EPFL

Deep Learning – EE559 Influence of Weight Sharing and Auxiliary Loss in Neural Networks for Image Recognition

Théophile Bian Mathias Dankl
Samuele Mercan

May 17, 2019

1 Introduction

The construction and development of deep neural networks (dNNs), has led to the solving of a vast range of common categorization tasks with varying complexity, by means of artificial intelligence. Applications range from simple 1D categorization tasks, to more complex tasks such as audio and image recognition [1, 2, 3, 4]. Along with their success in solving everydays classification tasks, challenges still remain in terms of the design and the training of a NN. A-priori it is not clear how to design and train a NN in order to solve a certain categorization task. Instead, one is confronted with a high-dimensional (hyper)-parameter space (hPS) from which to choose and benchmarking studies become a necessity in order to find a suitable NN. Clearly, such studies are limited by the available computational resources and time. With an increasing number of NN parameters to tune, the latter becomes the bottleneck and one is confined to focus on a smaller subspace of the hPS. For that, different techniques have been developed, two of which – namely, weight-sharing (w.s.) and the addition of an auxiliary loss (aux. loss) – will be explored in more detail in this work.
For studying the influence of weight-sharing and auxiliary losses, the following image classification task is considered: given a pair of 14×14 grayscale images containing digits from 0 - 9, a deep NN should be designed, which predicts on whether the first digit is less than the second one. For the design, different architectures, ranging from simple 1D (deep) neural networks up to a combination of convolutional 2D and conventional 1D layers, will be exploited.
For that, a benchmarking framework, which will be discussed in more detail in the following section, is implemented, allowing an efficient systematic scanning of the NNs' (hyper)-parameter space.

2 Methods

2.1 Implementation

In order to allow an efficient systematic benchmarking of the (hyper)-parameter space, a framework is implemented in Python, which trains and tests different architectures. In general, the framework follows the notion of obtaining/writing any input/output from/to corresponding files. The implementation is structured in such a way that extensions (adding different losses, optimizers, ...) may easily be done in the future if needed.
Specifically, the framework implements for the purpose of this work regular/convolutional 1D/2D layers, as well as a selection of common loss functions and optimizers. In addition to that, the following features are implemented in order to facilitate the study of the influence of weight-sharing and auxiliary losses:

- Setting an arbitrary number of sub-networks, all of which are combined at the last (hidden) layer to yield the output.
- Switch on/off auxiliary losses at the last (hidden) layer of each sub-network.
- Specify number of repetitions to perform for the training/testing in order to obtain a statistical estimate of the corresponding errors.

2.2 Computational

The study in this work is divided with respect to architecture type into two parts. The first part is concerned with conventional 1D NNs followed by an investigation of NNs with 1-2

convolutional 2D hidden layers. Within each part, the investigation follows the outline:

1. Determine a suitable learning rate $\lambda \in [10^{-6}, 0.1]$ for a given architecture.
2. Train/test the neural network with fixed λ 10× and record the training/test error with the number of epochs. Use the following modii:
    - No weight-sharing, no auxiliary loss
    - Weight-sharing, no auxiliary loss
    - Auxiliary loss, no weight-sharing
    - Auxiliary loss and weight-sharing

Throughout this work, an entropy loss with an Adam optimizer is used during the training. For the outputlayer, the softmax function is applied to get the categorical data.

3 Results

3.1 Determination of a suitable learning rate

In Fig. 1, a plot of the training and test error for different learning rates of a conventional 1D NN with 1 hidden layer comprising of 600 perceptrons is shown. Clearly, the optimal learning rate for which the test error reaches a constant value within the first 100 epochs is for $\lambda = 0.001$. For higher values (0.01) and lower values (10^{-6}), little to no progress is visible. After a further refined scan of λ around 0.001, an optimal learning rate of 0.001 is selected for the ongoing investigation. Analogous to the 1D NN, Fig. 2 shows exemplary the training of a 2D convolutional network comprising of 4 hidden layers with 20 perceptrons per layer. In a similar manner, a learning rate of $\lambda = 0.001$ gives low test errors within 100 epochs. For higher and lower values, the training is significantly slower and/or gives higher test errors. Therefore, also for the 2D convolulional network a $\lambda = 0.001$ is selected.

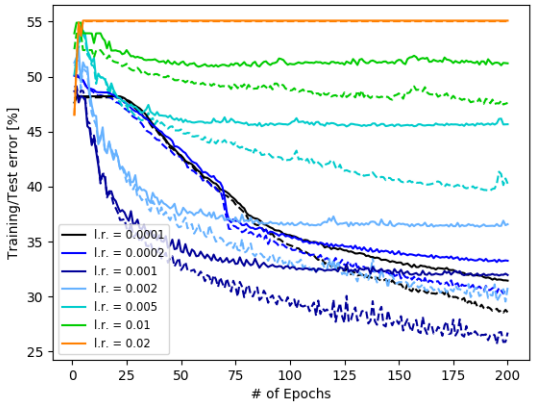

Figure 1: Training error and test error for different learning rates (l.r.), exemplary shown for a 1D conventional NN with 1 hidden layer comprising of 600 perceptrons.

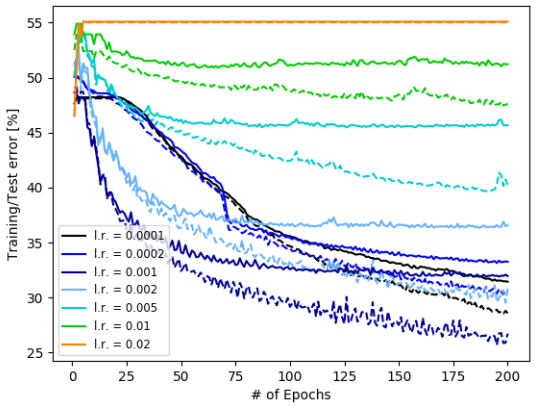

Figure 2: Training error and test error for different learning rates (l.r.) exemplary shown for a 2D convolutional NN with 4 hidden layers, each comprising of 20 perceptrons.

3.2 Impact of weight sharing and auxiliary loss

Fig. 3 and Fig. 4 show exemplary the training evolution of a 1D and 2D NN respectively when using weight-sharing and/or auxiliary losses. One can clearly see that weight sharing significantly improves the training time and furthermore leads to an improved accuracy of almost 10% for the test error. The introduction of an auxiliary loss alone improves the accuracy in a similar manner, however the training time is significantly larger (200 epochs) than for the

weight-sharing (50-75 epochs). For the 2D convolutional network a similar trend is visible, however longer training times are required to reach a conclusion on the obtained accuracies.

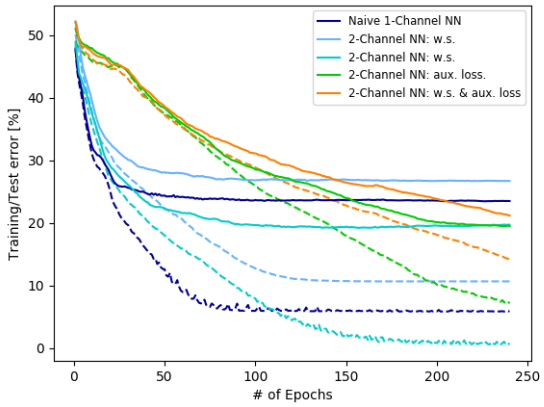

Figure 3: Training error and test error for different learning rates (l.r.), exemplary shown for a 1D conventional NN with 1 hidden layer comprising of 600 perceptrons.

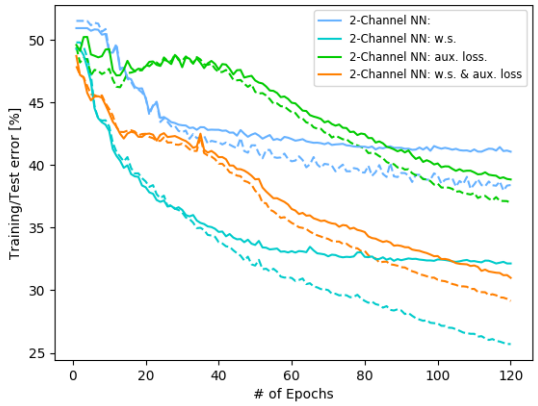

Figure 4: Training error and test error for different learning rates (l.r.), exemplary shown for a 2D conventional NN with 1 hidden layer comprising of 600 perceptrons.

The results shown above have also been observed in models with the same architecture but with higher number of perceptrons per layer, in the range of [20 - 100] and [100 - 600] for the convolutional and conventional model architectures respectively.

4 Discussion

Based on this work we can conclude that weight sharing improves training time and accuracy significantly. Additional auxiliary loss, however, only slightly improves the accuracy while increases the training time significantly as compared to weight sharing only using regular loss. The combination of both of these features – weight sharing and auxiliary loss – does not have a strong additive effect on accuracy. These observations hold up in both cases of convolutional and non convolutional networks architectures. Moving forward, one could still investigate more architectures, vary the number of layers, and add new features such as batch normalization or dropout in order to verify these observations.

References

[1] Robin Tommy, Gullapudi Sundeep and Hima Jose, 2017. Automatic Detection and Correction of Vulnerabilities using Machine Learning.

[2] Sanjay Kumar, Ari Viinikaineny and Timo Hamalainenz, 2016. Machine Learning Classification Model For Network Based Intrusion Detection System

[3] Gauri Kalnoor and Jayashree Agarkhed, 2016. Preventing Attacks and Detecting Intruder for Secured Wireless Sensor Networks.

[4] Hsiu-Chuan Huang, Zhi-Kai Zhang, Hao-Wen Cheng, and Shiuhpyng Winston Shieh, 2017. Web Application Security: Threats, Countermeasures, and Pitfalls