EE-559 Deep Learning - Project 1

Yan Fu[a], Yinan Zhang[a], Zhechen Su[b]
[a]*Institute of Electrical Engineering, EPFL Lausanne, Switzerland*
[b]*Institute of Computer Science and Communication System, EPFL Lausanne, Switzerland*
{yan.fu, yinan.zhang, zhechen.su}@epfl.ch

***Abstract*–The goal of this project is to test several architecture on MNIST hand wrting dataset. Generally, we tested four models: original neural network, neural network with auxiliary loss, siamese neural network and siamese neural network with auxiliary loss. As a result, we reached test error of 0.15 after 25 epoch with siamese model trained with auxiliary loss.**

I. Introduction

In project 1, we used MNIST hand writing dataset to test different deep learning architectures. We generated 1000 pairs of images, for each pair there are two images, and the aim of this task is to predict if number in image 1 is larger than that of image 2 (label 1) or not (label 0).

In this project we tested the impact of weight-sharing and auxiliary loss. Also, to improve the performance of our models, data augmentation and batch normalization are used in all models.

II. Useful Structures

In this part we introduce used structure in our project.

1) **Data Augmentation** We adopted data augmentation in our model mainly to avoid over fitting in training set and also add more samples for training. We rotated the image with random angles and concatenated with original data set, and then shuffled it.
2) **Batch Normalization** We used nn.BatchNorm2d() to implement batch normalization. Simply speaking, batch normalization normalizes layer input for each training batch input, to reduce internal covariate shift. With batch normalization we can consider less on initialization and train our model with larger learning rate.
3) **Siamese Network** To test the effect of weight sharing here we constructed siamese network. For each pair of input data (which contains two images), we applied the same neural network with same parameters, and networks for this two images would be trained simultaneously. In Figure 1, the neural networks on two side share their weights, which helps the model to converge better and greatly reduce the number of trained parameters. And one thing to note, we do not use nn.DropOut() in siamese network. Because the nn.DropOut() cannot guarantee dropping same neurons in siamese network and thus not guarantee weight sharing for networks on two sides. To avoid over fitting we used data augmentation as mentioned before.
4) **Auxiliary loss** And we also introduced auxiliary loss in our final model. 'Auxiliay loss' is the loss that we extracted outputs from shallow layers and compared with certain target. And as researchers [1][2] point out, auxiliary loss helps optimize the learning process and encourage discrimination in the lower stages in the neural classifier. Besides, losses from lower layers can help to reduce gradient vanishing problem. Figure 1 shows how we extracted our auxiliary loss. Output1 and output2 are vectors with size (10,1) and we want the siamese network should first classifier the digits on input images (in this case, the neural network should classify input images as train class '8' and '1'). And the auxiliary loss is cross entropy loss of siamese output with train class. Main output is the prediction of 0 or 1 (input image 1 larger or smaller than input image 2), and the main loss is cross entropy loss of branch output and train target ('0' in this case). Total loss is the sum of auxiliary loss and main loss. Back-propagation will be performed on total loss.

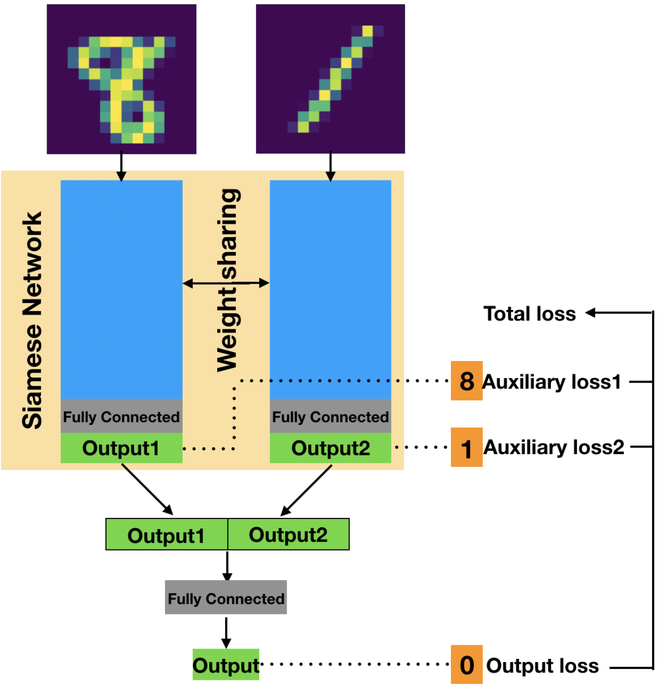

Fig. 1: Siamese Network with auxiliary loss

III. Model description

In our project we compared performance of four architectures:

- Basic neural network (which is used as the baseline)
- Basic neural network with auxiliary loss
- Siamese neural network without auxiliary loss
- Siamese neural network with auxiliary loss

The structures of these models are almost the same, except the siamese network 'double' the basic neural network. So here for simplicity, we only show the structure of siamese network in following table.

Layer	Structure	Parameters	Activation
1	Convolution Layer	(1,16,2)	
	Batch Normalization	16	
	Max Pooling	2	ReLU
2	Convolution Layer	(16,64,2)	
	Batch Normalization	64	
	Max Pooling	2	ReLU
3	Convolution Layer	(64,128,2)	
	Batch Normalization	64	
	Max Pooling	2	ReLU
4	Linear Layer	(128 *2 * 2, 64)	
5	Linear Layer	(64,10)	
6	Linear Layer	(20,2)	

TABLE I: Result of disambiguation

Total number of trainable parameters is 71076.

IV. Training

All models were trained under same conditions:

- batch size: 100
- epoch: 25
- learning rate: 0.001
- optimizer: Adam optimizer

And for each model the performance was tested through 20 rounds on splited test set. The mean and standard deviation test error of each model will be used as final evaluation.

V. Result

Table 2 shows the final result. We can see that both siamese net and auxiliary loss can help improve the performance.

Epoch	Model	No. of para	Test error
25	Base line	71140	0.22±0.08
25	Base line with AL	71140	0.20±0.07
25	Siamese Net	71076	0.17±0.05
25	Siamese Net with AL	71076	0.15±0.05

TABLE II: Comparison of four models

And we use T-SNE to plot the final layer of these architectures on 2D and label each point with input digit number. We can see that the base line model separates digits in two clusters, one with larger digits like 7,8,9 and the other one with smaller digits like 0,1,2,3 in general. It seems that this model learns the numeric order. And for Siamese network without auxiliary loss, we can see that the classification of digits is more reliable, and still it learns numeric order (from bottom to top the values of digits grow) but the boundaries of clusters of digits is ambiguous.

From figure 3 and figure 5, we can see the introduced auxiliary loss can somehow 'force' the last layer to predict digits and siamese network has better prediction. However, numeric order disappears in visualization of models trained with auxiliary loss.

In short, siamese network improves the performance because weight sharing reduce the number of parameters for models to obtain same performance; and auxiliary loss helps in that it can encourage classification in lower layers. And the model logic using or not using auxiliary loss seems vary: with auxiliary loss, the model first try to predict the digits on the image, then compare their values, while without auxiliary loss, it seems the model learns the numeric order of digits, as we can see big numbers in one side and small ones in the other side.

References

[1] Szegedy C, Liu W, Jia Y, et al. Going deeper with convolutions[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2015: 1-9.

[2] Zhao H, Shi J, Qi X, et al. Pyramid scene parsing network[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 2881-2890.

[3] Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift[J]. arXiv preprint arXiv:1502.03167, 2015.

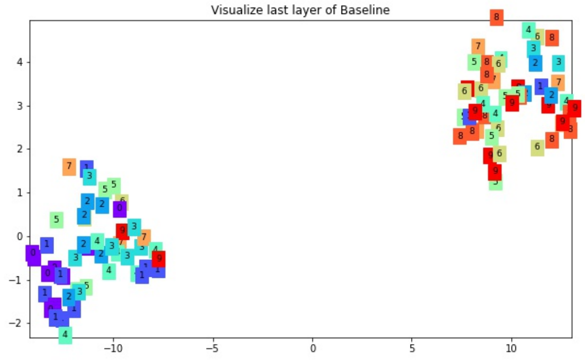

Fig. 2: Last layer of Baseline model

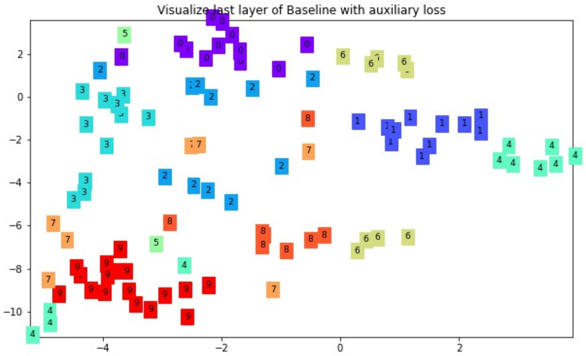

Fig. 3: Last layer of Baseline model with auxiliary loss

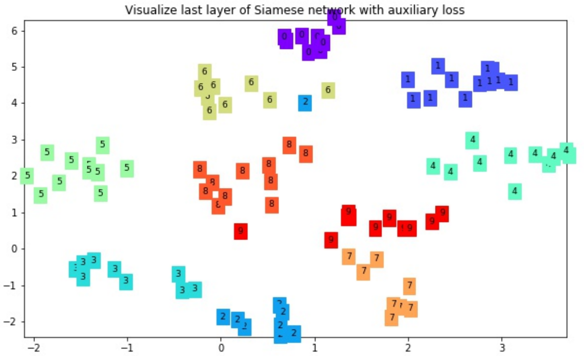

Fig. 5: Last layer of Siamese Network with auxiliary loss

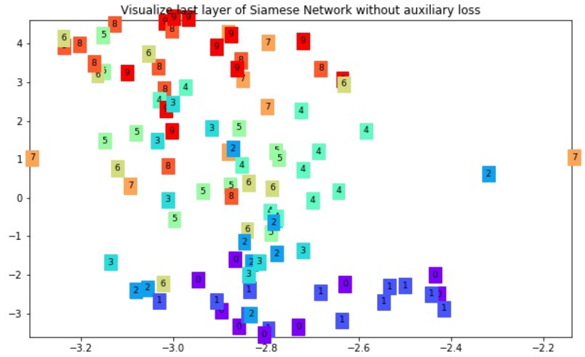

Fig. 4: Last layer of Siamese Network