a computer. Handwritten characters only come in the form of images instead of a few standardized bytes, and those images come in all shapes and forms. There is no algorithm that can reliably classify handwritings due to how much people's handwritings vary from each other. Instead of writing algorithms to classify handwritings, we can build a machine learning model and train it to do this classification task.

This project will attempt to build a neural network model that can classify handwritten digits. These are Arabic numerals consisting of 10 digits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. A model that can classify these digits can be applied in many places around the world, as they are used in a wide variety of languages and regions with different alphabets.

II. Experiment Methodology

A. Data Set

MNIST data set is a modified version of a handwritings database of the National Institute of Standards and Technology (NIST). It contains 60,000 examples in the training set and 10,000 examples in the test set. The MNIST dataset takes these samples from 2 different NIST datasets. One contains more readable handwriting of Census Bureau employees, while the other contains less readable samples from high school students.

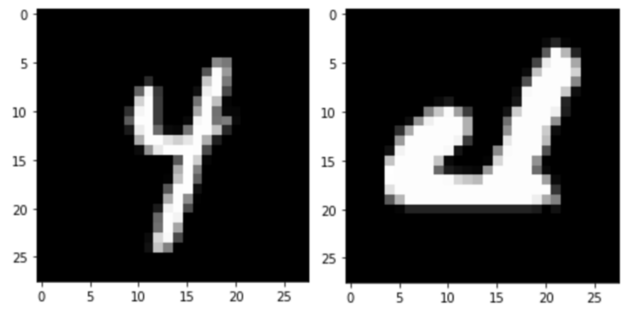

Figure 1. Samples digits, one more readable and one less readable

The images have been normalized and fit into the center of a 28x28 grid. As introduced in Yann LeCun's website [1], MNIST is a simple dataset of

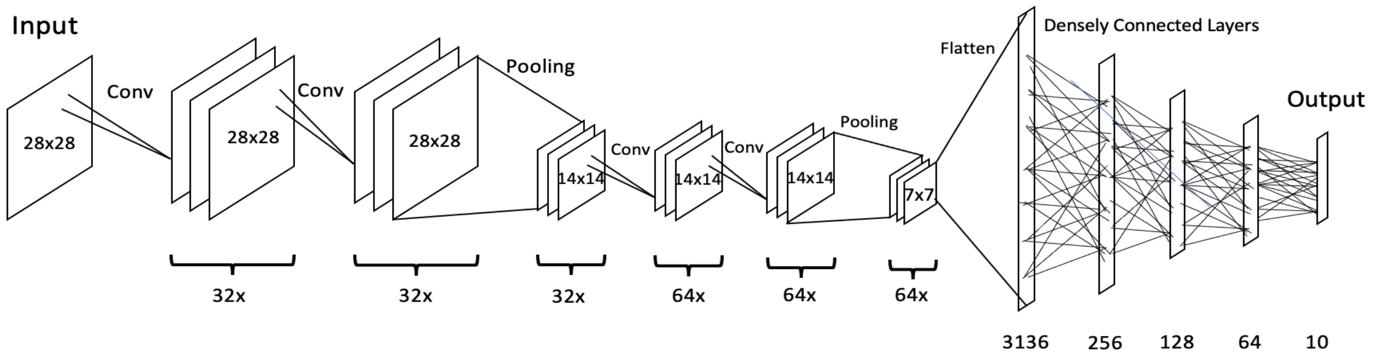

Figure 2. Architecture of the best performing network

real-world data that does not require too much preprocessing and formatting. In this project, the data does not get processed significantly. In some parts of the program, these images got added dimensions to fit in with the input requirement of the model.

B. *Algorithms*

This project uses Deep Neural Networks and Convolutional Neural Networks to classify handwritten digits. These networks are trained using backpropagation with the optimizer algorithm Adam. Adam is an optimization algorithm that can adapt and scale the learning rates of the network [2]. The loss function used for training is cross-entropy.

All neural networks built in this project are feed-forward. All their neurons use the activation function ReLU (Rectified Linear Unit), with ReLU being defined as $f(x) = \max(0, x)$. Deep neural networks will use densely connected layers, with each neuron of a layer being fully connected to every neuron of the previous and next layers. Convolutional networks will also utilize convolutional layers, which are filters that can learn to detect features of an image such as edges or circles. These features are contained in multiple feature maps, one for each filter.

As convolutional layers create multiple feature maps, they drastically increase the number of units in the network. To simplify the model and decrease training time, we downsample the feature maps using max-pooling layers. Using max-pooling layers of size 2x2 can decrease the number of pixels in feature maps by 4 times.

Additionally, densely connected layers can apply dropout to reduce overfitting. Dropout is a technique that can prevent overfitting in a neural network by randomly dropping neurons along with their connections during training. According to Srivastava et al., this method prevents the neurons from co-adapting too much [3].

C. *Setup Methodology*

This project runs on Jupyter Notebook in Google Colab. The main library used to create neural networks are TensorFlow and Keras. An advantage of running TensorFlow and Keras on Google Colab is to utilize Google machines instead of local machines to train models. The platform also gives access to GPU for faster model training than CPU.

We train each network 5 epochs at a time on the training set and evaluate its performance on the test set. Networks are trained with mini batches of size 128. Some network architectures are trained both with and without dropout to compare results.

D. *Network Architecture*

Neural network architectures gradually increase in size and complexity through the project. The output layer is fixed to always have 10 units, corresponding to 10 different classifications of digits. The project starts with simple 2-layer neural networks with the hidden layer having 16, 64, and 128 units, respectively. After that, we try with 3-layer architectures 16-16-10, 64-16-10, 64-64-10, and a larger network of 1000-500-10 units. This network is then extended with one more layer to become 1500-1000-500-10.

Convolutional neural network models also start from simple and gradually expand. We first try with 1 convolutional layer of 32 filters, followed by a max-pooling layer of size 2x2 and some fully connected layers. The fully connected layers are 32-10, 100-50-

10, and 150-100-50-10. The convolutional layers are now expanded to contain 2 layers of 32 filters, each followed by a pooling layer, with 100-50-10 fully connected layers at the end.

We also test out a replica of Yann LeCun's LeNet 5 [4], using the same architecture (6-P-16-P-120-84-10) but different and more modern activation functions and optimization algorithms. Finally, we expand previous networks to make the final model, 32-32-P-64-64-P-256-128-64-10 (Figure 2). This network contains over 900,000 trainable parameters and takes 10 GPU seconds to train one epoch.

III. Results

Networks are evaluated by their accuracy or, equivalently, error rates. Results for each network in this section are recorded by their best performance on the test set, so the numbers of epochs trained for each network will vary.

Table 1. Performance of 2-layer neural networks

Network Architecture	Test Error Rate (%)
2-layer NN, 16 hidden units	5.01
2-layer NN, 16 HU (with dropout)	5.77
2-layer NN, 64 HU	2.39
2-layer NN, 64 HU (with dropout)	2.57
2-layer NN, 128 HU	2.18
2-layer NN, 128 HU (with dropout)	2.14

Results for 2-layer neural networks (Table 1) show minimal improvements by the dropout layers. Small networks do not show to be affected by dropout too much, and dropout layers might even hinder their training process. 3-layer neural networks will not use dropout layers (Table 2).

Table 2. Performance of 3-layer neural networks

Network Architecture	Test Error Rate (%)
3-layer NN, 16+16 hidden units	4.92
3-layer NN, 64+16 HU	2.61
3-layer NN, 64+64 HU	2.65
3-layer NN, 1000+500 HU	1.78

Larger neural networks start to show signs of overfitting only after 10-15 epochs of training. The 3-layer 1000+500 hidden unit network reaches 98.22% accuracy after 10 training epochs but drops to 97.93% accuracy after 15 training epochs. 4-layer neural networks utilize dropout layers at different rates to prevent overfitting. Results (Table 3) show the best performing architecture of a vanilla neural network: a 4-layer network with 1500+1000+500 hidden units and a dropout rate of 15%. This network is the only one up to this point that breaks the 98.50% accuracy barrier.

Table 3. Performance of 4-layer neural networks

Network Architecture	Dropout rate	Test Error Rate (%)
4-layer NN, 1500+1000+500 hidden units	0	1.71
	0.15	1.47
	0.2	1.57
	0.25	1.90
	0.5	1.89

To achieve better results, convolutional layers will be used to help the network recognize features of the input images. These networks utilize dropout layers to limit overfitting.

Table 4. Performance of convolutional neural nets with 1 convolutional layer

Network Architecture	Test Error Rate (%)
Conv net, 32-P-32-10	1.39
Conv net, 32-P-32-10 (with dropout)	1.46
Conv net, 32-P-100-50-10	1.04
Conv net, 32-P-100-50-10 (with dropout)	1.03
Conv net, 32-P-150-100-50-10	1.40
Conv net, 32-P-150-100-50-10 (with dropout)	1.28

Convolutional networks perform significantly better while using fewer units (Table 4). Even the smallest one with 1 convolutional layer, 1 pooling layer, and 1 hidden layer of 32 units achieves higher accuracy than the best performing 4-layer neural networks with 3000 hidden units. However, these networks still fall short of the 99% accuracy barrier, with the 32-P-100-50-10 model approaching really close at 98.96% without dropout and 98.97% with dropout.

Table 5. Performance of convolutional neural nets with 2 convolutional layers

Network Architecture	Test Error Rate (%)
Conv net, 32-P-32-P-100-50-10	0.77
Conv net, 32-P-32-P-100-50-10 (with dropout)	0.71
Conv net, 32-P-32-P-1000-500-10	0.92
Conv net, 32-P-32-P-1000-500-10 (with dropout)	0.95
LeNet 5, 6-P-16-P-120-84-10	1.10
LeNet 5, 6-P-16-P-120-84-10 (with dropout)	0.98

With another convolutional layer, these networks manage to break the 99% barrier, with the best performing model being the 32-P-32-P-100-50-10 using dropout (Table 5). A modified version of Yann LeCun's LeNet 5 is trained to reach 1.10% error rate without dropout and 0.98% error rate with dropout applied. This result is similar to what LeCun achieved with the original LeNet 5 in 1998 (0.95% error rate) [4].

Table 6. Performance of deep convolutional neural nets

Network Architecture	Test Error Rate (%)
Conv net, 32-32-P-64-64-P-256-128-64-10	0.68
Conv net, 32-32-P-64-64-P-256-128-64-10 (with dropout)	0.46

Finally, we build a network using double convolutional layers before each pooling layer, and the fully connected layers contain 3 hidden layers of 256+128+64 units. This network manages to break the 99.5% accuracy barrier when trained with dropout. It reaches 99.32% accuracy without dropout and 99.54% accuracy with dropout.

IV. RELATED WORK

The MNIST dataset is put together by Yann LeCun, Corinna Cortes, and Christopher Burges [1]. This dataset is a famous benchmark for testing machine learning algorithms in image recognition. The dataset's website also presents state-of-the-art results for different techniques. Many of these results were presented in 1998 article "Gradient-based learning applied to document recognition" by LeCun, Bottou, Bengio, and Haffner [4].

The paper "Dropout: A Simple Way to Prevent Neural Networks from Overfitting" by Srivastava, Hinton, Krizhevsky, Sutskever, and Salakhutdinov provide one way to regularize and improve the network's performance. Their technique is successful at preventing overfitting in neural networks, especially in large and deep networks that are prone to overfitting.

Michael Nielsen's book *Neural Networks and Deep Learning* helps provide the fundamentals of neural networks to begin this project [5]. Along with theoretical knowledge, the book also explores this MNIST dataset and some strategies to improve network performance.

V. CONCLUSION

Overall, convolutional neural networks significantly outperform vanilla neural networks in image recognition tasks. Our results show that the simplest convolutional neural network (Table 4) can outperform a much larger multi-layer vanilla neural network (Table 3).

This project manages to build a high-performance convolutional network (Figure 2) that approaches the state-of-the-art performance on this dataset. With a test error rate of only 0.46%, it outperforms every classifier listed on the MNIST website that does not utilize any method of data augmentation or expansion [1].

The network architecture in Figure 2 has achieved a very high accuracy of 99.54% by training only on the raw dataset of 50,000 images from the MNIST training set. This limits the model to fewer and less diverse training data than many models listed on the MNIST website. This limitation could open up opportunities to improve for future work by applying data processing techniques like validation set and data augmentation techniques like elastic distortion. It can also improve using an ensemble of multiple models.

References

[1] Y. LeCun, C. Cortes, C. Burges, "MNIST handwritten digit database." http://yann.lecun.com/exdb/mnist/ (accessed Dec. 01, 2021).

[2] V. Bushaev, "Adam — latest trends in deep learning optimization.," *Medium*, Oct. 24, 2018. https://towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization-6be9a291375c (accessed Dec. 05, 2021).

[3] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting,"

[4] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE*, 86(11):2278-2324, November 1998.

[5] M. A. Nielsen, *Neural Networks and Deep Learning*. Determination Press, 2015. Accessed: Dec. 06, 2021. [Online]. Available: http://neuralnetworksanddeeplearning.com