Deep Learning Project1 Report

Chenkai Wang
323452
chenkai.wang@epfl.ch

Siran Li
321825
siran.li@epfl.ch

Shijian Xu
327448
shijian.xu@epfl.ch

***Abstract*—In this project, we aim to test different architectures to compare two digits visible in a two-channel image. We use two main architectures: Multi-Layer Perceptron (MLP) and Convolutional Neural Network (CNN). For each type of the models, the effect of the detailed structure choices, like activation types, dropout rates, etc., are investigated. Besides, we also assess the performance improvement that can be achieved by incorporating weight sharing and auxiliary loss.**

I. Introduction

This project aims to implement a deep network that can be used to predict for a two-channel digits image that if the first digit is lesser or equal to the second. More specifically, the data are generated from MNIST by first downsampling the image size and then concatenate two digit images together to form a $2 \times 14 \times 14$ tensor. The label is 0 or 1, indicating whether the first digit is larger, which results in a binary classification task.

We choose two types of deep networks: MLP and CNN. For each model, we first search for the best architecture via extensive experiments and determine the best architecture configurations. Then, based on the selected best architecture, we evaluate the effect of different activation functions, dropout rates and batch normalization on the final performance. We then make decisions whether to apply these tricks to the final models based on the evaluation results. After that, we test the performance improvements achieved by using weight sharing and/or auxiliary loss.

II. Experiment Design

A. The Basic Structure

To simplify the architecture searching space, we pre-define that the MLP feature extraction part consists of 4 linear layers. The classifier parts of the MLP are composed of 1 linear layer. Activation layers are added at appropriate positions. For CNN, we pre-define that the ConvNet feature extraction part consists of 3 convolutional layers. The classifier parts of the ConvNet are composed of an MLP with 1 hidden layer. Max pooling layers are used in the feature extractor. The detailed structures are listed in Table I.

Module	MLP	ConvNet
Feature Extractor	Linear ReLU Linear ReLU Linear ReLU Linear ReLU	Conv2d ReLU MaxPool2d Conv2d ReLU MaxPool2d Conv2d ReLU MaxPool2d
Binary Classifier	Linear Sigmoid	Linear ReLU Linear Sigmoid
Digit Classifier	Linear	Linear ReLU Linear

TABLE I: Basic Structures of the models.

B. Experiment Protocol

We conduct the experiments following the below experiment protocol. First, we define the vanilla MLP and vanilla ConvNet. The vanilla MLP uses *ReLU* activation function. No *dropout* layer is used. Besides, we do not incorporate the weight sharing and auxiliary loss into this model. Similar to the vanilla MLP, the vanilla ConvNet uses *ReLU* activation function. No *BatchNorm* layer is used. No weight sharing and auxiliary loss. The illustration of these vanilla models are shown in Fig. 2.

Based on the vanilla models, the first experiment is searching for the best architectures. Since we have pre-defined the depth of these models, what we need to search are **hidden layer dimensions** for MLP and **channel numbers** for ConvNet. We determine the best architecture based on the test accuracy. Second, we do experiments to determine the best activation function. The candidate activation functionsa are *ReLU*, *Tanh*, *Sigmoid* and *SELU*. After determining the activation function, we then evaluate the effects of different dropout rates for MLP. For ConvNet, we evaluate the effect of adding *BatchNorm*.

After determining the network structure via the above experiments, we assess the performance achievements by using weight sharing and auxiliary loss. The network structure with weight sharing and auxiliary loss is shown in Fig. 3.

III. Experiment Results

A. Searching for the best architecture

In this part, we optimize the network parameters to obtain the possible optimal architecture. We achieve a list of all possible combinations from the values in [16, 32, 64, 128, 256, 512], which yields the number of trainable parameters between 65000 and 75000. For each of these sets of parameters, we train the corresponding network (MLP and CNN) 10 times. During the training, the learning rate is fixed to 0.001 and the batch size is 100. The searching results are shown in Fig. 1.

From the results, we can see that the combination [128, 64, 32, 256] achieves the best performance for MLP and the

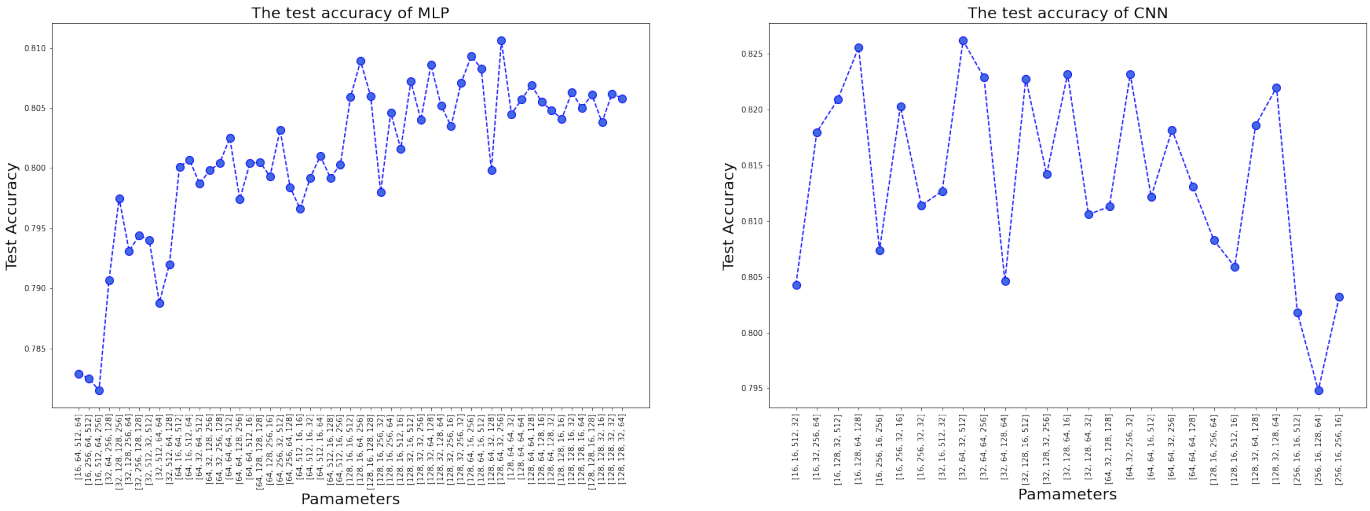


Fig. 1: The test accuracy of MLP and CNN under different parameters.

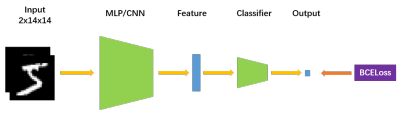


Fig. 2: The vanilla network structure.

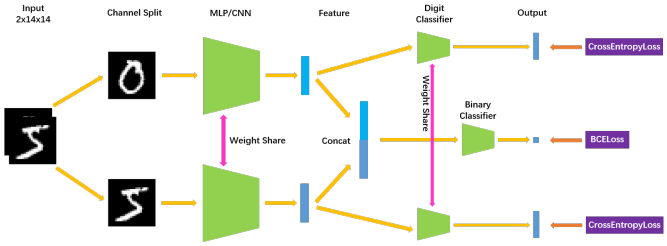
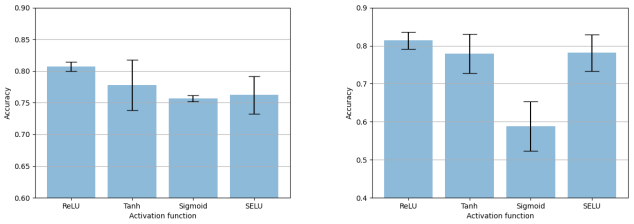


Fig. 3: The network structure with weight sharing and auxiliary losses.

combination [32, 64, 32, 512] achieves the best performance for ConvNet.



(a) MLP (b) ConvNet

Fig. 4: The accuracy of choosing different activation functions.

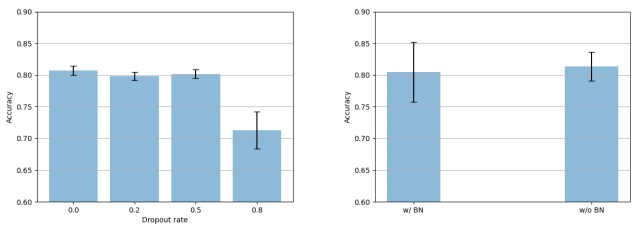
B. Searching for the best activation function

Based on the results in III-A, we continue to search for the best activation function. The candidate activation functions are *ReLU*, *Tanh*, *Sigmoid* and *SELU*. For MLP, the hidden layer unit numbers are set as: 128, 64, 32, 256. For ConvNet, the channel numbers for each conv layer are set as: 32, 64, 32, and the hidden dimension for the classifier is 512. We test for 10 rounds for each activation function choice, and the results are shown in Fig. 4. From the results we can see that for both MLP and ConvNet, the *ReLU* activation function always achieves the best performance while the *Sigmoid* activation function always produces the worst performance. Is this reasonable? In the following experiments, we will stick to use *ReLU*.

C. Effect of Dropout and BatchNorm

Dropout is usually applied on fully-connected layers, but seldom be used on convolutional layers. why? So, in this section, we only evaluate the effect of adding dropout layers to MLP. For ConvNet, we evaluate another trick, which is Batch Normalization.

To evaluate the effect of dropout layer, we test with different dropout rates, which are 0, 0.2, 0.5 and 0.8. Each choice is evaluated for 10 rounds. The results are shown in Fig. 5.



(a) MLP-Dropout (b) ConvNet-BatchNorm

Fig. 5: The accuracy of applying Dropout to MLP and BatchNorm to ConvNet.

It is strange to see adding Dropout to MLP and BatchNorm to ConvNet do not improve the performance. The best performances are achieved when no Dropout and no BatchNorm is

applied. We hypothesize that it might be because the shallow models are not overfitting and why???

D. Effect of Weight sharing and Auxiliary loss

From the above experiments, we have came to the conclusion that we should use ReLU activation and should not apply Dropout or BatchNorm to these models. Based on these observations, we will evaluate the effect of weight sharing and auxiliary loss.

Weight sharing is simply split the input 2-channel image into two 1-channel images and then process them separately with the same feature extractor. The extracted two features are then concatenated together to pass through the classifier, as shown in Fig. 3. The results are shown in Fig. 6. From this figure, we can see that weight sharing improves the performance greatly for both MLP and ConvNet.

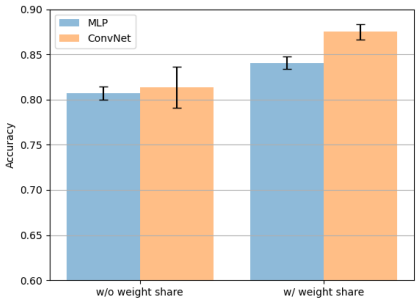


Fig. 6: The effect of weight sharing for MLP and ConvNet.

We have seen that weight sharing is very useful. So we evaluate the effect of auxiliary loss based on weight sharing. The idea is, the extracted two features are passed through the digit classifier separately to predict the corresponding digit labels (0-9), and the concatenated features are passed through the binary classifier to predict the binary label (0/1). In total it will produce 3 losses and we add these losses together for backward propagation. Apart from naively adding 3 losses together, we also evaluate the trade-off between the binary classification loss and the digit prediction loss. So the final loss is: $loss = loss_{digit1} + loss_{digit2} + \lambda loss_{binary}$, where λ ranges from 0.5 to 1.5, with step size 0.1. The results are shown in Fig. 7.

From this figure, we can see that auxiliary loss further improves the performance. For MLP, the best performance is achieved when $\lambda = 0.9$. For ConvNet, the best performance is achieved when $\lambda = 0.8$.

IV. Conclusion

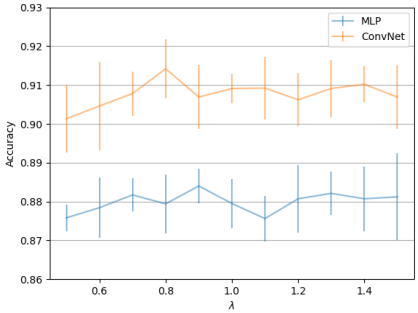


Fig. 7: The effect of auxiliary loss for MLP and ConvNet.