Handwritten digital recognition system

Brian Cheng

11/29/2022

Abstract

Building a handwritten digital recognition system with machine learning methods. Start by building a neural network model and try to modify the model to reach a better score then test it with my own handwritten image. Then build a CNN model also modify it to reach better result and test it with my own image. Last compare the difference between the two models.

Introduction

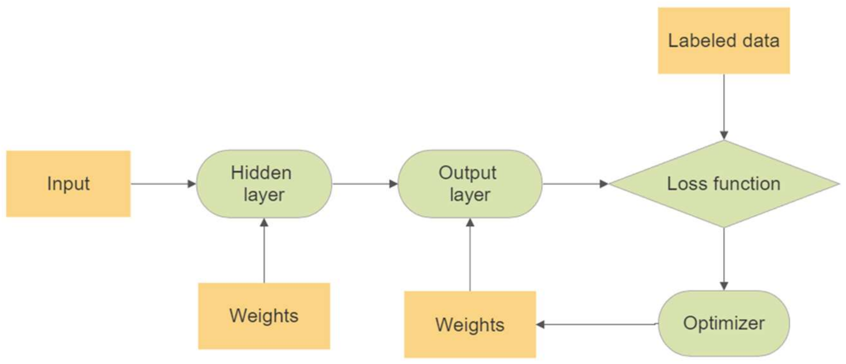
Handwritten digital recognition system function by reading in an image of handwritten digital letter and process it to recognize what digital letter is on the image. First after computer read in the image it will first grayscale the image process it into a square matrix, every matrix would contain a number which represent the darkness of the block. And using machine learning methods we will be able to process these data to train a model that will recognize the digital letter.

Some research about Neural Network

The artificial neural network is inspired by the study of neurons in biology. The nervous system is composed of multiple neurons, which transmit signals to each other through synapses.

In order to simulate the behavior of nerve cells, each neuron is set as an excitation function, which is actually a formula. After inputting an input value (input) to a neuron, the operation of the excitation function and the output value (Y) are output. This output value passed to the next neuron.

The Input Layer is the neuron that receives the signal, the Hidden Layer is the hidden layer, and the Output Layer is the output layer that responds, and the power transmitted by each neuron is called the weight (Weight, represented by W), which is the model requirement If the parameters of the solution are calculated, we will get a formula. As long as the input signal is passed through layers of conduction, the result can be inferred.



The use of the activation function is to increase the nonlinearity of the neural network model so the output will no longer be a linear combination of the inputs. It decides whether a neuron should be activated or not.

The optimizer is to guide the parameters of the loss function (objective function) to update the appropriate size in the right direction during the deep learning backpropagation process, so that the updated parameters make the loss function (objective function) value continuously approach the global minimum

A loss function is a way of evaluating "how well your algorithm/model predicts your dataset". If your prediction is completely wrong, your loss function will output a higher number. If the estimate is good, it will output a lower number. When tuning an algorithm to try to improve the model, the loss function will reflect whether the model is improving. "Loss" helps us understand the difference between the predicted value and the actual value

When a complete data set is passed through the neural network once and returned once, this process is called an epoch. When the data cannot be passed through the neural network at one time, it is necessary to divide the data set into several batches

Metrics is also used to calculate the loss of the current model (how wrong), but it is used to tell the AI engineer the current state of the model. The loss of the loss function is basically a direct counterpart to the optimizer. Provides cues for the optimizer to minimize the loss.

Procedure of building neural network model:

Data preparation:

1. First read in the training data. In this dataset, there are 60,000 data, and each data is a 28 * 28-pixel matrix graphic.
2. Encode the label Y value into one hot coding to convert them into the machine-readable form.
3. Flatten the 28*28 image to 784*1 array.
4. Normalize the pixel value into grayscale by dividing 255 (the pixel value is between 0~255).

Building models:

1. Add an input layer of 784 nodes (same number as inputs) and using relu function connecting to the hidden layer.
2. Add a hidden layer with 64 nodes (get by a formula from Stack Overflow Figure 1) and connect to output layer with SoftMax function.
3. Add an output layer with 10 nodes (the number of total digits).
4. Compile the model using cross entropy loss function, Adam optimizer and accuracy as metrics.

Training model:

1. Splitting the train data into 8:2 the 8 will be use for actual training while the 2 will be use as a validation.
2. Setting the epochs to 20 and batch size to 64.

Testing model:

1. Write a plot function to visualize the accuracy and loss function over the Epoch.
2. Evaluate the test data to see accuracy.

Optimize the model:

1. Checking the result of test data and see if there is overfitting or under fitting.
2. Redesign the model making it more complex by adding more layers or adding neuro numbers.
3. Modify the batch size.

Model prediction:

1. Entering 10 digits image of my own hand write and see if the prediction is correct.

The report of implementation of neural network model codes:

At start of the project, I first tried to build a neural network with Keras neural network model library example with base minimum of modification to the model and letting most of my result to be preset, I get a result of around 70% accuracy of the model. The result of the model is underfitting, so I take some research on some previous project and start to modify the number of neuros in the hidden layer until it reaches a point that the neuro does not increase the training accuracy to around 90%. Then I look into the model training settings and play around the split percentage epochs and batch size and eventually reach accuracy around 97%. And at this point I found it hard to improve the model, I reach a point that more modification will only cause overfitting. I feel like have reached limit of my implement model, so I stop and try to predict with this model.

I first started to predict with image of digits that I write on a white paper, however the model can't recognize any of them. I think the issue is that the shade is bringing a lot of noise to the model causing the prediction can't run successfully, so I use the painter app from Windows to draw the digits. This time the model is able to recognize digit 0, 2, and 6 correctly, however the result is still far from what I expected.

After the model predict I notice that the data could also be affecting the prediction, so I decided to work on doing more process to the data. I did some research and found a process to data called data augmentation. However, I don't want to keep on the current model, I want to do a more complex model that is CNN (Convolutional neural network model).

Convolutional neural network model

CNN also imitates the cognitive method of the human brain. For example, when we recognize an image, we will first notice the brightly colored points, lines, and surfaces, and then form them into different shapes (eyes, noses, mouths...), this This process of abstraction is the way the CNN algorithm builds models. The Convolution Layer is converted from point comparison to local comparison. By analyzing and judging the features of each block, and gradually stacking the comprehensive comparison results, a better identification result can be obtained.

Convolutional is to take each point of the image as the center, and take the surrounding N x N grid points to form a surface (N is called Kernel Size, and the N x N matrix weight is called "convolution kernel"), and each grid is given a different weight. , calculate the weighted sum, as the output of this point, and then move to the next point and process it in the same way until the last point of the image, this is the convolutional layer of CNN

Data Augmentation is commonly used in image processing. When we are doing machine learning, we sometimes encounter the problem of data imbalance. When we have very few pictures of a certain type, or when the entire data is not enough, our neural network will find it difficult to learn. At this time, we will use data enhancement methods to increase the data which is data augmentation.

Procedure of building Convolutional neural network:

Data preparation:

1. Load data
2. Reshape the x data
3. Normalize x value and encode y label
4. Split train data and validation data to 9:1

Building models:

1. Add one Input layer
2. Add two Conv2D layers
3. Add one layer of MaxPool2D
4. Add one layer of Dropout
5. Add two Conv2D layers
6. Add one layer of MaxPool2D
7. Add one layer of Dropout
8. Add one layer of Flatten
9. Add one layer of Dense
10. Add one layer of Dropout
11. Add one Output layer

12. Compile the model using cross entropy loss function, Adam optimizer and accuracy as metrics.

Data Augmentation:

1. Use ImageDataGenerator() to add rotate, shift, flip, zoom, whitening, or normalizing to the data to reduce overfitting

Training model:

1. Use datagen.flow() to do data augmentation while training.
2. Setting the epochs to 13 and batch size to 128 verbose to 1.

Testing model:

1. Visualize the accuracy and loss function over the Epoch.
2. Evaluate the test data to see accuracy.

Optimize the model:

1. Checking the result of test data and see if there is overfitting or under fitting.
2. Modify the epoch and batch size.

Model prediction:

1. Entering 10 digits image of my own hand write and see if the prediction is correct.

The report of implementation of Convolutional neural network model codes:

For the CNN model the overall implementation is a lot alike to the neural network model. The main difference is on the model layer and data augmentation. The CNN model layer is special at going through the convolution process before the hidden layer and output layer. The convolution is done by adding Conv2D and MaxPool2D layer. Overall, this model was built by consulting a preexisting code from Kaggle in [1] so there is not a lot of optimize overtime. Before the data augmentation is added the accuracy of the model is around 99.3%, and after adding the data augmentation the model has reach the accuracy of around 99.45%. Then I try to prediction with the model.

I again first try the handwritten on white paper version but the shade is still providing too much noise so the model couldn't recognize any of the digits. However, when I try the computer drawing version it appears to recognize all the digits correctly.

Conclusion:

For this project I was able to overcome most of the issue in this project. The handwritten digit recognition system I had is able to read clear image of digits like the ones draw on computers. And to be able to recognize an actual handwritten digit there is two ways I've thought of could be possible. First one is to do more work on the preprocessing of the data to cancel out the noise letting the shaded part to be read into grayscale as value 0, so the model could focus on the actual written part. The second way I thought of is to train the model on image of actual handwritten digits, by doing so the shade issue should be consider into model. But it will leave a concern that if the image is taken under different condition the model made be harder to train.

Overall, The CNN model I have shown to be doing well in recognizing a computer draw single digit, and the Neural Network model I have seems to only be good at the dataset I trained it with. But The Neural Network model I made still has a lot potential because I was only using a single hidden layer, and I believe it could perform better if I had given it a more complex model.

Reference:

$$N_h = \frac{N_s}{(\alpha * (N_i + N_o))}$$

N_i = number of input neurons.
N_o = number of output neurons.
N_s = number of samples in training data set.
α = an arbitrary scaling factor usually 2-10.

Figure 1 from stack overflow

[1] https://www.kaggle.com/code/yassineghouzam/introduction-to-cnn-keras-0-997-top-6

[2] https://ithelp.ithome.com.tw/articles/10191725

[3] https://ithelp.ithome.com.tw/articles/10291101