Handwritten Digit Recognition with Convolutional Neural Network

Gaganpreet Kaur
Computer Science Department
Hood College
Frederick, MD
gk7@hood.edu

Abstract— Image recognition requires complex and computationally intensive tasks. Many applications demand recognizing images for various purposes such as MRI and X-Ray scans. Images are inherently large; thus, it requires extra computing power to process. As imaging technology advances to capture more data in pixels, image sizes will continue to increase in size. The resulting image will make image processing more cumbersome to process. The image complexity magnifies when there are more patterns to understand and analyze. The goal of this project was to demonstrate deep learning skills. This project uses deep learning to recognize handwritten images from the MNIST (Modified National Institute of Standards and Technology) dataset [2]. Deep learning can help process images efficiently by reducing their complexity and computation time. The problem of recognizing the handwritten digits was processed by the CNN (Convolutional Neural Network) algorithm. The CNN algorithm reached about 99.45% accuracy with a test loss of 1.8%. This paper discusses the solution approach and implementation process to achieve better accuracy.

Index Terms— Algorithms, Computer Vision, Convolutional Neural Network, Deep Learning, Deep Neural Network, Feed Forward Neural Network, Imaging Process, Image Augmentation, Max Pooling, MNIST, Normalization

1 Introduction

Since the past decade, Deep Learning has made significant contributions to many areas. It is part of a machine learning family that helps solve problems dealing with computer vision, image recognition, speech recognition, and natural language processing. Deep learning algorithms generate multiple layers from input layers, explore hidden layers, and produce output layers for evaluation. Fig. 1 portrays a feed-forward deep neural network that interprets images of hand-written digits and classifies them as one of the 10 possible numerals [3]. This algorithm uses the activation function to determine the output layers. This concept of processing information mimics the human brain to find the information and draw conclusions. According to one article, neural networks imitate the inner workings of the human brain to process data, create patterns, and inform decision-making [4]. These techniques help extract key information from large datasets in a highly efficient and accurate way.

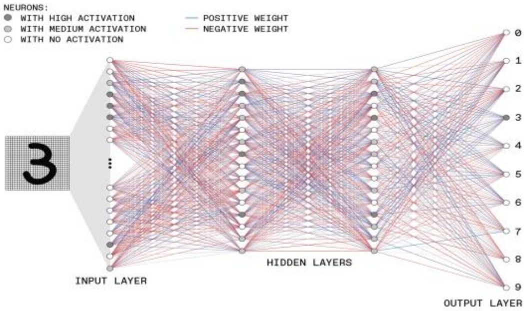


Fig. 1. Feed-Forward Deep neural network
Source: Adapted from [3]

In this project, the handwritten digits were processed by using the CNN (Convolutional Neural Network) algorithm. CNN is one of the most popular deep neural network algorithms. It is commonly used to analyze visual images by reducing the images into a form that is easier to process, without losing its features to make better predictions. CNN works by taking image data as input, applying convolution processes, generating a fully connected network, and producing an output. The illustration of how CNN works is provided in Fig. 2. This figure shows a multilayer CNN workflow which reflects the model used in this project. In this process, an image with a dimension of 28 x 28 x 1 is transferred into a fully connected network of neurons where every neuron in a layer is connected to all the neurons in other layers.

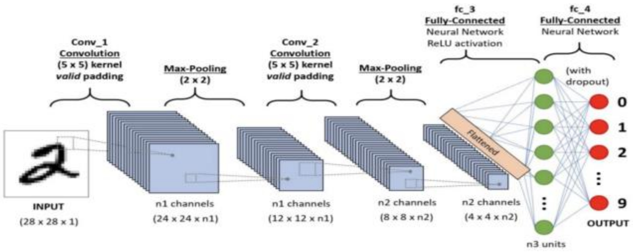


Fig. 2. Convolutional Neural Network for recognizing handwritten digits.
Source: Adapted from [5]

2 MNIST Dataset

The standard MNIST dataset was used and tested on the Kaggle platform using Python libraries. It is the standard dataset used by data scientists to practice their skills. To prepare the data, it was split into two tuples: test, and train tuple. Each tuple contains labels and dataset elements. The training dataset contained sixty thousand images and the test dataset contained ten thousand images. The handwritten images consist of numbers from 0 to 9 along with multiple variations of each dataset. So, there are 10 classes of a dataset in this project. Each image shape in a dataset has a 2D dimension of 28x28 pixels. Also, each pixel value in an image is an unsigned integer in the range between 0 and 255. The value 0 represents black and 255 represents white.

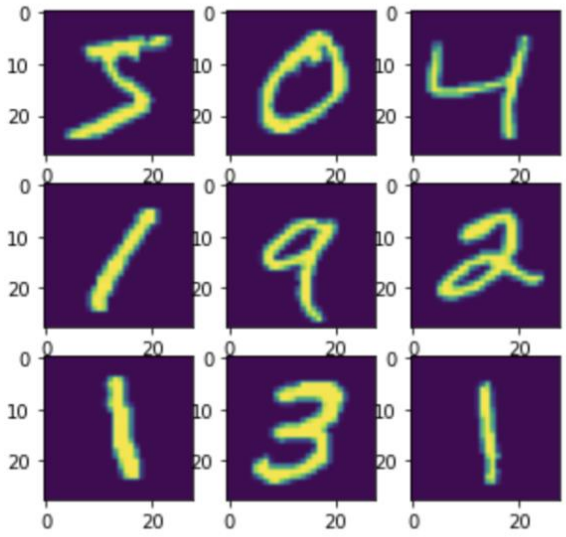
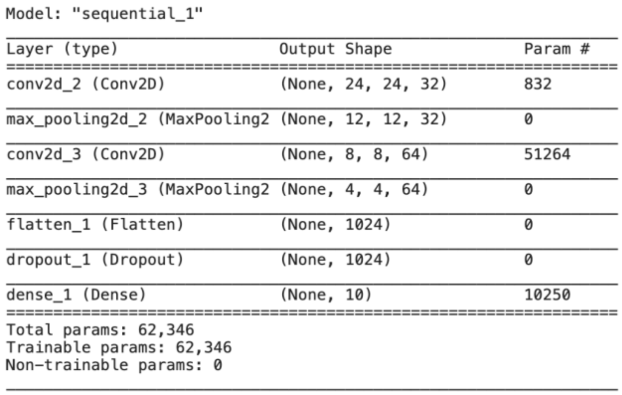


Fig. 3. Sample images from train dataset.

3 Implementation Process

Although this data has been solved previously, this project uses the CNN algorithm to achieve an accuracy of 99.45% percent which is a nearly optimal solution. A new model was developed to predict handwritten digits with high accuracy. There are multiple steps taken to achieve better accuracy. The first step in the implementation process is to load the data and split it into test and train images along with their labels. The second step is to prepare the images by scaling down the image pixels and encoding labels into binary matrices. The image pixels were normalized by converting its data type from unsigned integer to float and then dividing the resulting value by 255. This step helps reduce the problem size to work with. The test and train were also expanded from a two-dimensional matrix to a three-dimensional matrix. The next task was to build a CNN model or network consisting of multiple layers. This is a crucial step in the process as it will impact the performance of the model. In this step, multiple layers were added to transform the three-dimensional image matrix into a fully connected neural network. Fig. 1 represents the workflow of this network model prepared for this assignment. Another visual of the model can be seen in Fig. 4.



```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_2 (Conv2D)            (None, 24, 24, 32)        832
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 12, 12, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 8, 8, 64)          51264
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 4, 4, 64)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 1024)              0
_________________________________________________________________
dropout_1 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                10250
=================================================================
Total params: 62,346
Trainable params: 62,346
Non-trainable params: 0
_________________________________________________________________
```

Fig. 4: Model summary shows multiple layers with layer type and output shape.

The above model summary generated from this setup shows that each Conv2D and Max_Pooling2D layer outputs 3D tensors of a shape (height, width, channels). As the CNN algorithm goes deeper in the network, the height and width dimensions tend to shrink. The first argument of Conv2D controls the number of channels of each layer such as 32 and 64. In this model, the number of channels was doubled from 32 to 64 in the second Conv2D layer because the dimension size is reduced, allowing more computations to occur. The flattening layer flattens the input of 3D tensors of shape into 1D output. The dropout layer helps prevent overfitting during the training time. In the last, the dense layer was added to perform image classification.

After the network model was developed, the compilation was done using the optimizer method, a loss function, and an accuracy matrix for evaluation. Next, the model was trained using a fit method with epochs value of 20 and batch size of 128. Also, the fit method was set up to stop the training early if a monitored metric stopped improving.

4 Results and Evaluations

The model developed for recognizing handwritten digits reached an accuracy of 99.445 percent with a test loss value of .0176. The accuracy value can be observed from Fig. 5. It is an optimal solution and it reached above 99 percent. It means that this model has correctly identified 9,945 out of 10,000 test images. Only a small portion of images was not identified correctly. The original version of the model achieved an accuracy of 94 percent. However, after tweaking with model layers and epoch size, the accuracy was improved. It was a dramatic improvement in terms of accuracy.

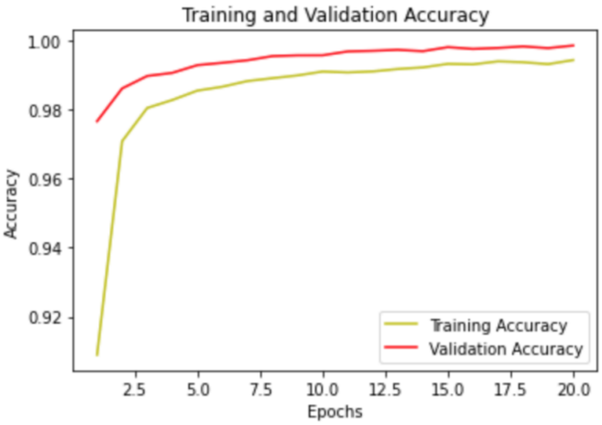


Fig. 5: Training and Validation Accuracy Graph

5 Performance Considerations

Although the results from this project were near perfect, it is important to know strategies to improve model performance. CNN algorithm largely depends on a high volume of train datasets. It means that the more training dataset it has, the better accuracy it will provide. The other way performance can be improved is to fine-tune the parameters of the model. One can learn and try different filters, kernel size, activation method, epochs, and optimizers to reach better performance. For example, the performance was improved when the epoch size was changed from 11 to 20. As mentioned earlier, deep learning is effective when it has a larger dataset. It may not always be the case. To overcome that, one should use image augmentation to increase the data sample count such as zoom, shear, rotation, preprocessing function, etc. Another way to improve performance is by managing the overfitting and underfitting of models. Overfitting refers to a model that models the train data too well. In overfitting, the model produces good accuracy on the trained dataset. However, it does not provide good accuracy on test data. In other words, the model does not generalize well from the training data. Underfitting refers to a model which does not work well on both test and train data. Hence, it is crucial to know the right balance between overfitting and underfitting. It is encouraged to use more training data, leverage early stopping, and adjust epochs' value to avoid this problem.

6 Conclusion

Deep learning is a powerful tool for analyzing large volumes of data and making informed decisions. In this project, the use of the CNN algorithm showed excellent outcomes. The CNN algorithm can also help solve real-world problems such as analyzing MRI images, finding disease patterns, and identifying earthquakes. This project is a testament to proving the application of deep learning.

References

[1] S. Karanam, Y. Srinivas, en M. Krishna, “Study on image processing using deep learning techniques”, *Materials Today: Proceedings*, 10 2020.

[2] “Digit recognizer,” *Kaggle*. [Online]. Available: https://www.kaggle.com/c/digit-recognizer. [Accessed: 15-Dec-2021].

[3] S. K. Moore, D. Schneider, and E. Strickland, “How deep learning works,” *IEEE Spectrum*, 29-Sep-2021. [Online]. Available: https://spectrum.ieee.org/what-is-deep-learning. [Accessed: 15-Dec-2021].

[4] E. David, “Council post: How the future of deep learning could resemble the human brain,” *Forbes*, 10-Nov-2020. [Online]. Available: https://www.forbes.com/sites/forbestechcouncil/2020/11/11/how-the-future-of-deep-learning-could-resemble-the-human-brain/?sh=1580e9e2415c. [Accessed: 11-Dec-2021].

[5] S. Saha, “A comprehensive guide to Convolutional Neural Networks- the eli5 way,” *Medium*, 17-Dec-2018. [Online]. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53. [Accessed: 11-Dec-2021].