EE-559 Project 1 – Classification, weight sharing, auxiliary losses

Barış Sevilmiş Furkan Karakaş Sena Necla Çetin
School of Computer and Communication Sciences, EPFL, Switzerland
{baris.sevilmis, furkan.karakas, sena.cetin}@epfl.ch

***Abstract*—In this first project of EE-559 Deep Learning course at EPFL, we test the performances of different architectures in an image classification task. The aim of this project is to determine whether the digit given in the first image is less than or equal to the digit given in the second image. We experimented with using a fully connected dense network, a convolutional network, and an auxiliary network. The best validation accuracy was obtained using an auxiliary network, yielding an accuracy of $97.93 \pm 0.90\%$.**

I. Introduction

The aim of this project is to compare the performances of different neural network architectures using weight sharing or auxiliary losses. In the following, we start by describing the dataset (Section II). Then, we explain the models and methods that we used in our architectures (Section III). In Section IV, we discuss the results that we obtain from hyperparameter tuning in different models that we use. Finally, we finish the report with concluding remarks and discussions (Section V).

II. Data description

The MNIST dataset is an image dataset that is commonly used in image classification tasks, which contains 14×14 gray scale handwritten digit images [1]. The subset of data we use in this project contains 1,000 pairs of images in the training and test sets each. The input is given as a series of $2 \times 14 \times 14$ tensor, where the first channel of the tensor contains the image of the first hand-written digit and the second channel of the tensor contains the image of the second hand-written digit. We use the given prologue file to load the train and test datasets, which involve train input ($2 \times 14 \times 14$), train target (1), and train classes (2).

III. Models & Methods

Three different model architectures have been used for this project. Namely, a fully connected dense network (FCN), a convolutional network (CNN), and a convolutional auxiliary network (AUXN) have been chosen. Each network is fine-tuned with three different hyperparameters, namely "learning rate", "weight decay" and "auxiliary parameter". Weight decay is an L2 penalization factor for the weights in the network. In a sense, this parameter adds the weights of the parameters in the network to the loss function, which is the basic idea of L2 regularization. Auxiliary parameter is the coefficient to combine the classification error of true interest and auxiliary losses from the digits in channel 1 and 2. Note that this hyperparameter is not tuned in the first two networks FCN and CNN. The results of hyperparameter tuning is shown in Table I. We run each model for hyperparameter tuning in 15 rounds. We also randomize the data generation so that each round yields different results and gives a less biased result. The networks are explained individually in next set of subsections. All weight parameters of convolutional and dense layers in the following architectures are initialized with Xavier initialization. Adam optimizer is utilized with step-wise learning rate scheduler. Scheduler would decrease learning rate by gamma (0.1) every 15 epoch multiplicatively. Loss criterion is chosen as cross entropy loss. As the number of trainable parameters, we have **142,786** in FC, **52,450** in CNN, and **60,396** in AUX, respectively.

A. DenseNet

First proposed network is a network that consists of fully connected layers, dropout layers and 1-dimensional batch normalization layers. Therefore, image is flattened out before it enters network where $2 \times 14 \times 14$ shape is transformed to 1×392 shape. This input is forwarded into the architecture 1, in which all the activation functions are ReLu. First set of layers consist of an input dense layer with 256 neurons followed by batch normalization layer and lastly, a dropout layer with probability equal to 0.2. Next set of layers consist of a hidden layer with 128 neurons followed by a 1-d batch normalization layer and lastly, a dropout layer with probability of 0.2 as well. Last set of blocks comprises a fully connected layer with 64 neurons, followed by a 1-d batch normalization layer and lastly, output is created from a fully connected layer with 2 neurons and softmax activation function.

B. ConvNet

Second proposed network is a convolutional network with a fully connected ending. Convolutional part of network is formed from 2-d convolution layers with 3×3 kernels and ReLu activations followed by 2-d batch normalization layers and dropout layers with probability of 0.2. There are such 2 sequential building blocks with only difference in convolutional layer channel sizes. First convolutional layer produces 32 channels and second convolutional layer ensures 64 channels. After the second dropout layer, there exists a hidden fully connected layer with 128 neurons and ReLu activation. Last, layer is a dense layer with 2 neurons and softmax activation which produces the output. Outputs of ReLu activations from convolutional layers are forwarded into 2-d max pooling layers with kernel size of 2×2 and stride 2. Figure 2 demonstrates the architecture.

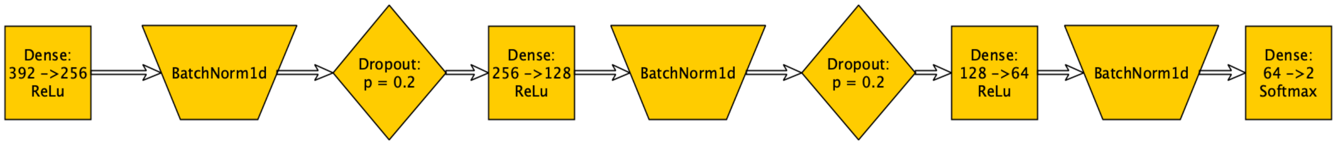


Fig. 1: Fully Connected Dense Network

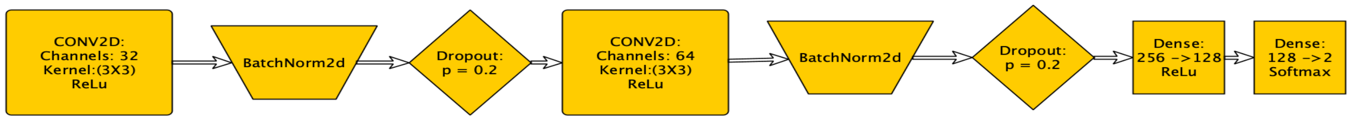


Fig. 2: Convolutional Network

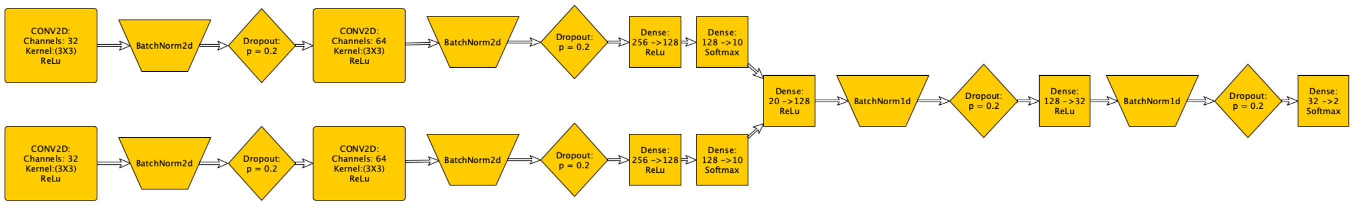


Fig. 3: Auxiliary Network with weight sharing and auxiliary losses from digits

C. *AuxiliaryNet*

Third network architecture consists of 2 main components. First component is convolutional network, in which actual classification of both training images occurs. Convolutional network architecture is the same as the previous convolutional network, but only final layer has 10 neurons with softmax activation outputting the actual class labels of images. Resulting 10×1 vectors from convolutional networks are stacked together and are forwarded to a small dense network, where the input is the probability class distributions of both images. Dense network starts with a dense layer of 128 neurons and ReLu activation followed by 1-d batch normalization and dropout layer of probability 0.2. This structure is followed by another dense layer with 32 neurons and again 1-d batch normalization and dropout layer of probability 0.2. Output layer is a dense layer with 2 neurons and softmax activation. This model utilizes auxiliary losses as the name indicates. In other words, softmax output of convolutional networks is not only used as stacked inputs to dense network. These digits are compared with their actual class labels, and 2 additional loss values are calculated. These loss values are used in total loss calculation, where weighted loss summation takes place for a total of 3 losses (2 for digit classification and 1 for comparison of digits). Loss is computed by the following equation:

$$total\_loss = \sum_{i=1}^{3} w_i loss_i \text{ with } \sum_{i=1}^{3} w_i = 1 \tag{1}$$

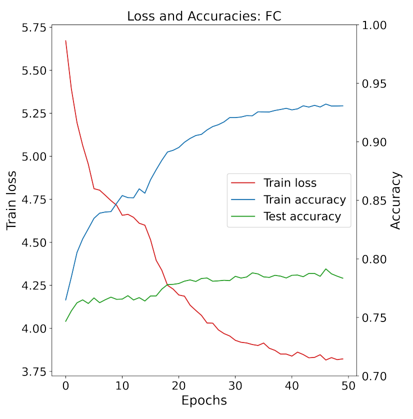
IV. Results

As we expected, we got the worst validation accuracy of 78.89% in the fully connected network architecture. The results are shown in Figure 4a. As we see in the figure, there is overfitting in the fully connected network since there is about 20% difference in the accuracies of the train and validation sets. We expected this outcome since the network does not utilize the structure of the images by applying convolutional filters to the images. Since the dataset involves images, it is inherently more suitable to use convolutional layers than fully connected layers in the network. We got the second best validation score of 85.01% in the fully convolutional neural network, which is a huge improvement in comparison to the previous network architecture. The results are shown in Figure 4b.

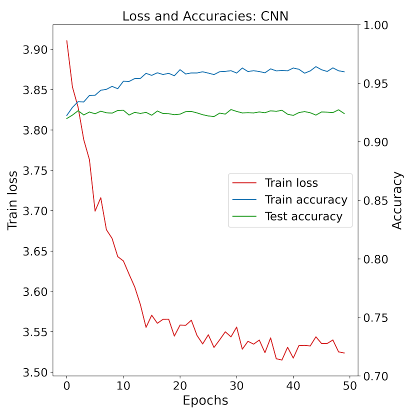
TABLE I: The best fine-tuned hyperparameters for different network architectures

	Learning rate	Weight decay	Aux param	Accuracy
FCN	5e-2	1e-4	N/A	$78.89 \pm 1.83\%$
CNN	5e-4	1e-3	N/A	$85.01 \pm 2.97\%$
AUXN	5e-3	1e-4	2e-1	$97.93 \pm 0.90\%$

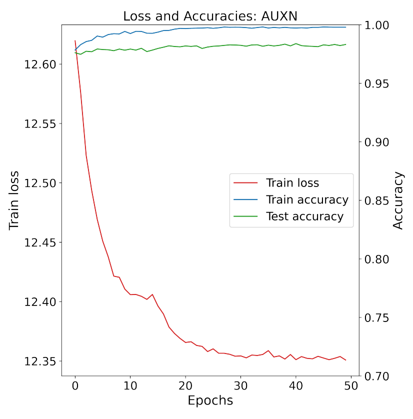
We get the best result (97.93% accuracy) with the auxiliary network, which is a combination of fully connected and convolutional neural networks. Furthermore, we use the auxiliary losses from the first and second digits in the input image on top of the classification loss of truly interest. We combine each of these losses with a hyperparameter called `AUX_PARAM`. The results are shown in Figure 4c.



(a) Fully Connected Network



(b) Convolutional Network



(c) Auxiliary Network

Fig. 4: Train losses, train accuracy and validation accuracy in different network architectures

Figure 5 contains a misclassified sample of two digits in our model. The model predicts that the first digit is 7 and the second digit is 5. As a result, the classification output is 0 since 7 is **not** less than or equal to 5. However, we note that the first digit is specified as 4 in the test class.

The validation accuracy scores for three models are plotted for 15 rounds in Figure 6. The corresponding mean and standard deviation values are shown in Table I. We realize that the standard deviation between rounds for the third model is the smallest among all models. So, we can conclude that the third model is the most stable one and it gives the most accurate results among these.

```
Output: 0
Softmax result class 1: 0.9994872808456421
Softmax result class 2: 0.0005126874893903732
Test target: 1
Test class 1: 4
Test class 2: 5
Digit 1: 7
Digit 2: 5
```

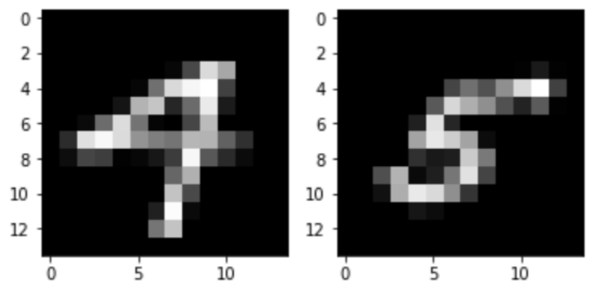


Fig. 5: A misclassified test sample of two digits

V. Discussion & Conclusion

We were fairly surprised that the auxiliary network worked so well with such a small amount of data. Normally, MNIST datasets with image sizes 28 $\times$ 28 are trained with 60,000 samples and tested with 10,000 samples to obtain an accuracy level of 99%. On the other hand, our task is not the same as detecting the digit from 0 to 9 but we only did a comparison between first and second digits in a given input. Our initial thoughts about this project were proven with the results that we obtained. We predicted that the third model would outperform the first two ones.

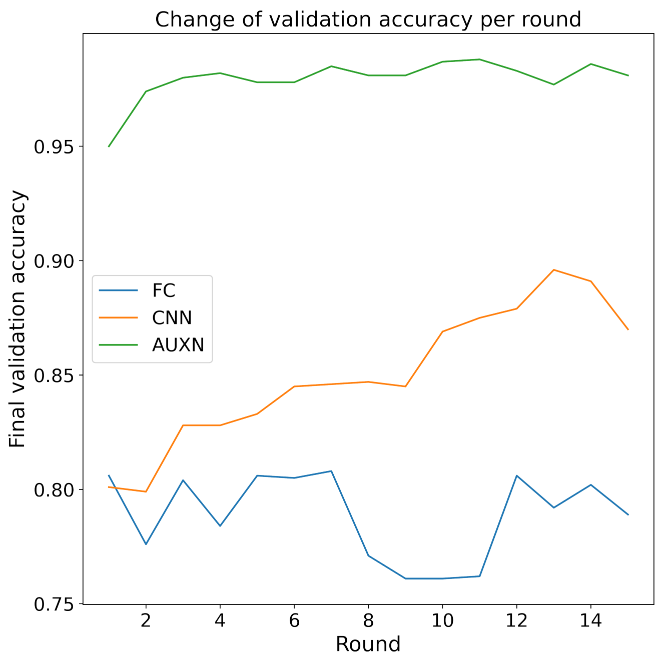


Fig. 6: Change in validation accuracy scores w.r.t. rounds for each model

References

[1] LeCun, Y., Cortes, C., & Burges, C. *The MNIST database of handwritten digits.* http://yann.lecun.com/exdb/mnist/.