Available Online: 11 March 2026 DOI: 10.54254/2977-3903/2026.32222

Handwritten digit classification with convolutional neural network

William Jiang

The Village School Houston, Houston, USA

weilian8888@outlook.com

Abstract. In this project, we develop a Convolutional Neural Network (CNN) to classify handwritten digits from the MNIST dataset, a widely used benchmark in computer vision. Unlike traditional image-processing pipelines that rely on engineered features, CNNs automatically learn hierarchical representations directly from raw pixel data. Our model consists of two convolutional layers, max pooling, dropout for regularization, and two fully connected layers. Trained for five epochs using the Adadelta optimizer with learning rate decay, the network achieves a test accuracy of 98.92%. These results demonstrate that even a relatively small CNN can achieve strong performance on MNIST with minimal tuning.

Keywords: Convolutional Neural Network, MNIST, deep learning, image classification

1. Introduction

Deep learning has revolutionized how computer recognizes images and understand languages [1-6]. Compared to classical computer vision methods for a task such as image classification, deep learning methods such as Convolutional Neural Networks (CNN) [7] do not require hand-designed heuristics, can achieve a very high success rate, and most importantly, can easily generalize to new unseen images.

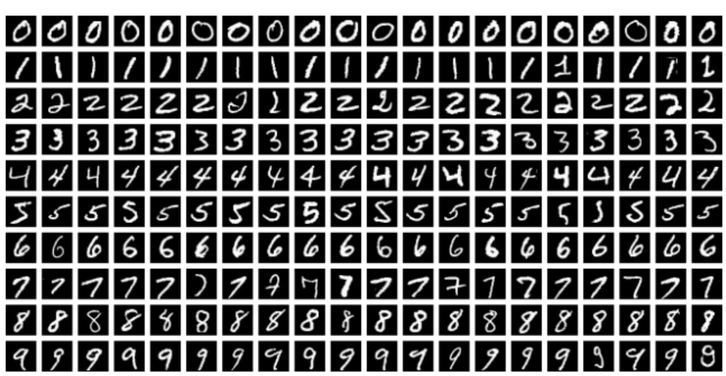

Figure 1. Samples from the MNIST dataset (source: Wikipedia)

With the progress made in the field of deep learning, there have recently been many new datasets emerging for benchmarking neural networks, such as the ImageNet [8] and the

MNIST [9] datasets. While ImageNet focuses on a larger dataset of RGB images, the MNIST dataset provides a simpler benchmark with images of handwritten digits from 0 to 9 (examples of these images are shown in Figure 1).

In this project, we choose the MNIST dataset as a test bench and build a CNN from scratch, using the PyTorch [10] library. PyTorch is one of the most popular packages for the Python language to code a neural network due to its readability and wide range of pre-built functions. We chose PyTorch over TensorFlow [11] for its dynamic computation graph. Compared to a Multilayer Perceptron (MLP), CNNs are better at capturing the spatial relationship among pixels, and the convolution behavior enables weight sharing such that it greatly reduces the size of the network and consequently, the training time. With our CNN, we can achieve 98.92% accuracy on the randomly selected held-out dataset.

2. Method

In this section, we describe how we build a CNN for handwritten digit classification using the MNIST dataset.

2.1. Network architecture

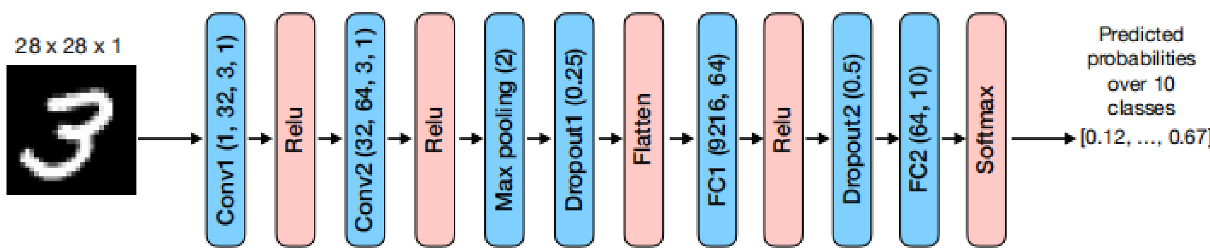

Figure 2. Our network architecture

For Conv layer, the first number is the number of input channels, the second number is the number of filters/kernels, the third number is the size of the square kernel, and the last number is the stride size. For Max pooling layer, the number is the size of the kernel. For FC layer, the first number is the number of input features, and the second number is the number of output features. For Dropout layer, the number is the probability of an element being zeroed.

In our network architecture (shown in Figure 2), we use a couple of convolutional layers to learn the spatial features of the MNIST images, and then we use pooling layers to reduce the dimension of the images. The original image is a $28 \times 28 \times 1$ gray scale image, and we first preprocess the images by normalizing them to mean 0 and standard deviation 1. This normalization step is a very important step for deep learning and ensures that all the pixel values are in a standardized range.

The preprocessed image is then passed through the first convolution layer Conv1 (1, 32, 3, 1). After this convolution layer, we apply a Relu layer that brings non-linearity to this network by activating values greater than zero. Then the data is passed through the second convolution layer Conv2 (32, 64, 3, 1), and followed by another Relu activation. After the convolution layers, we need to reduce the dimension of the images by using a max pooling layer of kernel size 2. To add regularization, we add a dropout layer that abandons 25% of the neurons randomly. Then the data is flattened into a long vector.

The principle that we are following is that we will reduce the width and length of the data while increasing the depth. We increase the depth by using a lot of different kernels, and in deep learning, the hope is that each kernel can learn some distinct features from the feature maps.

The flattened data is finally passed through a Multilayer Perceptron (MLP), which consists of two linear fully-connected layers with a Relu and a dropout layer in between. We then get a 10-dimensional logit vector for each sample, and we can pass it through a softmax layer to get the predicted probabilities over 10 classes.

2.2. Training recipe

We now describe how we train our network. We use 60,000 images for training, and 10,000 images for testing. We use the ADADELTA [12] optimizer with learning rate decay. We reduce the optimizer's learning rate by multiplying it by 0.7 at the end of every epoch. Starting from an initial rate of 1.0, the effective learning rate shrinks exponentially over time. Because the scheduler is stepped once per epoch, each new epoch begins with a smaller learning rate, helping the model transition from coarse, exploratory updates early in training to finer, more stable adjustments later on. We train our model for 5 epochs. This is a simple classification task, and we have a fairly capable network, so we don't need to train an extensive number of epochs. We will then save the last model as the final product. The training is conducted on a 2021 MacBook Pro with an Apple M1 Max chip and 64 GB unified memory.

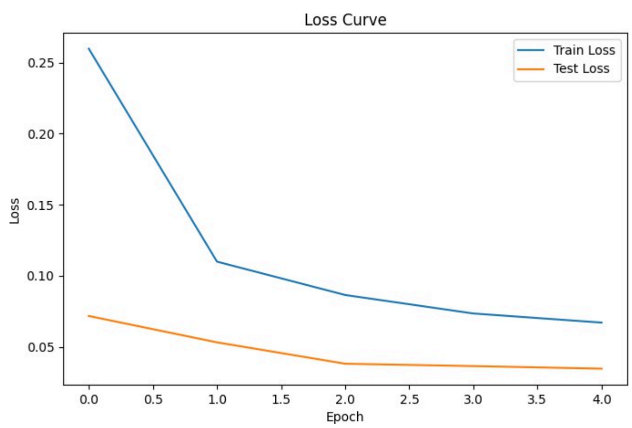

Figure 3. Learning plot for train and test losses

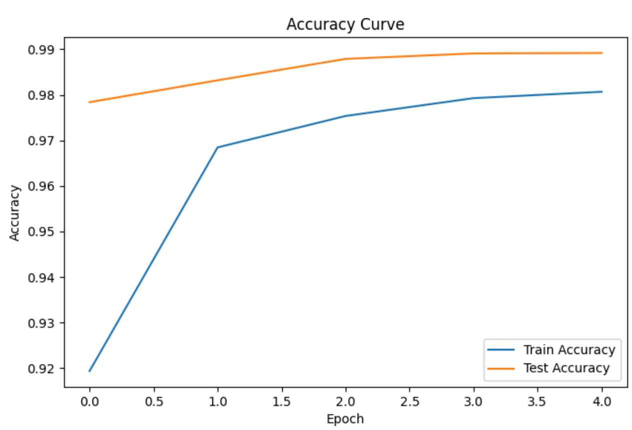

Figure 4. Learning plot for train and test accuracies

3. Experiments

We report the train and test losses in Figure 3, train and test accuracies in Figure 4. Both the train and test losses drop with the number of epochs, and both the train and test accuracies increase with the number of epochs. The performance converges at the end of training, and we don't see any sign of overfitting at the end. Our model reaches a success rate of 0.9892 on the test set.

We also load the saved model at the end of training and plot its prediction on 5 samples from the test set, as shown in Figure 5. Our model has a very high success rate, but it still makes mistakes from time to time. One such example is that it incorrectly predicts digit 6 as 0, which is an understandable mistake, since the two digits are very similar.

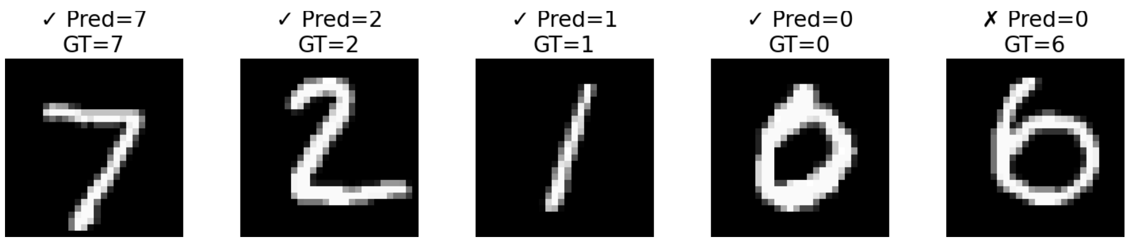

Figure 5. Some examples of our model predicting on the test set

The first 4 examples are correct predictions, the last one is incorrect.

4. Conclusion

In this work, we implemented and trained a convolutional neural network for handwritten digit classification on the MNIST dataset. The model converged quickly and achieved a high test accuracy of 98.92% without requiring extensive hyperparameter tuning or prolonged training. Our experiments confirm the effectiveness of CNNs in capturing spatial patterns in image data and demonstrate that a compact architecture is sufficient for MNIST-level tasks. While MNIST is a relatively simple dataset, the workflow developed here—data preprocessing, model design, learning-rate scheduling, and evaluation—provides a solid foundation for extending CNN-based approaches to more complex image classification problems.

References

[1] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 770–778). https: //doi.org/10.1109/CVPR.2016.90

[2] OpenAI. (2023). *ChatGPT: Chatbot based on GPT-4*. https: //chat.openai.com/

[3] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning.* MIT Press. https: //www.deeplearningbook.org/

[4] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). *Attention is all you need*. In I. Guyon, U. von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in neural information processing systems 30* (pp. 5998–6008). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[5] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). *Language models are unsupervised multitask learners.* https: //cdn.openai.com/better-language-

models/language_models_are_unsupervised_multitask_learners.pdf

[6] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems* (Vol. 25, pp. 1097–1105). https://papers.nips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html

[7] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation, 1*(4), 541–551. https://doi.org/10.1162/neco.1989.1.4.541

[8] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition* (pp. 248–255). IEEE. https: //doi.org/10.1109/CVPR.2009.5206848

[9] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE, 86*(11), 2278–2324. https: //doi.org/10.1109/5.726791

[10] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., & Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems, 32*. https://proceedings.neurips.cc/paper_files/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html

[11] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., & Zheng, X. (2016). *TensorFlow: A system for large-scale machine learning*. https: //www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi

[12] Zeiler, M. D. (2012). *ADADELTA: An adaptive learning rate method.* arXiv. https: //arxiv.org/abs/1212.5701