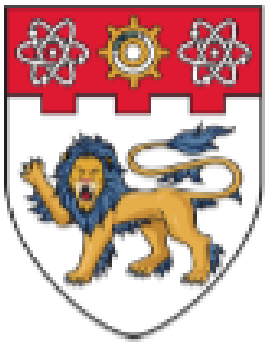

NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

Computer Vision

AI6121 Project

Mingzhi Lyu
Yifan Xu
Guiyuan Le

School of Computer Science and Engineering
Nov, 2020

AI6121 CV Project

Mingzhi Lyu*, Yifan Xu†, Guiyuan Le‡
School of Computer Science and Engineering
Nanyang Technological University
Singapore
Email: *LYUM0001@e.ntu.edu.sg, †S200048@e.ntu.edu.sg, ‡GLE001@e.ntu.edu.sg

***Abstract*—Handwritten recognition is an essential part of computer vision. Now the recognition of handwritten digits based on the MNIST dataset has become a hot topic in deep learning. Here, we will develop the former recognition to implement the recognition of handwritten letters. The dataset is A-Z handwritten alphabets and we have designed and developed our handwritten letters' recognition network by adopting multi-layer perceptron(MLP). So this report includes detailed description and discussion of our MLP network design as well as our evaluation results. Then we analyze in detail the effects of different network parameters, such as learning rates, on the training results. Finally, for improvement, at the end of the report, we adopt convolutional neural network, design and develop our CNN recognition network and have a evaluation over the test set.**

***Index Terms*—Alphabets' recognition, MLP, CNN**

I. Introduction

A. Dataset

The A-Z handwritten alphabets dataset comes from https://www.kaggle.com/sachinpatel21/az-handwritten-alphabets-in-csvformat. The dataset contains 370,000+ handwritten English Alphabets, the dataset contains 26 folders (A-Z) containing handwritten images in size 28*28 pixels, each alphabet in the image is centre fitted to 20*20 pixel box, and Each image is stored as Gray-level. The images are taken from NIST(https://www.nist.gov/srd/nist-special-database-19) and NMIST large dataset and few other sources which were then formatted as mentioned above.

We randomly divide the original dataset into two parts: training set and test set, and the proportion of test set is 0.05.

B. Multilayer Perceptron (MLP)

A multilayer perceptron (MLP) is a class of feedforward artificial neural network (ANN). The term MLP is used ambiguously, sometimes loosely to any feedforward ANN, sometimes strictly to refer to networks composed of multiple layers of perceptrons (with threshold activation). Multilayer perceptrons are sometimes colloquially referred to as "vanilla" neural networks, especially when they have a single hidden layer.

An MLP consists of at least three layers of nodes: an input layer, a hidden layer and an output layer. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training. Its multiple layers and non-linear activation distinguish MLP from a linear perceptron. It can distinguish data that is not linearly separable. MLPs are useful in research for their ability to solve problems stochastically, which often allows approximate solutions for extremely complex problems like fitness approximation. MLPs are universal function approximators as shown by Cybenko's theorem, so they can be used to create mathematical models by regression analysis. As classification is a particular case of regression when the response variable is categorical, MLPs make good classifier algorithms. MLPs were a popular machine learning solution in the 1980s, finding applications in diverse fields such as speech recognition, image recognition, and machine translation software, but thereafter faced strong competition from much simpler support vector machines. Interest in back-propagation networks returned due to the successes of deep learning. [1]

II. MLP Model Design

In order to implement this neural network, we will describe ideas of the MLP structure and develop the network step by step. As Fig 1 show, we initially design 784 input layers, 50 hidden layers and 26 output layers. In this report, we will discuss why and how to design the every code block in depth gradually and finally get a simple MLP framework to show some results.

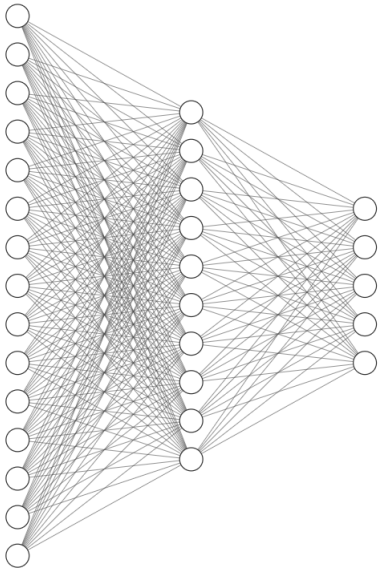

Fig. 1: Simplified MLP Structure

A. Data Feature

In deep learning, it's important to train the network basing on suitable training set. In this section, we discuss how we deal with raw data and generate the useful dataset.

Firstly, We import the Python libraries we need and use sklearn and pandas to cope with original file because the original CSV file didn't split dataset into training set and test set.

Secondly, we use the pandas function to transform original CSV data structure into pandas structure. And then extract some columns to split them into two parts. With the help of sklearn [2] functions, we choose 0.95 of the dataset as training set and 0.05 as test set, because there are 372450 datum in the original CSV file and 0.05 of the dataset is enough to be test set.

Finally, in order to improve the calculation speed, we set tensors in GPU environment. After that, we have got a new dataset that can be used as network training.

B. MLP Structure

In this part, we will design a vanilla MLP network. We use 2 linear layers and 1 active layer to construct MLP network basing on the Pytorch Class. The main code is shown below.

```
class two_layer_net(nn.Module):
    def __init__(self, input_size,\\
    hidden_size, output_size):
        super(two_layer_net, self).__init__()
        self.layer1 = nn.Linear(input_size, \\
        hidden_size, bias=False)
        self.layer2 = nn.Linear(hidden_size, \\
        output_size, bias=False)

    def forward(self, x):
        y = self.layer1(x)
        y_hat = F.relu(y)
        scores = self.layer2(y_hat)
        return scores
```

C. Training

After implementing the Pytorch [3] class, we use training set to feed the net and combine batch gradient descent to update weights. At the same time, we print the results every epoch and know whether the loss is decreasing and the error rates on training set and test set are being improved.

D. Results

At last, we can get a simple neural network and use it to recognize unknown letters. This simple network already has excellent abilities to classify the letters A-Z, and we will try to develop this network by different ways.

III. Experiment

Experiments on the designed MLP network was carried on for five hyper-parameters respectively: batch size, learning rate, the size of hidden layer, loss function, optimizers. The baseline to compare with the model with different hyper-parameters is the designed two-layer MLP network with $batchsize = 300$, $epoch = 100$, $learningrate = 0.2$, $hiddensize = 50$, $lossfunction = CrossEntropyLoss$ and $optimizer = SGD$. Here, we will compare the baseline with these models by adjusting each hyper-parameter one by one, while keeping other hyper-parameters unchanged.

Batch Size	Accuracy
100	0.974
300	0.979
900	0.983
1800	**0.984**
2700	0.983

TABLE I: Results of designed MLP with different batch sizes.

learning rate	Accuracy
0.2	0.9786
0.1	0.9834
0.05	0.9850
0.01	**0.9851**
0.001	0.9703

TABLE II: Results with different learning rates.

hidden Layer Size	Accuracy
50	0.9786
100	0.9853
200	0.9946
400	0.9960
800	**0.9963**

TABLE III: Results with different hidden size of the hidden layer.

According to Table I, compared to the baseline with $batchsize = 300$, the model trained with $batchsize = 1800$ gains the best performance. Larger the batch size is, faster the model converges, but less general it can be. For example, for the model trained with $batchsize = 2700$, its accuracy on the test dataset is lower than the one trained with $batchsize = 1800$.

According to Table II, with smaller learning rate, the performance of the model becomes better, because smaller learning rate can help parameters of the model approach the minimum more easily. However, when the learning rate is too small, it is likely to be trapped at the local minimum point, just as the result of $learningrate = 0.001$ shows.

According to Table III, layer size of the hidden layer is, the higher accuracy it can get. It is because the capacity of the model to simulate a template for the images increases as the number of the neurons increase. However, there will be some error that can not be erased because of the small different between training data and test data. Moreover, the model with more capacity tends to overfit the training data easily.

According to Table IV, we tried different loss function to train the baseline model. When the CrossEntropy Loss function is used, the accuracy reaches the maximum point. It is not surprising because the nature of this handwritten letter recognition problem is a multi-class classification, which cross entropy is designed for. For MSE Loss, L1 Loss and Smooth

Loss Function	Accuracy
MSE Loss	0.965
L1 Loss	0.543
Smooth L1 Loss	0.885
CrossEntropy Loss	**0.979**

TABLE IV: Results with different loss function.

Optimizer	Accuracy
Adam	0.122
AdamW	0.038
Adamax	0.769
SGD	**0.979**
Pprop	0.443
ASGD	0.968

TABLE V: Results with different optimizers.

L1 Loss, we firstly transferred the label of each instance with one-hot embedding. These loss functions try to minimize the distance between ground truth and predictions for each output feature without paying more attention to the feature obtaining the highest scores. Therefore, these loss function can do a better job on regression than the job on classification. However, Softmax operation on the output scores can alleviate this problem.

According to Table V, the baseline model utilizing SGD optimizer outperform all other optimizers. With Adam and AdamW, the model almost fails to make any correct prediction, especially AdamW optimizer, with which the model's performance is similar to random guess.

IV. Discussion And Improvement

MLP still has certain limitations for the fully connected network properties for image recognition. Moving and changing the target position in the image will greatly increase the difficulty of MLP training. Compared with complex image recognition processing, MLP performance is not very good. With the advent of convolutional neural network, CNN quickly became very popular in the field of machine learning image recognition.

In this section we would make a description of CNN and its advantages on image recognition, then design a proper CNN for the A-Z dataset training and make a report of the results on the test set.

A. Convolutional neural network

Convolutional neural network is a feed-forward neural network, its artificial neurons can respond to a part of the surrounding units in the coverage area, and it has excellent performance for large-scale image processing. The convolutional neural network consists of one or more convolutional layers and a fully connected layer at the top (corresponding to a classic neural network), and also includes associated weights and pooling layers. This structure enables convolutional neural networks to use the two-dimensional structure of the input data. Compared with other deep learning structures, convolutional neural networks can give better results in image and speech recognition. This model can also be trained using backpropagation algorithms. Compared with other deep, feed-forward neural networks, convolutional neural networks need to consider fewer parameters, making it an attractive deep learning structure.

CNN can be used to identify two-dimensional or three-dimensional images that are invariant to displacement, scaling and other forms of distortion. CNN's feature extraction layer parameters are learned through training data, so it avoids manual feature extraction, but learns from training data; secondly, neurons in the same feature map share weights, reducing network parameters, which is also a volume Jaeger Networks has a major advantage over fully connected networks. The special structure of sharing local weights is closer to the real biological neural network, which makes CNN have unique advantages in the fields of image processing and speech recognition. On the other hand, weight sharing reduces the complexity of the network and multi-dimensional input signals (voice, image) can directly input the characteristics of the network to avoid the process of data rearrangement in the process of feature extraction and classification. [4]

B. The peculiarities of CNN

CNN is a special deep neural network model. Its particularity is reflected in two aspects. On the one hand, the connections of its neurons are not fully connected, and on the other hand, the connections between certain neurons in the same layer are The weights are shared (that is, the same). Its non-fully connected and weight-sharing network structure makes it more similar to a biological neural network, reducing the complexity of the network model (for deep structures that are difficult to learn, this is very important), and reducing the weight quantity.

CNN is a machine learning model under deep supervised learning. It has strong adaptability. It is good at mining local features of data, extracting global training features and classification. Its weight sharing structure network makes it more similar to biological neural networks. Good results have been achieved in various fields of pattern recognition.

Sparse connection: In the BP neural network, the neuron nodes of each layer are a linear one-dimensional arrangement structure, and each neuron node of the layer is fully connected. In a convolutional neural network, the neuron nodes between layers are no longer fully connected, and the neuron nodes of each adjacent layer are only connected to the upper neuron nodes that are close to it by using the local spatial correlation between the layers. , That is, local connection. This greatly reduces the parameter scale of the neural network architecture.

Weight sharing: In the convolutional neural network, each convolution filter of the convolutional layer repeatedly acts on the entire receptive field to convolve the input image. The convolution result constitutes the feature map of the input image, and the image is extracted Local characteristics. Each convolution filter shares the same parameters, including the same weight matrix and bias terms. The advantage of sharing weights is that the location of local features does not need to

be considered when extracting features from an image. And weight sharing provides an effective way to greatly reduce the number of convolutional neural network model parameters to be learned.

Maximum pool sampling: It is a nonlinear down-sampling method. After obtaining image features through convolution, these features are used for classification. It is possible to use all the extracted feature data to train the classifier, but this usually produces a huge amount of calculation. Therefore, after obtaining the convolutional features of the image, the dimensionality of the convolutional features should be reduced by the maximum pool sampling method. The convolution feature is divided into several n*n disjoint regions, and the maximum (or average) feature of these regions is used to represent the convolution feature after dimensionality reduction. These reduced-dimensional features are easier to classify.

The value of maximum pool sampling in computer vision is embodied in two aspects: (1) it reduces the computational complexity from the upper hidden layer; (2) these pooling units have translation invariance, even if the image is small Displacement, the extracted features will remain unchanged. Due to the enhanced robustness to displacement, the maximum pool sampling method is an efficient sampling method to reduce the data dimension.

C. LeNet5 architecture

LetNet is an entry-level neural network model. It is a simple convolutional neural network that can be used for handwriting recognition.

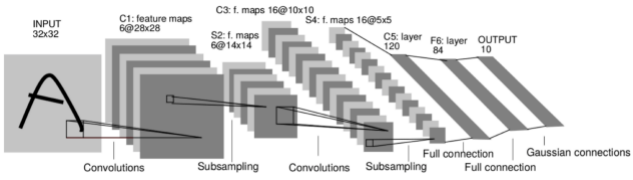

Fig. 2: LeNet5 Structure [5]

The handwritten font recognition model LeNet5 was born in 1994 and is one of the earliest convolutional neural networks. Through clever design, LeNet5 uses convolution, parameter sharing, pooling and other operations to extract features, avoiding a large amount of computational cost, and finally uses a fully connected neural network for classification and recognition. This network is also the starting point of a large number of neural network architectures recently.

The LeNet5 network model shown in the Fig 2 consists of seven layers of CNN (not including the input layer), which includes two convolutional layers, two subsampling layers (pooling layers), two fully connected layers, and the original image in the figure above The input size is 32*32 pixels, which will be adjusted to 28*28 in our training and testing. Other network parameters will be changed accordingly to adapt to the A-Z alphanumeric set for better training. The network architecture will be described in detail below. The activation function of the whole network is ReLU and the loss function is CrossEntropy.

1) Convolutional/MP Layers: Fig 3 shows the details of Convolutional layers and max-pooling layers. we would explain this below.

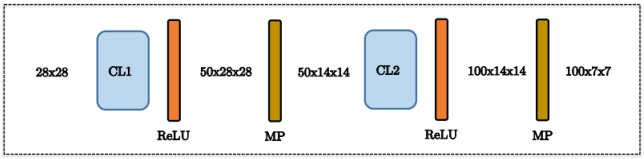

Fig. 3: Convolutional/MP Layers

The C1 layer is a convolutional layer, forming 50 feature maps. The input area size of the convolution is 3x3, and the parameters in each feature map are shared, that is, only one common convolution kernel is used in each feature map.

The S2 layer is a subsampling layer (Using the principle of image local correlation, sub-sampling the image can reduce the amount of data processing while retaining useful information). The 50 28x28 feature maps of the C1 layer were down-sampled with 2x2 as the unit to obtain 50 14x14 maps. The parameters in C1 and S2 are 50x3x3+50=500.

The C3 layer is also a convolutional layer with the convolution kernel size of 3x3 , forming 100 feature maps, so the output of this layer is 100x14x14.

The S4 layer is a max-pooling layer as before, and the output of this layer is 100x7x7. In this part, the architecture parameters would be 100x50x3x3+100=45100.

2) Linear Layers: After the convolution layers there are two Linear Layers.

The L5 layer, as the output from the former MP layer is 100x7x7, has 4900 input,and output is set to be 200. The parameters are 4900x200+200=980,200.

The L6 layer as the output layer has 200 input and 26 output because the label size of the dataset is 26. The parameters in this layer are 200x26+26=5226.

D. Performance of the LeNet5 on A-Z dataset

We applied the network architecture of LeNet5 described as before to train on the A-Z dataset to see the performance. The networks structure in Pytorch as shown below:

```
class LeNet5(nn.Module):
    def __init__(self):
        super(LeNet5, self).__init__()
        self.conv1 = nn.Conv2d(1,   50, \\
        kernel_size=3,  padding=1 )
        self.pool1  = nn.MaxPool2d(2,2)
        self.conv2 = nn.Conv2d(50,  100, \\
        kernel_size=3,  padding=1 )
        self.pool2 = nn.MaxPool2d(2,2)
        self.linear1 = nn.Linear(4900, 200)
        self.linear2 = nn.Linear(200,26)
    def forward(self, x):
        x = self.conv1(x)
        x = F.relu(x)
        x = self.pool1(x)
```

```
        x = self.conv2(x)
        x = F.relu(x)
        x = self.pool2(x)
        x = x.view(-1, 4900)
        x = self.linear1(x)
        x = F.relu(x)
        x = self.linear2(x)
        return x
```

We use the SGD optimizer and the cross-entropy loss function, and the batch size is set to be 1000 to reduce the number of updates. We used three different learning rates to train the dataset, and the training results are shown in the Table VI. The accuracy of the test set reaches the highest when the learning rate is 0.01, and the error curve of the test set is shown in the figure below. When the learning rate is 0.1, the loss decreases the fastest, but as the epoch increases, the loss decreases unstably and fluctuates up and down. When the learning rate is 0.001, the loss decreases slowly, but the most stably, and tends to be better as the epoch increases.

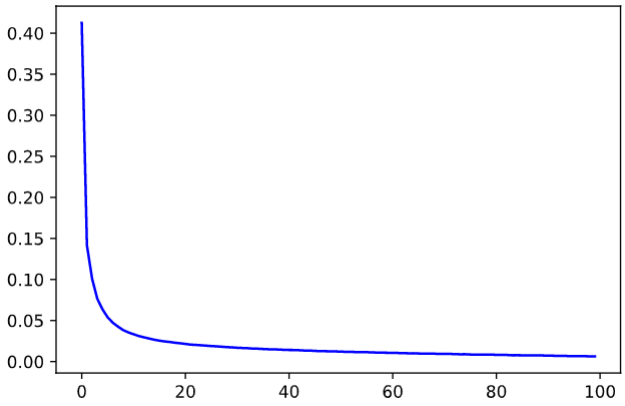

Fig. 4: Test error on learning rate 0.01

learning rate	Accuracy
0.1	0.9854
0.01	0.9927
0.001	0.9885

TABLE VI: Results of designed LeNet5 with different learning rates.

V. Summary

In this report, we implemented a handwritten letter recognition system on the A-Z Handwritten Alphabets dataset. We first introduced the dataset and the corresponding MLP network, introduced the MLP network and described the design in detail. This report contains the results of training and testing on the MLP with different parameters on this dataset, such as the number of neurons in the hidden layer, learning rate, loss function, SGD, etc., and achieved good results on the test set. For improvement, we used the CNN network to train and test the dataset. The network structure used the most popular LeNet5, which was introduced and analyzed in detail, and improved to adapt to the dataset of this experiment. And by using different learning rates to test, also achieved good results.

References

[1] Wikipedia contributors. Multilayer perceptron — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Multilayer_perceptron&oldid=961430969, 2020. [Online; accessed 23-November-2020].

[2] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[3] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[4] Wikipedia contributors. Convolutional neural network — Wikipedia, the free encyclopedia, 2020. [Online; accessed 24-November-2020].

[5] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[6] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[7] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90, May 2017.