COMP551 Mini Project3:
Classification of Image Data using MLP algorithm

Group 95

Yao Chen (260910483)
Chuyang Zhang (260840707)
Hanzhi Zhang (260908416)

Abstract

In this mini-project, a multilayered perception model(MLP) is implemented for classifying a set of fashion MNIST data. The primary goal is to correctly classify the type of clothes each input image belongs to. To achieve this objective, we performed a series of experiments on data preprocessing, modelling building and hyperparameter tuning. We find that to achieve a higher classification, a model with more hidden layers, a lower learning rate and a reasonable batch size is required. As a trade-off, a high training accuracy is correlated with a longer training time. The experimental results show that an MLP with two hidden layers could achieve a classification accuracy of 89.6% when using a learning rate of 0.03, a batch size of 950, alpha of 0.6 for leaky-ReLU function and an epoch of 550.

Introduction

There are 86 billion neurons in the human brain that work together to process and understand words and numbers [1]. Once the eyes receive a signal, humans can naturally comprehend a piece of image content by visual inspection, even without thinking [1]. People can easily pass the google "I am not a robot" check. It is impressive that billions of neurons could cooperate and produce the correct signal, which leads to understanding [1]. However, humans are not good at extensive data processing. For example, it becomes difficult for humans to assign a label to an image beyond a certain number. Computer vision is increasingly employed in current industries to help search engines with shopping items [2]. It also involves the conversion of videos/pictures into signals for further processing [2].

Image classification is the main branch of computer vision via a supervised machine learning approach. It is a process of giving an image and assigning a specific class label to it or reporting the probability that an image belongs to a particular class label. This project introduced a new Modified National Institute of Standards and Technology (MNIST) dataset — a fashion version [3]. The fashion MNIST dataset contains images of clothes for a multilabel classification task. Using MNIST requires less time for image preprocessing, and the class label is already assigned to each picture.

In this project, we implemented a multilayer perceptron (MLP) model with a different number of hidden layers and many activation functions to classify ten different types of clothes and measure the accuracy of this multi-class classification.

Datasets

This dataset is an MNIST-like dataset of 70,000 labelled fashion images with the size of 28x28 pixels, with a total of 784 pixels for each image. It is split into a training set (60,000 image instances) and a test set (10,000 image instances) by default. Each picture belongs to one of 10 classes (T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, and Ankle boot). The ten classes are represented by numbers from 0 to 9 for simplification.

After the data is loaded, the values returned are two tuples. Each contains the grayscale value for each pixel and the corresponding class. We first subtracted the training and test data by the mean value and then divided it by the standard deviation to preprocess the data. The data is now normalized. Secondly, we reshaped the two-dimensional 28 by 28 pixels into one-dimension 784 pixels for a more straightforward calculation later. Finally, we vectorize the label value. Initially, one number for the label indicates which class the image belongs to, and then we changed it to an array with all zeros except for

one. The array at that number minus one index is 1, indicating which category the image belongs to, and the rest are 0. Besides, we also looked at the class distribution in the fashion MNIST dataset. There are 10 classes labelled in training set with numbers from "0" to "9", each indicating a particular item. In the training set, the number of instances is equally distributed for each class such that every class has 6000 instances. Meanwhile, every class has 1000 instances in the test set.

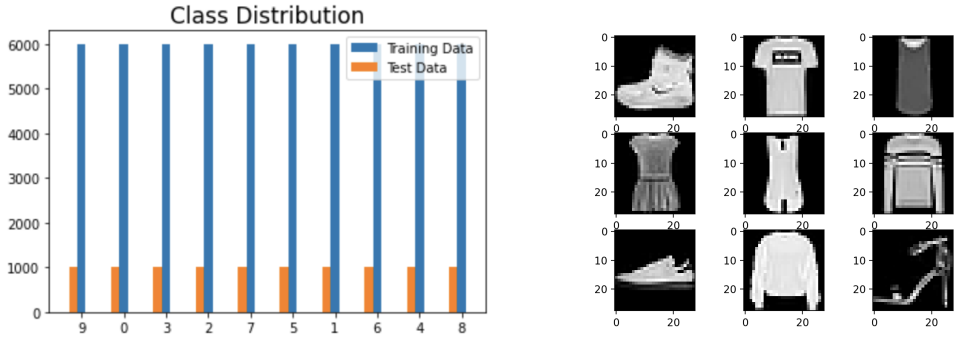

Figure1: The class distribution(Left) and A Subset of Images(Right) of Fashion-MNIST Dataset

Results

Different Layers used in MLP

We create three MLP models with different numbers of layers: (1) with no hidden layers, (2) with single hidden layer, (3) with two hidden layers. Each hidden layer has 128 units, and also we choose ReLU as the activation function. Then we tuned the learning rate, batch number and epoch number to achieve their best performance. We found that as the learning rate increases, the accuracy of prediction diverges, which due to it, the larger learning rate may miss the optima instead of hitting it. To be more specific, for the 2 hidden layers MLP model, since there is not much difference in accuracy when the learning rate is between 0.001 - 0.03, we pick 0.03 to save some training time. Then we find that the batch number overall would not affect the accuracy of prediction but can reduce the running time. Then for epoch number, increasing its value can first higher the accuracy until reaching maxima, then the accuracy decrease. This is due to overmuch of epoch numbers will lead to overfitting. Referring to Table1, plug in the tuned hyperparameters in each model, MLP model with 2 hidden layers has a better ability to filter the useful information to perform classification with an accuracy of 87.9%.

Table1: tuned hyperparameter on each MPL model

	learning rate	batch number	epoch number	accuracy
No hidden layer	0.003	800	90	81.3%
A single hidden layer	0.009	750	950	86.8%
2 hidden layers	0.03	950	550	87.9%

Different activation functions used

From here, we used the MLP model with 2 hidden layers for further experiments. We considered two different activation functions, hyperbolic tan(tanh) and Leaky-ReLU to see how they performed differently. We find that hyperbolic tan(tanh) did not work as well as the Leaky-ReLU. The test accuracy for tanh is 86.5%, and the test accuracy for Leaky-ReLU is 88.3% under the same setting of hyperparameters. The Leaky-ReLU activation function shows a better performance since it avoids the issue of gradient vanishing problem and fastens the learning process. The Leaky-ReLU function modifies the ReLU function to allow small negative values when the input is less than zero. It fixes the zero-gradient problem such that we have a very small gradient instead of zero gradients when the input

values are negative, which provides a chance for the network to continue its learning. Since Leaky-ReLU avoids the zero-gradient problem, it always shows similar or even better results than ReLU.

The tanh activation function is better than the sigmoid function because it has a higher range than the sigmoid. The tanh function is a zero-centred, symmetric function with a range of -1 to 1, while the sigmoid function has a range of 0 to 1. However, both tanh and sigmoid might cause the vanishing gradient problem. This is because the maximum threshold for both functions equals 1. During the backpropagation process, as we go from the output layer toward the input layer, the gradients always get smaller and smaller and close to zero, which leads the weights of the initial or lower layers to be nearly unchanged.

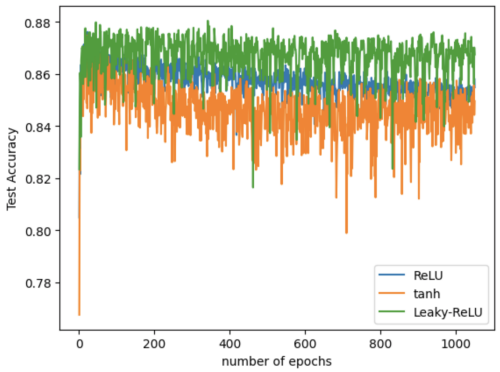

Figure2: Accuracy of using three different activation functions

Regularization: Dropout

Knowing that image data are rich in features, but if a model has too many parameters to deal with, the trained model is likely to have a long-running time and prone to overfitting. Therefore, we explored the idea of dropout to see how much it can affect the prediction accuracy. Referring to the graph below (Figure3 Left), we could see that when the dropout rate is around 0.1 - 0.2, the prediction accuracy(about 90.0%) is improved compared with the accuracy of the model without dropout (87.9%), under the other hyperparameters remains the same. However, when keep increasing the dropout rate, less information is captured by the model. Then underfitting exists and lowers the accuracy.

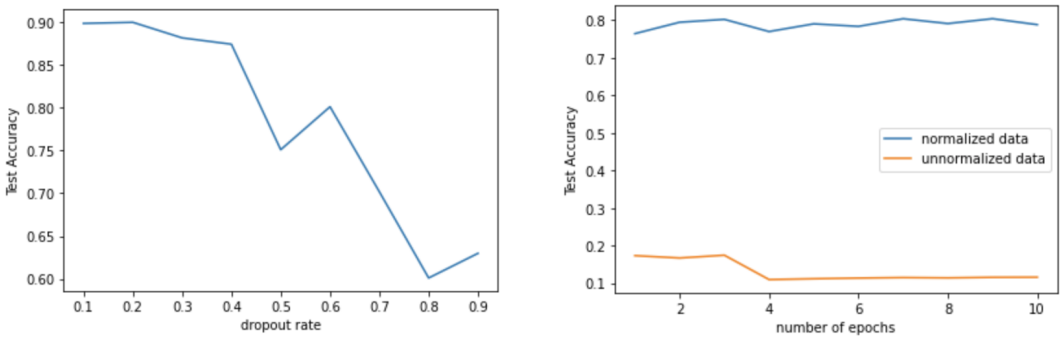

Figure 3: Accuracy of applying different generalization techniques Left: applying dropout with different dropout rates; Right: comparison of applying normalization on the dataset or not

Regularization: Normalization

Another regularization technique that we are interested in is aside from the training model. Instead, we tried to normalize the dataset when preprocessing it. By Figure3 Right, the accuracy of prediction on raw data is only between 10% - 20%, whereas training on the normalized data shows that

the accuracy improved significantly (rise to about 80%). Thus, we conclude that normalizing the dataset can raise the model's performance.

Convolutional Neural Network (CNN) VS. Multilayer Perception (MLP)

Here, we create a convolutional neural network (CNN) with two convolutional and two fully connected layers. Then we tuned this CNN model on the learning rate and the batch size. We first tuned the learning rate, referring to the plot below, which shows that the highest accuracy(about 0.85) is reached when the learning rate is set to be 0.003. A similar trade-off is happening again — a lower learning rate takes more running time but can find an optimum, while a larger learning rate has more chance to miss the optima. Then the batch size does not affect the prediction much, the accuracy floating around 85% to 88%. Which is similar to the 2-hidden layered MLP and slightly higher on average.

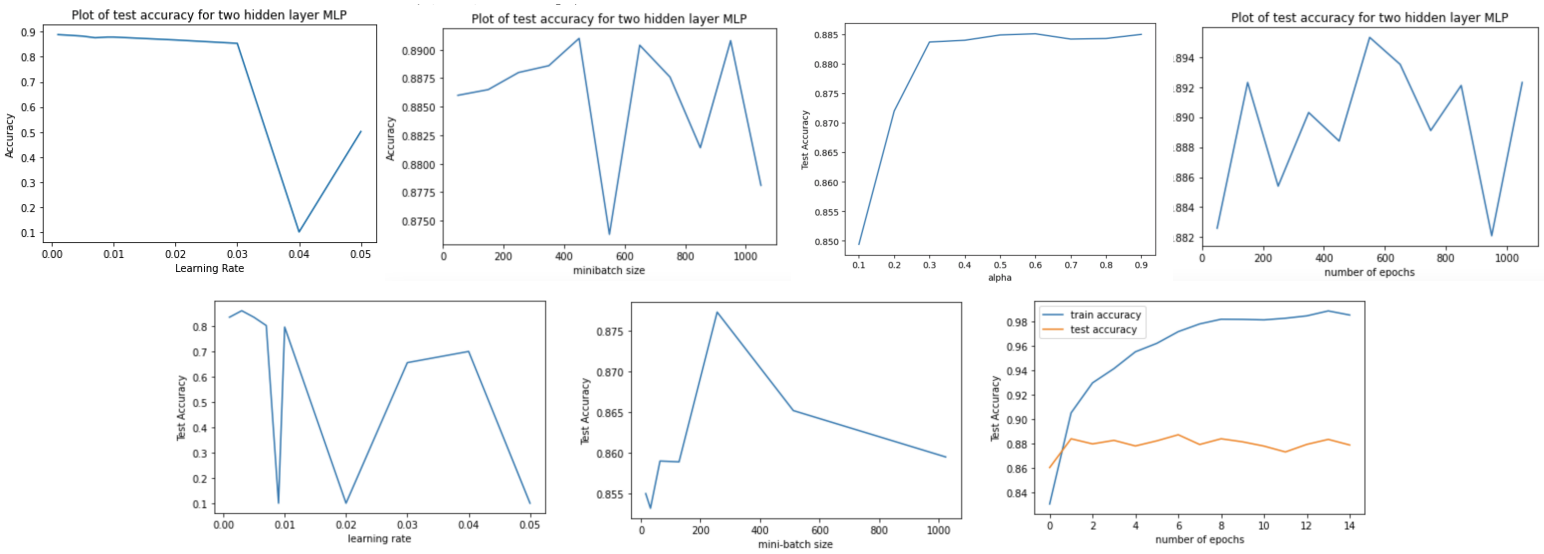

Figure 4: Plots of tuning different hyperparameters in the CNN and the 2-layer MLP model. Upper: MLP model; Lower CNN model

To achieve a better performance of the MLP model, we tuned the alpha value in the leaky-ReLU activation function. This shows that the model works the best when alpha is around 0.5 - 0.6. Compared to this CNN with a fully tuned MLP model, the prediction accuracy is 88% and 89.5%, respectively. So the final MLP model has a negligible higher accuracy than that of CNN. The table below indicates the hyperparameter values used.

Hyperparameter	Value
Learning rate	0.03
Batch size	950
leaky-ReLU _ alpha	0.6
Number of epoch	550

Table2: summary of hyperparameter used in MLP with two hidden layers

Discussion and Conclusion

Through this mini-project, we had fresh first-handed experiences with the MLP model, and we attempted to incorporate a few machine learning techniques to optimize the prediction accuracy.

We first implement the MLP model. And specifically made models with different numbers of hidden layers (no hidden layers to two hidden layers). Then for those three models, we focus on three hyper-parameters: learning rate, batch number, and epoch number. As a result, we find that the model with two hidden layers generally has a better performance. At the same time, the accuracy vs. training time trade-off was noticed.

Moreover, we also implemented two regularization methods: normalize and dropout. We first normalize our dataset by its mean and standard deviation, then fit data to our models. Our experiment turns out that the normalization of input data can improve the classification significantly. Then, we perform the dropout, which randomly disables a subset of neurons during training time to avoid overfitting. In our experiment, it does improve certain accuracy at a certain learning rate, but the improvement is not significant. This might be due to the input type of clothes being similar in shape.

However, when we performed experiments to see how CNN is different from MLP. Recall that MLP takes a vector as input and CNN takes tensor as input so CNN can understand spatial relation between pixels of images better. So theoretically, CNN should work better than MLP. Yet in our experiment, there is no significant difference based on the final result. But it is worth noticing that the run time of the CNN model is less than that of running the MLP model, which matches our expectation.

After performing all the experiments, we discussed how we could do better. We notice that the way we process the image data is only based on a statistical idea where there actually exists more approaches to increasing the quality and quantity of the input data, such as Gaussian blurring, which can remove the noise in the image yet preserve the characters [4].

To wrap up, we had explored how different numbers of the hidden layers and different hyperparameters affect the classification accuracy of the MLP model. And finally, achieved an accuracy of 89.5%, we can believe that it is good enough for classification on the fashion MNIST dataset.

Statements of contribution

The group members distributed the workload equally.
Yao: data preprocessing, graph generation, MLP algorithm implementation, creativity.
Chuyang: data preprocessing, MLP algorithm implementation, creativity.
Hanzhi: data preprocessing, write-up, creativity.

Reference

[1]Herculano-Houzel S. (2009). The human brain in numbers: a linearly scaled-up primate brain. *Frontiers in human neuroscience*, *3*, 31. https://doi.org/10.3389/neuro.09.031.2009

[2]David Forsyth, Jean Ponce. Computer Vision: A Modern Approach. (Second edition). Prentice Hall, pp.792, 2011, 978-0136085928.

[3]J. Brownlee, “Deep learning CNN for fashion-mnist clothing classification,” *Machine Learning Mastery*, 27-Aug-2020. [Online]. Available: https://machinelearningmastery.com/how-to-develop-a-cnn-from-scratch-for-fashion-mnist-clothing-classification/. [Accessed: 03-Apr-2022].

[4]ScienceDirect, “Gaussian blur,” *Gaussian Blur - an overview | ScienceDirect Topics*. [Online]. Available: https://www.sciencedirect.com/topics/engineering/gaussian-blur. [Accessed: 03-Apr-2022].