EE559 Project 1: Classification, Weight Sharing, Auxiliary Losses

Lingjing Kong, Jiahua Wu, Phan Huy Thong
Ecole Polytechnique Federale de Lausanne, Switzerland

***Abstract*—We report a binary classification model with weight sharing and auxiliary losses, which produces an average test accuracy of 95.2%. In this report, we fully unfold our exploratory steps and reasoning behind them with an emphasis on the benefits of weight sharing and auxiliary losses.**

I. Problem Statement

In this project, we are tasked to create a neural network which is able to predict for pairs of digit images whether the first digit is equal or less than the second. The core part of this project is to evaluate and discuss the effects of weight sharing and auxiliary losses.

II. Overview

Convolutional neural nets (CNN) are well-suited for image classification for its invariance to translation and 'only local interaction matters' inductive biases. In section IV, we built several baselines: models with only fully connected layers, models with only convolutional layers and a conventional CNN with both types of layers. Here we studied the impacts of convolutional channel number, fully-connected layer feature number and the number of both layers.

In section V, we split each input into two 1-channel images. Since the two images in each original input are different objects, they do not carry the meaning of channels, i.e. different views of the same object. The similarity between two inputs allows us to create our weight sharing model (Figure 2(a)) where the weights of the convolutional block are shared by the two input paths. We created another model (Figure 2(b)) which processes two images with un-shared convolutional block weights in order to better understand the effects of splitting the 2-channel inputs and the benefits of the weight sharing scheme.

In section VI we added an auxiliary task of classifying each digit to our CNN in order to utilize the information stored in the training data class labels. To gain a better understanding, we experimented with various loss_ratio (main_task_loss/auxiliary_task_loss) and plotted run-time loss curves for the two tasks.

In section VII, we compared models with different combinations of auxiliary loss and weight sharing to evaluate the effects of each. Also, we experimented with dropout rates to counter the observed overfitting. Finally, our conclusion is drawn in VIII.

III. Notations and Preliminaries

Data Pre-processing Before any training and testing, we normalized our training input by subtracting its mean and dividing by its standard deviation. We normalized our test input with the training data mean and deviation.

General Setup We decided to use Binary Cross-Entropy Loss for the binary target and Cross Entropy Loss for the auxiliary classifier regarding digit classification. Activation functions in hidden layers were defaulted as rectified linear units (ReLU), for its overwhelming popularity in the deep learning community. Adam optimizer with learning rate of 10^{-3} has been used in our experiments for its robustness to learning rate tuning.

Validation During tuning, we performed 4-fold cross-validations to check the mean and standard deviation (std) of the cross-validation accuracy. During testing, ours model were trained with the full training data for 25 epochs and were tested on the test data. We repeated the test by 10 times with randomized model initialization and data import to ensure the fairness.

Notations In order to demonstrate the network structure compactly, we adopt notations. Convolutional layers are denoted as conv(in_channel, out_channel, kernel_size), fully-connected layers are denoted as fc(input_size, output_size), P denotes max-pooling. ReLU activation is used uniformly as mentioned as above and is not noted in the structures. For convolutional layers, padding is set accordingly to preserve the spatial size. We use max-pooling to reduce the image size.

IV. Baselines

A. Only Fully-connected Nets and Only Convolutional Nets

Typically image classification tasks are carried out by CNNs with both conv-layers and fully-connected layers. To measure the advantage of the conventional CNN structure, we built models with only one type of layers. We tuned layer parameters and model depth to gain understanding of their impacts on the model performance. Later we designed a CNN baseline model with 3 conv-layers and 3 fully-connected layers as the starting point of our CNN models (Figure 1). Model performance figures are shown in I.

For models with only fully-connect layers, increasing the layer number alone could not improve the performance (fc_1 to fc_2), however, a significant increase of the feature number (fc_1 to fc_3) on hidden layers boosted the performance by more than 3% but the training load is too heavy to carry

Net ID	Structure	Number of parameters	Mean validation accuracy	Standard deviation
fc_1	Fc(n[1],n/2) Fc(n/2,n/4) Fc(n/4,1)	96,433	0.765	0.007
fc_2	Fc(n,n/2) Fc(n/2,n/2) Fc(n/2,n/4) Fc(n/4,n/4) Fc(n/4,1)	144,747	0.763	0.004
fc_3	Fc(n,2n) Fc(2n,2n) Fc(2n,1)	924,337	0.811	0.003
conv_1	conv(2,10,3) P(2) conv(10,20,3) conv(20,1,3)	2,571	0.785	0.01
conv_2	conv(2,32,3) P(2) conv(32,64,3) conv(64,1,3)	48,851	0.801	0.009
conv_3	conv(2,10,3) P(2) conv(10,20,3) conv(20,40,3) conv(40,1,2)	9,411	0.804	0.006
cnn_base	Figure 1	105,889	0.826	0.011

TABLE I

BASELINES: MODELS WITH ONLY FC LAYERS AND MODELS WITH ONLY CONVOLUTIONAL LAYERS AND A COMBINATION CNN BASELINE; CONVOLUTIONAL LAYERS ARE DENOTED AS CONV(IN_CHANNEL, OUT_CHANNEL, KERNEL_SIZE), FULLY-CONNECTED LAYERS ARE DENOTED AS FC(INPUT_SIZE, OUTPUT_SIZE), P DENOTES MAX-POOLING

Fig. 1. CNN Baseline: our first baseline model with both conv layers and fc layers

(924,337 parameters). For models with only convolutional layers, we can observe both the increase of channel number (conv_1 to conv_2) and the increase in depth (conv_1 to conv_3) matter.

Notably to achieve similar performances (fc_1 c.f. conv_1 and fc_3 c.f. conv_3), fc models contain much more parameters than conv models, which illustrates the fact that convolutional layers are more suitable in image classification. With similar performance, conv_3 only uses merely one fifth of the parameter number of conv_2. Thus, in the design of CNN models, increasing the convolution block depth might be more advantageous than only increasing channel number. A further progress of 2% is achieved by our baseline CNN model from conv_3 and fc_3 with 105,889 parameters.

V. WEIGHT SHARING

As mentioned in II, it is sensible to split each input into two images as they are representations of two different objects.

[1] n denotes the flattened size of one input i.e. $2 \times 14 \times 14$

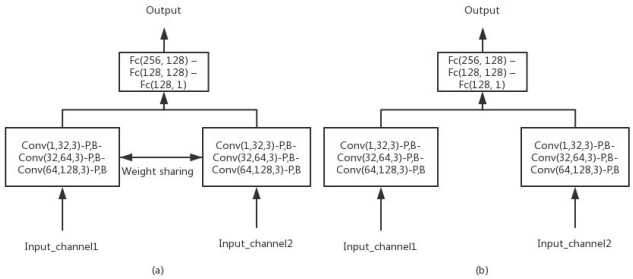

Fig. 2. Left:CNN with weight sharing; right: CNN w/o weight sharing

Further, digit images have very similar formats and thus two images can share one feature extractor. Following the idea of weight sharing, we designed our weight sharing network w_net (Figure 2(a)), where the two images are processed by the same CNN block and then get combined and fed into the fully-connect block (classifier). Here we also created an equivalent model without weight sharing (Figure 2(b)), in order to distinguish the impact of splitting input and the impact of weight sharing.

From Table III, we can observe that to accommodate the two separate input images, the number of parameters gets doubled without weight sharing (from base_cnn to no_w_net; please note apart from modifying and duplicating the conv block in base_cnn, we also had to modify the fully-connect block to match the size change). But the sensible processing of inputs indeed brings about a progress of 2% in terms of validation accuracy.

Notably, the advantage of weight sharing scheme is significant. From no_w_net to w_net, the parameter number decreases almost by half and the validation accuracy rises by around 3%. The performance progress can be explained as: with fewer parameters to train, the demand for data is reduced and the likelihood of encountering overfitting is lowered. Intuitively, by sharing weights of the conv block, we represent the two images by vectors with the same semantics, which eases the classification later. Compared with cnn_base, w_net achieves almost 5% increase in validation accuracy with less than 1.5 times of parameters.

VI. AUXILIARY LOSSES

By introducing a more challenging auxiliary task (classifying the digit in each image), information conveyed in the given data class labels can be exploited to enhance the model performance in relevant tasks (determining which digit is larger).

During training, an extra classifier is attached to the feature extraction CNN block of our w_net, with a soft-max output of 10 classes. This is due to the understanding that the CNN block acts as an feature extractor, and will learn more meaningful features with the extra information in the digit class labels. The two classifiers are split as they perform different classification tasks. Note for each input data, we have two input images, therefore weight sharing can be applied here again. The same auxiliary classifier is used to process the two images and their

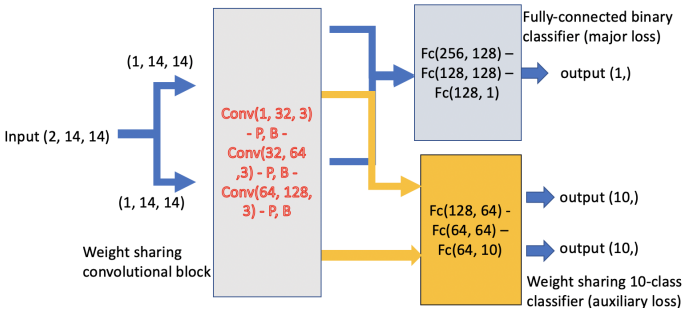

Fig. 3. Net with Auxiliary Losses: grey block on the left is the weight sharing conv block; grey block on the top left is the binary classifier (main loss); the yellow block is the weight sharing auxiliary loss classifier.

Fig. 4. Run-time for a_w_net with loss ratio of 1: here we can observe the net learns to classify digit classes; also for the main task, overfitting is severe.

classification losses are summed as the term L_{aux}. The total auxiliary loss is added to the weighted main task loss i.e. $L = \text{loss\_ratio} \times L_{\text{main}} + L_{\text{aux}}$. The model diagram is shown in Figure 3. At the validation/testing stage, this extra classifier is disregarded.

We plotted the run-time validation loss for the model with both weight sharing and auxiliary loss (see Figure 4). With the initial loss_ratio as 1, validation accuracy rises by almost 4% (Table II) compared with w_net on our main binary classification task, which confirms our intuition mentioned above. The increase on parameter number is insignificant. In order to compare the effects between auxiliary losses and weight sharing, we also incorporated the auxiliary loss part into our no_w_net (baseline model without weight sharing) in V, which results in our model a_net.

We explored different loss_ratio and recorded corresponding validation accuracy in II. The optimal one is taken on in our a_w_net, with its performance figure shown in III.

loss_ratio	Mean validation accuracy	Standard deviation
2	0.910	0.005
1.5	0.907	0.007
1	0.913	0.011
0.5	0.936	0.0019
0.1	0.904	0.009

TABLE II
VALIDATION ACCURACY AND LOSS RATIOS

VII. Evaluation and Discussion

In this section, we compare and discuss the effects of auxiliary loss and weight sharing. In TableIII we summarize performance figures of all our main models.

Model	Description	Number of parameters	Mean validation accuracy	Standard Deviation
cnn_base	baseline (w/o splitting input)	105,889	0.826	0.011
no_w_net	baseline	235,777	0.847	0.009
w_net	baseline + weight sharing	142,658	0.874	0.006
a_net	baseline + aux. loss	248,843	0.913	0.010
a_w_net	baseline + both	155,275	0.936	0.0019
final_model	a_w_net with dropout	155,275	0.952	0.007

TABLE III
MAIN RESULTS: ALL THE EXPERIMENTS WERE RUN BY 10 TIMES WITH RANDOMIZED INITIALIZATION AND DATA

From Table III we can observe that the impressive performance enhancement brought about by the auxiliary loss part. Specifically, auxiliary loss alone increases the validation accuracy from 0.847 (no_w_net) to 0.913 (a_net) with minimal increase of parameters. On the weight sharing net, the improvement from 0.874 (w_net) to 0.936 (a_w_net) is also considerable. The performance enhancement from weight sharing is also noticeable: it alone increases the accuracy figure from 0.847 to 0.874 on the baseline net; on the net with auxiliary loss (a_net), it elevates the accuracy score from 0.913 to 0.936. Despite of the smaller score increases compared with that of the auxiliary loss, the major benefit from weight sharing is the great reduction on the number of parameters (from no_w_net to w_net, 39.49% reduction; from a_net to a_w_net, 37.6% reduction), which imposes a lesser demand on the training data and alleviates the training load.

The 'U' shape of validation loss in Figure 4 is a clear sign of overfitting, which is commonly mitigated by dropout layers. As our model is shallow we only added one dropout layer to the last CNN layer in a_w_net and tested with different dropout rates. With the optimal dropout rate (0.4), we obtained our final_model in Table III.

VIII. Conclusion

In this project we investigated the effects of weight sharing and auxiliary losses (section VII. In addition, we also studied impacts of the number of parameters on each layer, the depth of neural networks in section IV. If more time and computation resources are available, we would like to tune the learning rate and try different optimizers. Also, we could try to extend the weight sharing to the fully-connected layer block to further reduce the number of parameters.