Classification, Weight sharing, Auxiliary Losses

Kang Fu, Hangqi Li, Junhong Li
EPFL Switzerland

***Abstract*—The aim of this report is to use different neural networks, especially convolutional neural networks to achieve a classification goal. Different architectures need to be compared in this project, and assess the performance improvement that can be achieved through weight sharing, or using auxiliary losses. The project shows that using auxiliary loss could greatly increase the accuracy of the classification model, while weights sharing could also increase the model's accuracy.**

I. Introduction

In this project, The aim is to implement different architectures of convolutional neural networks to predict for the generated test dataset of a [1000,2,2,14] tensor size, which is corresponding to 1000 pairs of 14×14 gray-scale images. The generated training dataset is of the same size. The train target and test target indicate whether the first digit in the image is less or equal to the second. The other purpose of this project is to compare the impact of weight sharing and auxiliary loss on improving the CNN performance.

II. Project Structure

A. Methods

To improve the performance of our models, some useful methods and techniques are introduced in this part.

Batch normalization:

Batch normalization is a method used to make artificial neural networks faster and more stable through normalization of the layers' inputs by re-centering and re-scaling. It could make the network more robust regardless how the raw data distributes.

Dropout:

Randomly dropping some units can help avoid the result relying on the specific units of the input. Dropout is a widely used method to avoid overfitting in the training process.

B. Parameters

In this part, some important parameters in the models would be presented.

Epoch: 25
Batch Size: 50
Optimizer: Adam optimizer
Learning rate:

Model	Learning rate
MLP	2e-3
simple CNN	3e-3
weight sharing	2e-3
weight sharing&auxiliary models	3e-3
one-by-one	2e-3

TABLE I: Learning rate

Learning rates for different models are shown in table I.

Activation function: ReLU

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

Loss function:

The aim of this project is classification, so cross-entropy loss is preferred instead of mean square error. The loss function is defined as:

$$loss = -\sum_{k} p(k)log(q(k))$$

Where p and q represent the probability of class k (1 when the sample belongs to class k otherwise 0) and predicted probability of class k relatively.

Pooling function:

Max pooling function is chosen in the model. Max-pooling computes max values over non-overlapping blocks. Its objective is to down-sample an input representation, to reduce the input image dimensionality.

C. Models

In this part, 5 model structures that are used in the project will be introduced.

Multilayer Perceptron (MLP):

Firstly MLP with only linear layers will be used to do this classification task. It is necessary to convert the 2×14×14 picture pair into a 392×1 linear structure. In this process, some information of neighboring pixels may be lost and the final classification may be affected. The MLP structure is shown in Table II.

Layer	Model
1	Linear: 392→200
2	ReLU
3	Linear: 200→50
4	ReLU
5	Linear: 50→20
6	ReLU
7	Linear: 20→2

TABLE II: Structure of MLP

Simple CNN:

The structure of a simple CNN is shown in Table III

Layer	Model
1	Conv2d(2, 32, kernel=5)
2	BatchNorm2d(32)
3	maxpool2d
4	ReLU
5	Conv2d(32,64,kernel=2)
6	BatchNorm2d(64)
7	maxpool2d
8	ReLU
9	Dropout
10	Linear: 256→256
11	ReLU
12	Linear: 256→2

TABLE III: Structure of Simple CNN

CNN with weights sharing:

Similar to the simple CNN, here we try to reform the 2 channels of the image pair and let them take the forward pass separately in order to update the parameters according to 2 channels simultaneously.

Layer	Model	
1	Conv2d(1, 32, kernel=5)	
2	BatchNorm2d(32) 3	maxpool2d
4	ReLU	
5	Conv2d(32,64,kernel=2)	
6	BatchNorm2d(32) 7	maxpool2d
8	ReLU	
9	Linear: 512→128	
10	ReLU	
11	Linear: 128→2	

TABLE IV: Structure of CNN with weights sharing

CNN with weights sharing & auxiliary loss:

Based on the CNN model with weights sharing above, this model is derived by using weights sharing and auxiliary loss (see table V). The loss function is shown below:

$$AuxiliaryLoss = Loss(label, pred\_label) + 0.5 * Loss(num1, pred\_num1) + 0.5 * Loss(num2, pred\_num2)$$

The loss function combines the difference of predicted label of the number pair, and difference of the predicted two numbers.

Layer	Model
1	Conv2d(1, 32, kernel=5)
2	BatchNorm2d(32)
3	maxpool2d
4	ReLU
5	Conv2d(32,64,kernel=2)
6	BatchNorm2d(64)
7	maxpool2d
8	ReLU
9	Linear: 256→128
10	Dropout
11	ReLU
12	Linear: 128→10
13	Linear: 20→200
14	Dropout
15	ReLU
16	Linear: 200→200
17	Dropout
18	ReLU
19	Linear: 200→2

TABLE V: Structure of auxiliary loss CNN

CNN comparing one by one:

Another idea is first to split the image pairs into two images. Then, A CNN model is trained with both 2000 images together to detect the figure in the image. In the end, figure is detected separately and then compared to output the result.

Layer	Model
1	Conv2d(1, 32, kernel=5)
2	maxpool2d
3	ReLU
4	Conv2d(32,64,kernel=2)
5	maxpool2d
6	ReLU
7	Linear: 256→128
8	ReLU
9	dropout
10	Linear: 128→10

TABLE VI: Structure of one-by-one model

III. Result Analysis

In this part, we will analyze our results. Figure 1-4 show the loss and accuracy in different epochs for 4 models separately.

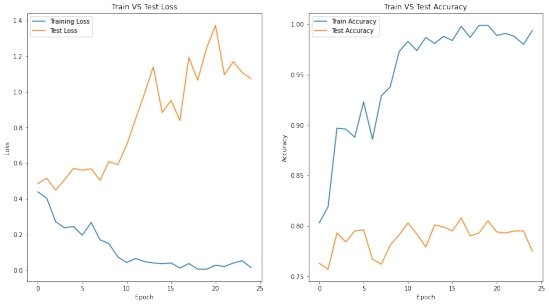

Fig. 1: MLP model

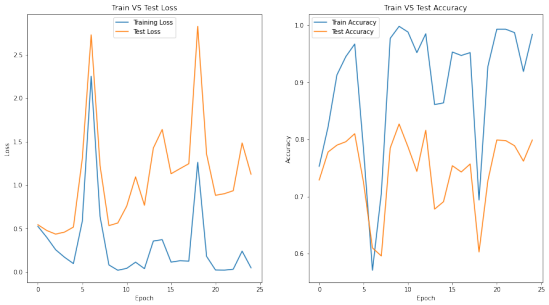

Fig. 2: Simple CNN model

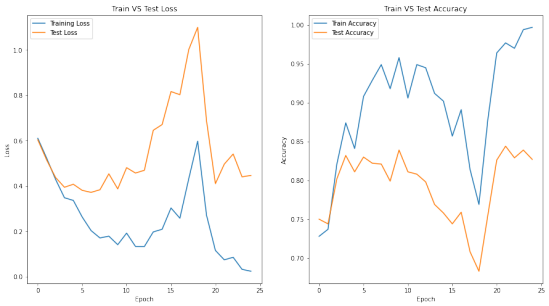

Fig. 3: Weight Sharing model

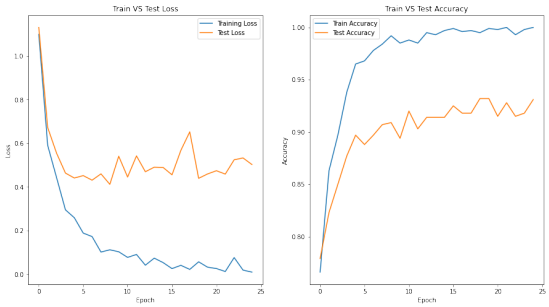

Fig. 4: Auxiliary loss model

It could be seen that in some figures, with the epoch of trainings increasing, and the loss function of the test data set is fluctuating and might increase. It is possible that this is because of overfitting.

We will firstly shed light on the MLP and the model comparing one by one: The result is shown below in Table VII

Model	MLP	One-by-one
Nb of Parameters	89712	43274
Average Train Error Rate	0.16%	13%
Train Error std	0.0025	0.0091
Average Test Error Rate	19.5%	12.54%
Test Error std	0.01	0.0095

TABLE VII: Reults Of MLP & Model comparing one by one

The model comparing one by one has about half of parameters comparing with MLP model. It seems that knowing the mechanism of the classification problem can help transform the problem to get a better result with less parameters.

Then simple CNN, CNN with weights sharing, and CNN with weights sharingauxiliary loss are compared in Table VIII.

Model	simpleCNN	CNN with w	CNN with w&a
Nb of Parameters	76386	75202	88268
Avg. Train Error Rate	1.3%	1.5%	0.2%
Train Error std.	0.0236	0.018	0.0024
Avg. Test Error Rate	19.0%	18.0%	8.5%
Test Error std.	0.0325	0.014	0.0093

TABLE VIII: Reults Of CNN Models

It could be seen that use the weight-sharing model could help reduce the number of parameters. At the same time, we can find that auxiliary loss could help to reduce the average test error rate and average train error rate efficiently. Compared with the comparing-one-by-one model, the model with weight-&auxiliary-loss haw about twice of parameters but has a quite more accurate result.

IV. Conclusion

In conclusion, these models could basically test different architectures to compare two digits visible in two-channel images. And comparing the impact of weight sharing and auxiliary loss to help training. In the future work, more models and structures are needed to be tested, such as the ResNet, different optimizers, to learn about deep learning.