Deep Network on Handwritten Digit Classification

Sarah Zhou
shi.q.zhou@mail.mcgill.ca
Ziqing Liang
ziqing.liang@mail.mcgill.ca
Zuchan Fang
zuchan.fang@mail.mcgill.ca

Abstract

Neural network models belonging to the category of deep learning have significant capabilities for image data recognition and classification. In this project, we investigated the performance of a Multi-layer Perceptron (MLP) on classifying images to the appropriate classes on the classic hand-written digit dataset: the MNIST dataset.(MNIST)(mni) Here, we investigated the model performance when using 0, 1 and 2 as the number of hidden layers in different model architectures. We found that by optimizing some of the hyperparameters of the MLP, such as increasing the hidden depth (number of layers), increasing the width of the hidden layer (number of hidden nodes), changing the activation function, and adding regularization terms, the test accuracy of the model can be improved. As expected, we found that the application of 2-hidden layers is mostly effective when Rectified Linear Unit (ReLU) is compensated as the activation function.

1 Introduction

The goal of this project is to take an input image (28x28 pixels) of a handwritten single digit (0–9) and classify the image as the appropriate digit. To tackle this problem, we first investigated the data normalization over the MNIST dataset to standardize the image data with a data range of 0-255 and convert it to floating-point data in the range (0-1). In terms of the classifer, we mainly implemented the Multi-layer Perceptron (MLP) from scratch. For MLP, we gained test prediction accuracy of 96.5% as our best attempt result when implementing with 2 hidden layers and ReLU as the activation function.

We found that the model accuracy was remarkably enhanced under the premise of optimizing some of the hyper-parameters. The results of experimenting with increasing the hidden depth (number of layers), increasing the width of the hidden layer (number of hidden nodes), changing the activation function, and adding regularization terms were discussed in the following study. Given the fact that the MLP model reached a decent test accuracy over the dataset, we have roughly reached the conclusion that deep networks model highly accurate for image recognition as a whole. In order to verify this conclusion in more depth, we made an additional attempt at the end of this study, using the Convolutional Neural Networks (CNN) for classification, and obtained a higher accuracy as expected. The general reason why CNN can achieve higher accuracy and the lack of MLP implementation will be discussed at the end of the study.

1.1 Related Work

Along the last decade, the advancement of Deep Learning Machines (DLMs) has permitted to improve the performance of traditional Shallow Learning Machines (SLMs) in the immense majority of application fields. A family of DLMs, Convolutional Neural Network(CNN), is very effective to deal with uni-dimensional or bi-dimensional signals because of their basic architecture. For example, CNN ensembles gave misclassification rates of 0.23% (Ju et al., 2017) with the traditional benchmark MNIST database for handwritten digit recognition, while a drop-connect CNN design offered an even better 0.21%. (Mobiny et al., 2019)
Aside from deep learning method, traditional machine learning techniques can also be useful when they apply to computer vision tasks. Simon Bernard(Bernard et al., 2007) had already experimented the MNIST dataset with Random Forest(Breiman, 2001)on digit classification. He finally earned an accuracy of around 93%, which was in fact a very good performance.

2 Dataset

The Modified National Institute of Standards and Technology database (MNIST) is a large database

of handwritten digits that is commonly used for training various image processing systems. In this study, the MNIST dataset is retrieved directly from Keras library in Python. The MNIST database contains 60,000 training images and 10,000 testing images.

The MNIST data set consists of images of hand written digits belongs to ten labeled classes from 0 to 9. Each image data is composed of 785 numbers, whose value is between 0 and 255. The first number labels the digit represented by the image data (ranging from integer 0-9), and the following 784 numbers can form a 28×28 matrix, and each value corresponds to the pixel value of the pixel at the position, thus forming a picture with a pixel of 28×28.

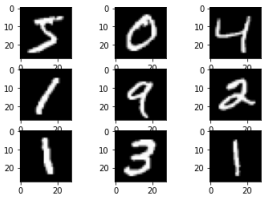

Figure 1: First 10 training images

In our study, our main goal is taking an input image (28x28 pixels) of a handwritten single digit (0–9) and classify the image as the appropriate digit.

A distribution of the class labels of the training dataset is shown in Figure 2.

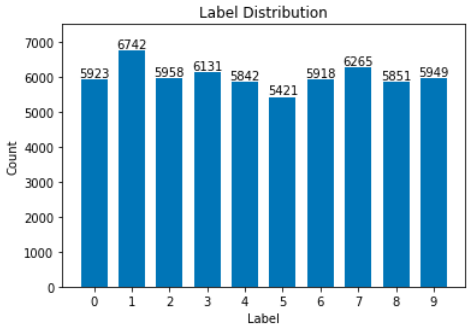

Figure 2: Label distribution of the training set

2.1 Preprocessing

In this study, the only preprocessing we made to the data set was to vectorize and standardize the pixel values. Knowing that the input value of the MLP needs to be a vector, we applied vectorization to the data set to make the input data better fit the model.

For uint-8 type image data, the data range is 0-255. Normalization can convert it to floating-point data in the range 0-1. One of the main reasons of doing so is because the floating-point type has high precision and is easy to process. Moreover, in machine learning, when the input data has multiple features, and the range of various features varies a lot. If one use the original data directly, the convergence speed of the gradient descent method will be slower. The appropriate approach is to normalize the data set to make it obey N(0,1), which follows the standard normal distribution.

3 Models

The MLP model implemented from scratch translates the theoretical knowledge given in the course COMP 551 Applied Machine Learning, Winter 2021. The implementation was largely inspired the implementation given in the article *Neural Network From Scratch with NumPy and MNIST* and follows the given examples(Hansen, 2020).

3.1 Multi-layer Perceptron (MLP)

Multilayer perceptrons learn in a supervised manner. The multilayer perceptron consists of a system of simple interconnected neurons(nodes), which is a model representing a nonlinear mapping between an input vector and an output vector.(Gardner and Dorling, 1998) Multilayer perceptron is generally composed of three parts: input layer, hidden layer and output layer (a sample of model with 2 layers can be shown in Figure 3). Both the input layer and the output layer are unique, and the number of hidden layers can vary. Among them, the input layer plays no computational role but merely serves to pass the input vector to the network. The nodes are fully connected by weights and output signals which are a function of the sum of the inputs to the node modified by a simple nonlinear transfer, or activation function. In the output layers of these models, the softmax activation function was used at the last. It is the superposition of many simple nonlinear transfer functions that enables the multilayer perceptron to approximate extremely non-linear functions. The output of the hidden layer and the output of the output layer can be calculated by the following formulas:

$$H = \sigma(XW_h + b_h) \tag{1}$$

$$O = HW_o + b_o \tag{2}$$

Where the H refers to the loss function, the σ refers to the activation function, X refers to the input vector.

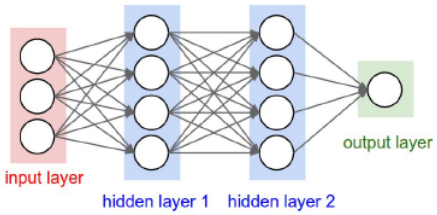

Figure 3: An MLP with 2 hidden layers each having 4 units

3.2 Forward Propagation

Forward-propagation refers to the sequence of transforming the neural network from the input layer to the output layer, calculating and storing the intermediate variables and output of the model in turn. At the end of this process, we get the loss function and bring it to the next steps.

3.3 Backward Propagation

Back-propagation refers to the method of calculating the gradient of neural network parameters. In general, back-propagation is based on the chain rule, along the order from the output layer to the input layer, and sequentially calculates and stores the objective function's intermediate variables and parameter gradients of each layer of the neural network. Using the loss function obtained from the forward propagation, the Gradient Descent algorithm is further applied to optimize the solution. In this study, we mainly worked with the Stochastic Gradient Descent (SGD) method. Each gradient descent will go through the back-propagation to update the parameter values in each layer to achieve the purpose of returning errors.

3.4 Activation Function

After a linear transformation is applied to the hidden layer, a nonlinear transformation is followed. This nonlinear transformation is called an activation function. In this study, we mainly investigated 3 activation functions: ReLU, Sigmoid and Tanh.

3.4.1 ReLU

The rectified linear activation function (ReLU) is a piece wise linear function that will output the input directly if it is positive, otherwise, it will output zero.

$$ReLU(x) = max(0, x) \tag{3}$$

3.4.2 Sigmoid

Sigmoid Activation function is non-linear. It is very simple which takes a real value as input and gives probability that 's always between 0 or 1.

$$Sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{4}$$

3.4.3 Tanh

Tanh help to solve non zero centered problem of Sigmoid function. Tanh squashes a real-valued number to the range [-1, 1].

$$Tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{5}$$

3.5 L2 Regularization

When the neural network has high variance, that is, overfitting, regularization (weight decay) is a very common method to help. L2 regularization is to add a regular term to the cross-entropy cost function

$$L2\ Regularization\ Cost = \frac{\lambda}{2m}||w||_2^2 \tag{6}$$

Where the L2 regular loss part is the sum of the squares of the Frobenius norm of the weight matrix of each layer in the neural network.

4 Results

4.1 The impact of hidden depth (number of layers

In this study, we investigated the MLP model performance when using 0, 1 and 2 as the number of hidden layers with ReLU activation. A detailed result could be found in Table 1. As expected, the model performance improved when the hidden depth was increased, and reached 96.5% of accuracy with 2 hidden layers. When there is no hidden layer, the classification ability of the model is basically equivalent to a logistic regression model. From the trend in Figure 4 below, it can be observed that the training accuracy and test accuracy increased greatly when the epoch is low at the beginning, and slowly increase as the epoch increases in the later stage. By comparing the result data in Table 1, we find that every time the hidden layers increases, the accuracy of the model as well as the

running time for each iteration will increase accordingly. This trend above aligns well with the content from class. In MLP, the input of the neurons in the next layer is the weighted sum of the output of the previous layer, and the features of the previous layer are abstracted in the next layer. The learning process is actually to adjust and optimize the weights of each connection. It can be seen that the level of abstraction of features that can be represented by a MLP with a shallow hidden depth is not high, and the deeper the level, the higher the level of abstraction of features, which results in better performances.

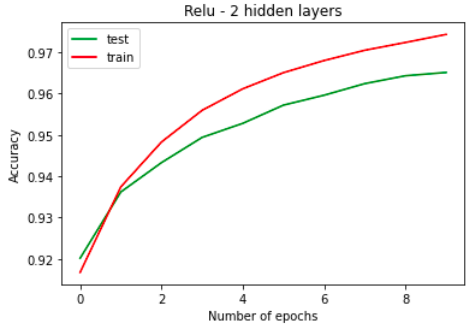

Figure 4: ReLU with 2 hidden layers

4.2 The impact of Activation Functions

Given the fact that the model with 2 hidden layers generally performed better, we then kept the number of the hidden layers as 2 and brought in the other 2 activation functions: Sigmoid and Tanh. By comparing the performance of the three models at epoch 10, we found that the model with ReLU activation still performed the best among the three (Table 1). The model with Tanh activation performed worse than the former but the result was somewhat similar, reaching an accuracy of 93.6%. However, the model with Sigmoid activation performed relatively worse, with an accuracy of 91.5%.

The above results are essentially in line with our expectations. Based on the theory, the gradient of the derivative of the Sigmoid and Tanh in the positive and negative saturation regions will be close to 0, which may cause the gradient vanishing problem. However, the result of the ReLU is the original value when it is greater than 0. This fact can alleviate the problem of gradient vanishing: since the derivatives are always 1, ReLU will not cause the gradient obtained by continuous multiplication to gradually vanished due to the small derivatives like Sigmoid and Tanh do. Therefore, we conclude that ReLU performs better than the other two activations in general.

4.3 The impact of Regularization (Weight Decay)

We observe significantly lower training and testing accuracies when applying the L2 regularization. This is expected as L2 regularization serves to prevent overfitting on the training data, which means that The optimal lambda parameter found was 0.001, which is also reasonable and falls into the expected range. (Brownlee, 2018)

Again, this finding partially aligns with our expectation. Due to the addition of the regularization term, the loss function derived from the forward propagation is modified. And this change must also modify to the gradient descent calculation in back-propagation. Due to the influence of the regularization part, no matter what the original weight updated after the gradient descent is, we try to make it smaller. In fact, it is equivalent to multiplying the matrix W by the weight of 1-λ/m times, and then subtracting it from the λ/m times the matrix W, so the weight is attenuated, thereby reducing the training accuracy.

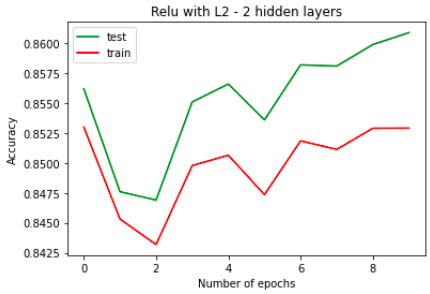

Figure 5: ReLU with L2 regularization

4.4 The impact of Normalization on the dataset

Running ReLU with 2 hidden layers of (128, 128) nodes on unnormalized data, we observed stagnant training and test accuracies around 10% (Table 1). This might be due to the dying of the gradients, since ReLU sets all gradients < 0 to 0. This is expected since although ReLU ensures the absence of vanishing gradients, when gradients all die to 0, it is unable to adjust the weights in the backward propagation step using SDG. Consequently, parameters are not adjusted, so the model stays stagnant and is unable to improve the accuracies.

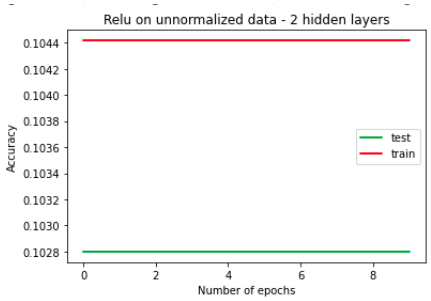

Figure 6: ReLU on unnormalized data)

5 Additional Experimental Attempts

5.1 Investigate different width of MLP

We investigated a range of alternative number of units for the MLP model with 2 hidden layers and ran experiments with RelU activation function.

The reported results in Table 1 shows a clear positive trend as the number of nodes within a hidden layer increases. Strictly speaking, the model with 512 units in both layers gives the highest accuracies. In reality, the accuracy always increases and then plateaus. This is because when the number of neurons within a hidden layer is small, some underfitting would occur. So, the model was not complicated enough to represent the entire MNIST dataset. As the number of neurons increases, the accuracy increases until it reaches a point of overfitting, which finally results in the plateau observed in the accuracy trend.

5.2 Convolutional Neural Networks

Up till now, we have roughly reached the conclusion that deep networks model highly accurate for image recognition as a whole. In order to verify this conclusion in more depth, we also experimented with another deep learning method, Convolutional Neural Networks (CNN) base on the inspiration of the previous work(Ju et al., 2017). In fact,there are various architectures of CNN, such as LeNet, AlexNet, and VGGNet. Among all the architectures of CNN, LeNet-5 cost less training time due to its simple structure. Therefore, in the study, we chose to implement the most classical CNN architecture, LeNet-5, directly from the Keras Library. There were 5 layers, including 2 convolutional layers, 2 subsampling layers and 2 fully connected layers in it. Compared to the MLP, LeNet-5 took 10 epochs to reach a higher test accuracy of 97.7%(Table 1), which is higher than any other attempt we did for MLP.

Researching further for the reason why the CNN model performed better than MLP, we roughly draw the following inference by reading the previous work done by T.Schaijk.(Schaijk, 2019). In a word, CNN takes advantage of local spatial coherence of images. This means that they are able to reduce dramatically the number of operation needed to process an image by using convolution on patches of adjacent pixels, because adjacent pixels together are meaningful. In addition, There are also the pooling layers, which downscale the image inputs. Compared to CNN, MLP as a model takes vectors as input, can not downscale the input size as there is no coherence between an input and the one next to it.

6 Discussion and Conclusions

The MLP model has very high performance on the MNIST dataset. This dataset is relatively simple facing a neural network. Lateral comparisons in the community shows the high performance of even classification models, which is another benchmark that would range the results of an MLP to be high.

Our findings per the activation functions are similar. Although one can observe a clear difference in accuracy after 1 epoch, all models eventually converge to similar high accuracies by the end of the 10th epoch. This may be due to the weight initialization used that are random numbers very close to 0. When initializing in this fashion, tanh, sigmoid and ReLU tend to all have similar performance.

An extension of this study would be to experiment with larger values for weight initialization or by drawing them from a gaussian distribution with standard deviation of $\sqrt{2/n}$, where n is the number of inputs to the neuron(sta). This appears to be the current recommendation for use in practice in the specific case of neural networks with ReLU neurons. We speculate that this would results in accuracy trends more inline with what we saw in class, meaning lower accuracies for tanh and sigmoid and higher ones for ReLU that has the benefit of elimination vanishing gradients.

7 Statement of Contributions

All authors contributed equally to the report and coding components of this project. In particular,

Author 2 contributed to the CNN experiment.

References

Cs231n: Convolutional neural networks for visual recognition.

Simon Bernard, Laurent Heutte, and Sébastien Adam. 2007. Using random forests for handwritten digit recognition. volume 2, pages 1043–1047.

Leo Breiman. 2001. Random forests.

Jason Brownlee. 2018. How to use weight decay to reduce overfitting of neural network in keras. *Machine Learning Mastery*.

M.W Gardner and S.R Dorling. 1998. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric Environment*, 32(14):2627–2636.

Casper Hansen. 2020. Neural network from scratch with numpy and mnist. *Machine Learning From Scratch*.

Cheng Ju, Aurelien Bibaut, and Mark J. van der Laan. 2017. [link].

Aryan Mobiny, Pengyu Yuan, Supratik K. Moulik, Naveen Garg, Carol C. Wu, and Hien Van Nguyen. 2019. Dropconnect is effective in modeling uncertainty of bayesian deep networks. *Scientific Reports*, 11(1).

Tom Van Schaijk. 2019. *Image classification: MLP vs CNN*.

A Reported Accuracy Table

Activation function	Units in hidden layers	Epoch	Accuracy		Epoch	Accuracy	
			Train	Test		Train	Test
ReLU	()	1	0.758	0.765	10	0.895	0.901
	(128)	1	0.909	0.913	10	0.963	0.955
	(128, 128)	1	0.917	0.920	10	0.974	0.965
Tanh	(128, 128)	1	0.515	0.528	10	0.936	0.936
Sigmoid	(128, 128)	1	0.734	0.740	10	0.913	0.915
ReLU with L2	(128, 128)	1	0.853	0.856	10	0.853	0.861
ReLU with unnormalized data	(128, 128)	1	0.104	0.103	10	0.104	0.103
ReLU for Creativity	(512, 512)	1	0.948	0.945	10	0.984	0.969
	(256, 256)	1	0.934	0.932	10	0.979	0.967
	(128, 64)	1	0.909	0.911	10	0.971	0.959
	(64, 64)	1	0.887	0.891	10	0.964	0.955
CNN for Creativity	LeNet-5	1	N/A	0.927	10	N/A	0.977

Table 1: Reported accuracy for model variants