Classification of Image Data Using Multilayer Perceptron(MLP)

Group 80: Pingsheng Li Jacob Wang Xiyuan Feng

Abstract

This report primarily focuses on the performance of Multilayer Perceptron(MLP) model on the task of classifying images of hand-written numbers(from MNIST hand-written digit database). We examined the effects of hidden layers depths, activation functions, and L2 regularization on the performance of the model. Also, we explored and compared different data preprocessing procedures. Additionally, we investigated the effects of the width of hidden layers, and compared the overall performance of MLP with that of the K-Nearest Neighbors algorithm(KNN). With 2 hidden layers of 700 units, proper L2 regularization, and ReLU activation function, the best accuracy of MLP we obtained is 95.7%, which is close to that of KNN classifiers. However, both models have their pros and cons in practice. Plus, we found that the initialization of the parameters of models and the preprocessing of raw data will have profound impacts on the performance of MLP models.

1.Introduction

Designing artificial intelligence that can interpret the real visual world is one of the core goals of computer vision. Nowadays, many practices of computer vision have shown great potential, such as facial recognition, image classifications, etc. In this report, we attempt to compare the different data preprocessing methods and analyze the effects of hidden layers, activation functions, and L2 regularization on the performance of Multilayer Perceptron(MLP). Also, we conducted a comparative analysis between MLP and KNN algorithms.

1.1 Background Information

Multilayer Perceptron(MLP) is a class of feedforward artificial neural network model, which consists of an input layer, hidden layer(s) that mostly use nonlinear activation functions to filter/transform the input from the last layer, and an output layer.

Activation function defines the output of a neuron in neural networks based on its inputs. Most activation functions will apply non-linear transformation to the inputs.

1.2 Task & Data Description

In this project, we analyzed the performance of MLP classifiers on the task of handwritten digit image classification. We used the MNIST handwritten digit database in tensorflow for evaluating the powers of MLP models. The main task is to map the images of digits to corresponding numbers(from 0 to 9). Each image is represented by a 2D matrix with 28 rows and 28 columns, where each entry gives the brightness of a pixel(0 is the darkest while 255 is the brightest).

This task can in general be divided into three sections. First we preprocessed the data by dividing the value in each pixel by 255, which will precisely scale the range of entries into the interval [0, 1]. Alternatively, we could have also normalized the data, so the data will have a mean of 0 and standard deviation of 1. The comparison between these two preprocessing procedures will be discussed later. Then, we implemented the MLP model with backpropagation and mini-batch stochastic gradient descent from scratch as required. Additionally, we conducted a series of experiments to examine the effects of hidden layer depth & width, activation functions, and L2 regularization on the performance of MLP algorithms. We also compared its performance with KNN algorithm, and analyzed the pros and cons of both models.

1.3 Important Findings

We discovered that scaling the pixels into the interval of [0, 1] by dividing 255 gives the best result(95.70%) given L2 regularization (lambda or alpha=0.1) and 2 hidden layers of 700 units with ReLU activation function. However, unprocessed data or normalized data(0 mean; 1 std) can rarely give a good result (mostly around 11.1%) maybe because of converging to a bad local minimum or numerical errors during computation. We also found that increasing hidden layer depths and applying proper L2 regularization can generally improve the performance of MLP, and different activation functions have distinct features and ReLU gives the best performance(94.74%) after 13 epochs when no regularization was applied into a MLP with 2 hidden layers. L2 regularization generally helps improve MLP's performance, but when excessive regularization is applied, the problem of weight decaying can lead to underfitting quickly. Besides, KNN algorithm can also give excellent performance(96.91%, K=5), even better than MLP, but when dealing with large testing datasets, the prediction time of KNN is hard to optimize(better data structure required), but for MLP, it only takes linear time(constant forward pass time).

1.4 Related Work

Previously, Cireşan and Meier, et al.(2012) summarized a series of techniques that are helpful for improving the performance of MLP (best 99.69%), including increasing the number of hidden layers, increasing the number of units per layer, applying regularization, and introducing many deformed training images to avoid overfitting. This work inspired us to design similar experiments to support the conclusion.

Also, Grover and Toghi (2020) attempted to utilize KNN algorithms to perform the same handwritten digit recognition task and obtained excellent results(97.73%). This work also inspired us to relate the performance of KNN algorithms with that of MLP on the same task, and analyze their advantages and limitations.

2. Data and Set up

In this section, we provide the context of our datasets, and common preprocessing procedures.

2.1 Data Source

The handwritten digits data are from Modified National Institute of Standards and Technology database(MNIST). In this report, data was imported via tensorflow package.

2.2 Pre-processing

The preprocessing phase includes: first reshape all images, i.e. 2D arrays, into 1D arrays. Then, we divided all entries by 255, which scaled the data within [0, 1] interval. Alternatively, we can subtract entries by mean and divide them by standard deviation, which will normalize the data into ones having 0 mean and 1 standard deviation.

2.3 Information

In the MNIST dataset, there are 70,000 images of handwritten digits(from 0 to 9), 60,000 of which are in the training set and others are in the testing set. The ten classes of digits(from 0 to 9) are relatively evenly distributed. Each image is represented as a 2D square matrix with 28 rows & columns, which gives 784 pixels in total. In each pixel, a number ranging from 0 to 255 is used to represent the brightness in that pixel(0 is the darkest while 255 is the brightest).

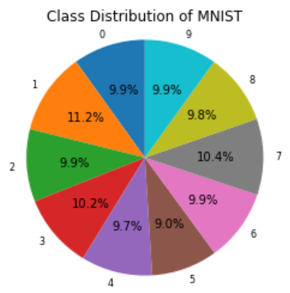


Figure 1: The class distribution of each number

3. Results

3.1 The Effects of Hidden Layer Depth

We examined the performance of MLP models with different hidden layer depths. The accuracy of MLP with no hidden layer and those of MLP with 1&2 hidden layer(s) of 128 units using ReLU were compared together. No regularization was applied.

	No HL	1 HL	2 HL
Best epoch (Test)	No. 8	No. 10	No. 13
Training Score	94.23%	76.34%	99.83%
Accuracy (Test)	92.69%	71.67%	**94.74%**
Variance (Test)	Low	High	Low

HL: hidden layer
Mini-batch size for SGD is 200.

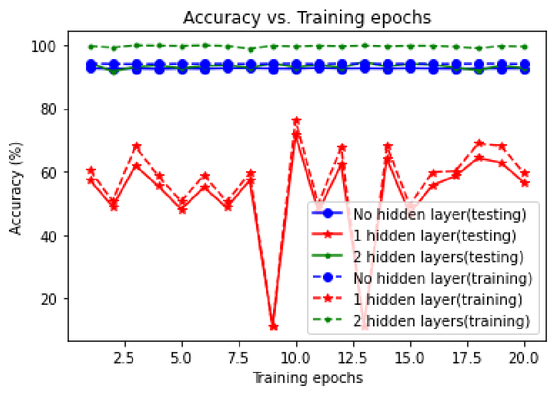


Figure 2: Accuracy of 3 MLP with different depths vs. training epochs

The conclusion is that adding more hidden layers, which gives non-linear properties to the model, mostly could be helpful for better accuracy, but it's not always the case. MLP with a certain number of hidden layers could easily get stuck in a bad local minimum. In figure 2, having 1 hidden layer is even worse than having no hidden layer, but adding another hidden layer rescues the performance. Since the optimization of MLP is a non-convex problem, at a certain epoch, the MLP with one hidden layer converges to a bad local minimum, generating major failure. But the stochastic gradient descent(SGD) method can help the model to get out of the local minimum after one more epoch, so the accuracy bounced up. Besides, more detailed analysis shows that MLP with 2 hidden layers has greater variance and needs more epochs to achieve optimum than MLP with no hidden layer. References also suggest deeper MLP models are more likely to achieve better accuracy(Cireşan; 2012).

3.2 Activation Functions

There are three activation functions considered in this experiment: ReLU, sigmoid, and tanh.

The definition is the following:

$$ReLU(x) = max(0, x)$$

$$sigmoid(x) = \frac{1}{(1+e^{-x})}$$

$$tanh(x) = 2sigmoid(2x) - 1 = \frac{2}{1+e^{-2x}} - 1$$

Among these, sigmoid and tanh are 2 historically traditional activation functions, but ReLU has become more popular in recent years. We compared the influence of these activation functions in MLP with 2 hidden layers of 128 units and illustrated how their accuracies change with training epochs.

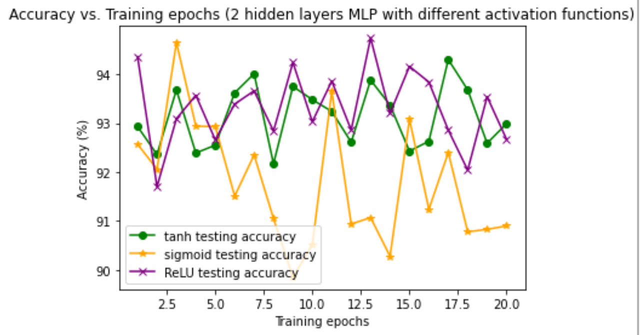


Figure 3: Testing Accuracy of MLP with 2 hidden layers using different activation functions vs. training epochs

In figure 3, we found that the sigmoid function gave a relatively unstable performance: despite its high accuracy at epoch 3, the performance on the testing set deteriorated with more training epochs. In contrast, the performance of tanh function is most stable(low variance), but it never achieves the same level of high accuracy like ReLU and sigmoid do. ReLU has both relatively stable performance (relatively low variance) and capacity of achieving high accuracy, therefore considered as the winner among all three functions.

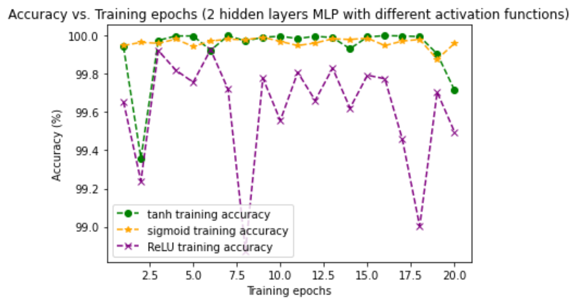


Figure 4: Training Accuracy of MLP with 2 hidden layers using different activation functions vs. training epochs

In figure 4, we can see all three functions were able to fit the training data with mostly 99+% accuracy, but sigmoid and tanh have less variance than ReLU. However, sigmoid and tanh cannot maintain such stability on the testing set, which is a sign of overfitting. Proper regularization, however, should be able to ease this issue.

	Sigmoid	Tanh	ReLU
Best epoch (Test)	No. 3	No. 17	No. 13
Training Score	99.96%	99.99%	99.83%
Accuracy (Test)	94.65%	94.3%	**94.74%**
Variance (Test)	High	Low	Medium

Besides the experimental results above, ReLU is cheap to compute and perfectly resolves the issue of vanishing gradient, a problem that prevents MLP from further training when using sigmoid or tanh, because ReLU guarantees a constant 0-or-1 gradient while the gradients of tanh and sigmoid diminish as the model fits more to the dataset. Therefore, ReLU was considered as the winner among all candidates.

3.3 Regularization

In the previous two experiments, no regularization was applied during training, so we now explored how regularization improves the performance of MLP models. We focused on L2 regularization.

We found that proper L2 regularization effectively promotes the accuracy and reduces the variance of MLP models.

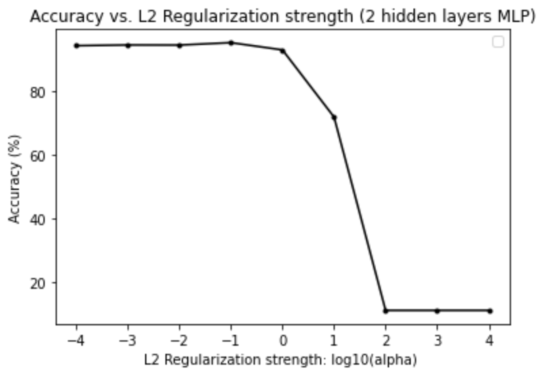
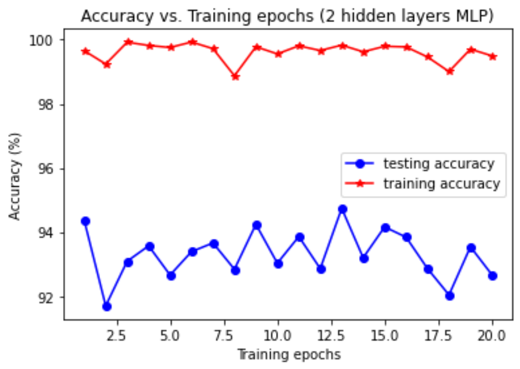


Figure 5: Testing Accuracy of MLP with 2 hidden layers vs. log (L2 regularization strength alpha)

In figure 5, the testing accuracy achieved optimum (95%) when regularization strength alpha = 0.1, which is better than no regularization. Also, excessive regularization causes rapid weight decay, leading to severe underfitting as expected. Since MLP optimization is a con-convex problem, and the cost landscape in high dimensional space is rather bumpy, hampering gradient descent method to achieve global minimum, so adding regularization would smooth the cost landscape and facilitate optimization. Therefore, proper L2 regularization indeed increases the accuracy.



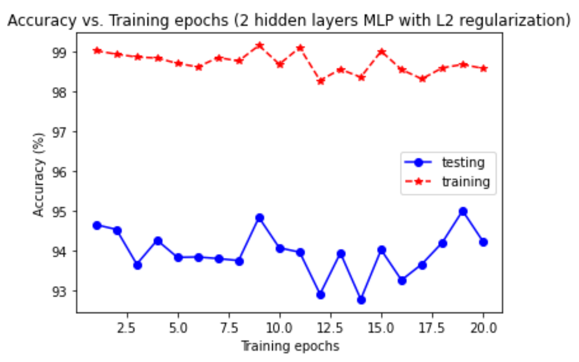


Figure 6: Performance of MLP(2 hidden layers) with(top) or without(bottom) L2 Regularization (alpha = 0.1) vs. epochs

In figure 6, the fluctuation of testing accuracy, especially for the first 8 epochs, was effectively eased when regularization was applied. This observation supports that regularization reduces the variance and improves accuracy.

3.4 Image Normalization

In previous experiments, pixels of images were divided by 255 before feeded into all models. In this section, we examined the consequence of using unnormalized data or using an alternative normalization approach.

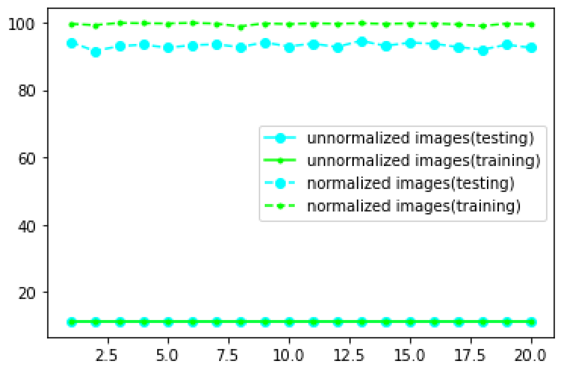


Figure 7: Accuracy of MLP(2 hidden layers) using raw or divided-by-255 images vs. training epochs

In figure 7, we observed that if no normalization was applied, MLP would fail catastrophically. One possible reason is that large values are less numerically stable for complex models. Thus, arithmetic overflow could be fatal. Also, larger values are more difficult to compute, especially for computationally expensive models like deep MLPs, so normalization is pivotal here.

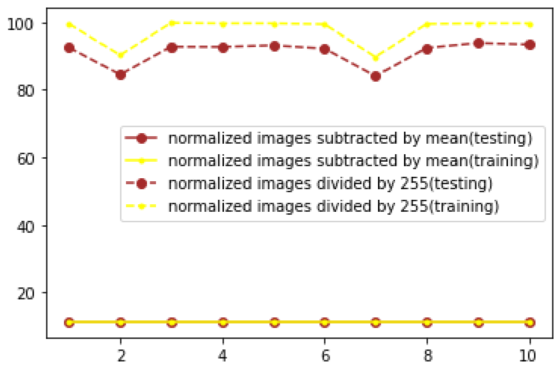


Figure 8: Accuracy of MLP (2 hidden layers) using different normalization techniques vs. training epochs

Besides, we also tried to normalize images by subtracting mean and dividing standard deviation. However, it seems that such normalization also leads to major failure. One possible reason is that such normalization will convert pixels with 0, a major component of images, into ones with negative values, causing many inactive hidden units when using ReLU.

Another observation is that the percentage of number 1 in the dataset is 11.2%, which coincides with the accuracy of failed MLP models, so it's reasonable to assume that these failed models are never trained but to guess every input to be 1. To summarize, preprocessing that gives positive and small input is better when training MLP models.

3.5 Additional Analysis & Optimization

3.5.1 Hidden Layer Width

Besides the experiments above, we also investigated the effect of hidden layer width. L2 regularization was applied to get better performance (alpha=0.1).

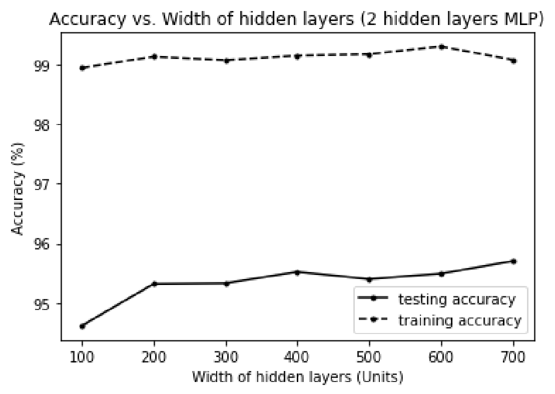


Figure 9: Best accuracy of regularized MLP (2 hidden layers) vs. the width of hidden layers (units in each hidden layer)

In figure 9, it's clear that wider hidden layers stably promotes the best accuracy of MLP. The best accuracy obtained is 95.7% when width=700. So increasing the width of hidden layers is a good approach to improve MLP performance.

3.5.2 Comparison with KNN

A comprehensive analysis has been conducted (Grover, et al.; 2020) for the performance of KNN on MNIST dataset, and the best accuracy they obtained is 97.73% (K = 3) using various techniques. Using a pre-implemented KNN model in sklearn package, we approximated this result.

	K = 5	K = 10	K = 20
Accuracy	96.91%	96.65%	96.25%

But, one disadvantage of KNN is that when the training dataset is huge, KNN needs a very long time to do searching, and its optimization relies on data structures like KD Tree, etc. Also, if the dataset is unbalanced, the prediction(search) time of KNN would be very unstable. Also, the prediction time increases as the size of training data grows. In contrast, MLP always uses constant time for prediction (a single forward pass); its prediction time is independent of the size of the training set. Therefore, in practice, when a model trained with a large training dataset needs to make a large amount of predictions, MLP would complete the task faster.

3.5.3 Normalized Initialization

Besides the required features in MLP(mini-batch SGD & backpropagation), we also implemented initialization normalization(Xavier, et al., 2010) in our MLP model from scratch. In initialization, bias was set to 0 and the weights W_{ij}, at each layer follow the uniform distribution U:

$$W \sim U\left[-\frac{\sqrt{6}}{\sqrt{n_j+n_{j+1}}}, \frac{\sqrt{6}}{\sqrt{n_j+n_{j+1}}}\right]$$

n_j & n_{j+1} is the width of previous & current layer.

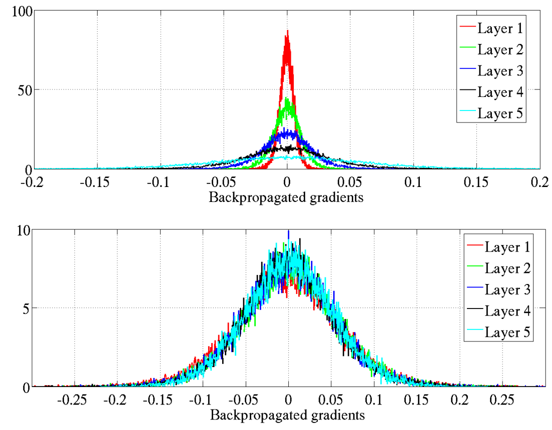


Figure 10: Back-propagated gradients with standard (top) vs normalized (bottom) initialization using tanh. Adapted from "Understanding the difficulty of training deep feedforward neural networks" by Xavier G. et al., 2010, *Journal of Machine Learning Research*

Figure 10 illustrates that such normalized initialization effectively maintains the variance of gradients over layers, which can be an advantage of normalized initialization since having gradients of inconsistent magnitudes at different layers may yield to ill-conditioning and slower training. Maintaining variance across all layers allows the weights at each layer converge at similar rate, so we implemented this method of initialization.

3.5.4 Momentum in SGD

In stochastic gradient descent method(SGD), we implemented momentum feature, which takes the gradient of previous iterations into consideration. The final gradient in an iteration is determined by an exponentially weighted average of gradients from the previous and current iterations. This technique accelerates the convergence of SGD, and also it helps models escaping the unstable local minimum since convergence using SGD with momentum requires a series of stably decreasing gradients. In this report, stucking in the local minimum is a common issue, so it's pivotal to implement momentum features in SGD to acquire better results.

4. Discussion & Conclusion

4.1 Takeaways

First, deeper hidden layers generally improve accuracy; shadow MLP sometimes would get stuck in bad local minimums and may be underfitting. For activation functions, ReLU could achieve a high accuracy while keeping a medium variance. Also, ReLU is cheap to compute and perfectly resolves the issue of vanishing gradient which prevents MLP from further training when using sigmoid or tanh. So ReLU wins over sigmoid and tanh. Additionally, proper L2 regularization can boost the performance of MLP and effectively reduce model's variance. Plus, in order to avoid numerical issues during the training of MLP, it's a good practice to transform the input within a non-negative, small interval, like [0, 1]. Otherwise, there could be fatal arithmetic overflow causing major performance failures. Having many negative inputs can also destroy the model by causing a large number of inactive units when using ReLU activation function. Besides, increasing the width of MLP is often a good approach to improve accuracy although it also increases the training time of MLP. Lastly, both KNN and MLP can achieve excellent accuracy in digit recognition, but unlike KNN, whose searching time increases as the training set grows larger, trained MLP takes constant time to make each prediction, and MLP doesn't need a balanced data structure to keep a stable prediction time, therefore more suitable for large scale classification tasks when the training set is large as well.

4.2 Future Work

In the future work, we may try how different regularization techniques, such as L1 regularization, early stopping, and dropout, etc., affect MLP's performance. Another inspiration from random forest is that we could train multiple MLPs simultaneously and take the majority vote to determine the output, which can reduce the errors produced by MLP accidentally stuck in a bad local minimum. Lastly, data augmentation could also be considered. All topics above could be further explored in the future.

Statement of Contribution

Pingsheng Li: Data preprocessing & experiment design and analysis of data
Jacob Wang: Implementation and optimization of Multilayer Perceptron(MLP) models
Xiyuan Feng: Collect experimental data & compilation of the report

Reference

Cireşan D.C., Meier U., Gambardella L.M., Schmidhuber J. (2012) Deep Big Multilayer Perceptrons for Digit Recognition. In: Montavon G., Orr G.B., Müller KR. (eds) Neural Networks: Tricks of the Trade. Lecture Notes in Computer Science, vol 7700. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-35289-8_31

Grover D., Toghi B. (2020) MNIST Dataset Classification Utilizing k-NN Classifier with Modified Sliding-Window Metric. In: Arai K., Kapoor S. (eds) Advances in Computer Vision. CVC 2019. Advances in Intelligent Systems and Computing, vol 944. Springer, Cham. https://doi.org/10.1007/978-3-030-17798-0_47
Yann LeCun, Corinna Cortes, Christopher J.C. Burges: THE MNIST DATABASE of handwritten digits http://yann.lecun.com/exdb/mnist/

Glorot, Xavier & Bengio, Y.. (2010). Understanding the difficulty of training deep feedforward neural networks. Journal of Machine Learning Research - Proceedings Track. 9. 249-256.

LeCun, Y. & Cortes, C. (2010). MNIST handwritten digit database. , .