Excel@FIT 2021
http://excel.fit.vutbr.cz

Neural Network Training Process Visualization

Silvie Němcová*

Abstract
The neural network training process involves a highly dimensional non-convex optimization problem. The training is a highly computationally demanding task with a risk that the optimization algorithm will be stuck at a local optimum or a saddle-point. Despite these concerns, modern neural networks are trained successfully only using straightforward training with SGD algorithm and regularization. A simple technique for analyzing the training process is presented in this paper, it consists of a linear interpolation of the parameters of a neural network.
The developed tool examines the training process on the level of layers and individual parameters. The results obtained in the experiments confirm that the linear path of training is avoiding local optima, identifies the ambient and robust layers in the neural network and the results are consistent with the examination on the level of layers.

Keywords: Neural Network Training — Loss Function — Training Process Examination

Supplementary Material: *N/A*

*xnemco06@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

In the past, machine learning has suffered from several problems. One of the problems was that the training of neural networks was a highly complex and computationally demanding process. With increasing computational power and the discovery of new algorithms, machine learning is now experiencing a new wave of great popularity. Despite the popularity, there are fears about finding only local optima or saddle points during the optimization of the loss function.

The loss function evaluates how well the model represents the dataset. It is non-convex, highly dimensional, and is known to have many local optima and saddle points. Its optimization is a difficult task [1]. The loss function is usually minimized using first-order methods based on gradient descent, such as stochastic gradient descent (SGD). In this paper, the loss function is qualitatively analyzed.

Performing a linear interpolation of the parameters of a model and evaluating the performance of the model after each interpolation step is one of methods to analyze the training progress. This experiment shows that when the final parameters are known, the training of the neural network could be done using simple line [2]. However, the method does not look inside the model and does not examine what does happen in separate layers or even individual parameters of the model. The work of Goodfellow and others has served as a great inspiration.

The examination of loss function progress during training can unveil more information about the importance of layers. Like that the layers of a neural network model do not have equal significance for the model to make a prediction. Some layers are critical to form an accurate output while others are ambient. This effect is examined by Zhang et al. [3] by Zhang et al. Several models of various complexity are examined in the paper authored by Zhang et al. The paper is inspirational to reproduce the introduced experiments on a different model. In this paper, it is successfully reproduced on a relatively small convolutional neural network. Layers of the examined model were categorized as ambient and robust with confidence. This could indicate that the examined model is too big and overparametrized for its task.

This work is inspired by both of the mentioned papers. Here, we qualitatively examines the training progress and it extends the examination with a focus on the progress on the level of layers and individual parameters.

The detailed examination projected to a linear space allows us to examine the training process in separate layers and individual parameters of the model, which can show the progress of the loss function on a lower level. It also provides us with information on how much each layer or parameter affects the final result of the training. This allows us to loosely distinguish more and less important layers or even individual parameters.

2. Introduction to the Training of Neural Networks

Neural networks (NN) are computational models inspired by biological brain. They consist of simple units called *artificial neurons* [4], which are connected with weighted connections. The computational power of neural networks derives from the density and complexity of the interconnections. The neural network training consists of updating the weights of the connections until the output of the neural network is accurate enough.

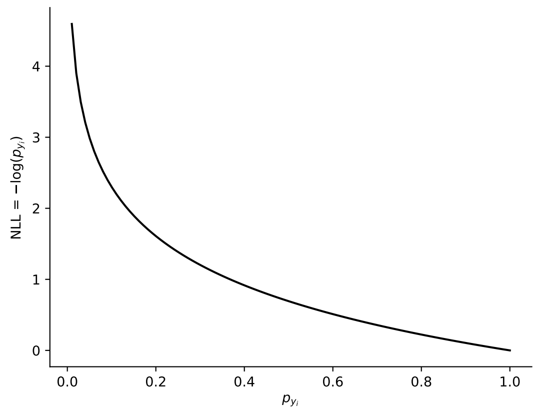

Figure 1. Negative Log Likelihood. The figure represents an output neuron corresponding to one of available classes. On the graph it is clear to see that maximizing the probability p_{y_i} of a sample belonging to class P_{y_i} will minimize the NLL value.

Formally, the training can be defined as optimizing a loss function. The loss function choice is part of the configuration of the model. It should represent how much an output of the neural network model differs from the expected output. The lower the value of the output of the loss function, the better fit.

Then, the training is done by incrementally updating the parameters of the NN model. At the start, the parameters of the network are initialized to small random values. Then the parameters are updated until the algorithm converges.

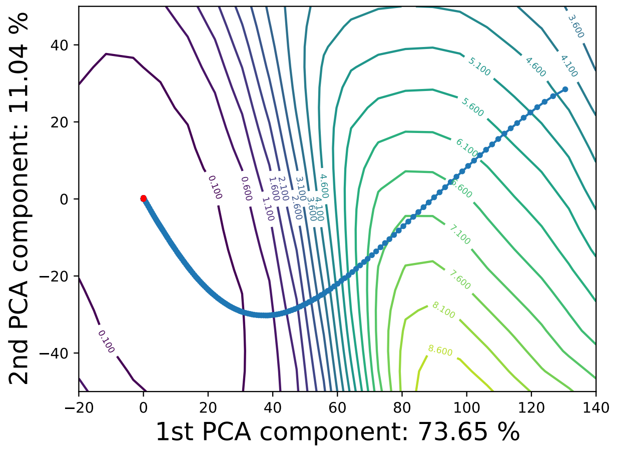

Figure 2. SGD path visualization. The SGD begins its path from the right and ends in the red dot on the left side of the path. It chooses a path that in the beginning goes through worse values of loss, but in the end, it stops in an optimum. This visualization is done by carefully choosing directions in which the training process is projected [5].

The way the weights should update is given by gradient and the change is then applied to the parameters. This algorithm is called *backpropagation*. Backpropagation usually uses first-order iterative gradient descent methods. Gradient descent almost always converges for a convex problem. However, when optimizing highly dimensional, non-convex loss functions, the gradient descent is slow and could end up in local optima [4]. Both of these flaws are solved by a stochastic gradient descent algorithm (SGD). SGD can be regarded as a stochastic approximation of gradient descent. It replaces the actual gradient with an estimate, calculated from a randomly selected subset of data. Stochastic methods avoid convergence to a local optimum [6] and using a smaller set of training examples decreases the demands on computational power. The SGD can be represented as:

$$\Theta_i \leftarrow \Theta_{i-1} - \eta \nabla J_k(\Theta_{i-1}), \tag{1}$$

where Θ are the parameters of the model, η is the learning rate, and $J_k(\Theta_{i-1})$ is the loss function for a set of randomly chosen training data subset k.

The training process can be visualized by various methods [5]. Figure 2 represents a visualization of the surface of the loss function and the path that the SGD takes during the training.

The visualization can unveil interesting facts about a neural network model and its training progress. Modern neural networks are typically too big for a task that they are assigned to. The model can be fairly in-

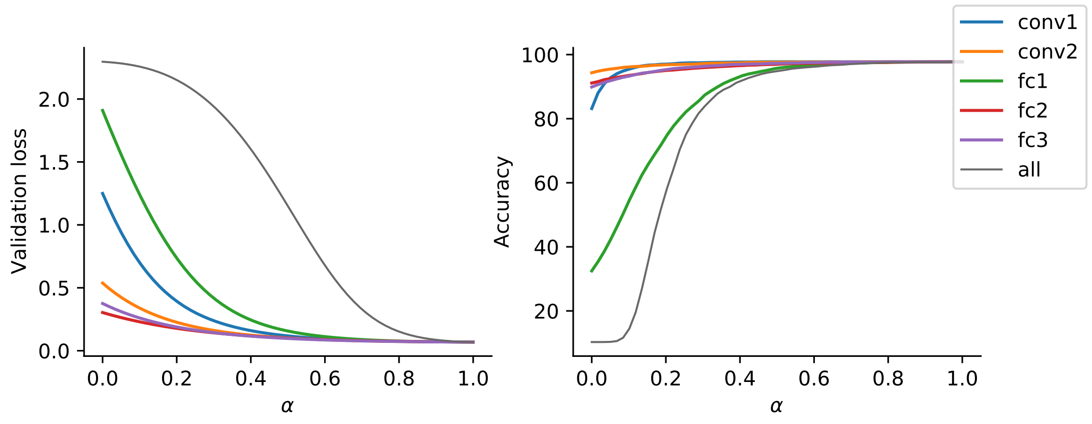

Figure 3. Impact of re-initialization of the parameters. In this experiment the model is in trained state, parameters of one layer are re-initialized and then interpolated according to (2).

sensitive to a change of parameters of certain layers. These layers are called *ambient*. Layers that do have a big impact on the performance are called *robust* [3]. Deciding whether a layer is robust or ambient can be supported by examining how fast and how much their parameters change during training. The parameters of robust layers should change in a significant way, unlike the parameters of ambient layers which should change slowly and by a little.

3. Linear Path Examination

The method I chose to visualize the training process on the level of layers is to choose two sets of parameters Θ_0 and Θ_1 and plot the values of the loss function $J(\Theta_\alpha)$ along a series of points for varying values of α. This shows a cross-section of the loss function along the line [2].

$$\Theta_\alpha = (1-\alpha)\Theta_0 + \alpha\Theta_1 \tag{2}$$

In addition, the experiment examines the distance that the parameters have traveled during the training. The distance is measured using the Euclidean distance between the initial and the final value of an examined parameter.

4. Experimental Setup

We experimented with a neural network based on LeNet-5 [7], which is trained and evaluated on the MNIST[1] dataset.

The model includes two convolutional layers (`conv1`, `conv2`) and three fully connected layers (`fc1`, `fc2`, `fc3`) arranged in this order. The activation function for all layers except the last is *ReLU*. The output of the last layer is normalized to the probability of a data sample being one of the digits using the *softmax* activation function. The loss of the network is measured using the negative log likelihood presented in Figure 1.

The parameters Θ_i are obtained before the training itself. The model is then trained and after 14 epochs the final parameters Θ_f are obtained.

[1] http://yann.lecun.com/exdb/mnist/

Table 1. Architecture of the examined model

Name	Number of Parameters
`conv1`	60
`conv2`	880
`fc1`	69240
`fc2`	10164
`fc3`	850

5. Visualizing Training Process on Layers

In the first experiment, individual layers of the model are examined. The parameters of each layer are calculated according to the linear interpolation presented in Eq. 2, with Θ_i set as Θ_0 and Θ_f set as Θ_1. The model is set in the trained state, having loaded parameters Θ_f. The observed performance of the model is evaluated after one layer of parameters loaded in the model has been replaced by an interpolated value from Θ_α.

Figure 3 shows the results of the experiment. It can be clearly seen that the first fully connected layer is *robust*, and its parameters have changed in a significant way during the training. This layer is critical for the network to be accurate as changing its parameters to the initial state causes the prediction of the model to be very inaccurate.

The second most important layer is the first convolutional layer, but its parameters have not changed as much as parameters of the `fc1` layer. Despite that, this layer also could be classified as robust, since changing the parameters of the `conv1` cause a deterioration in

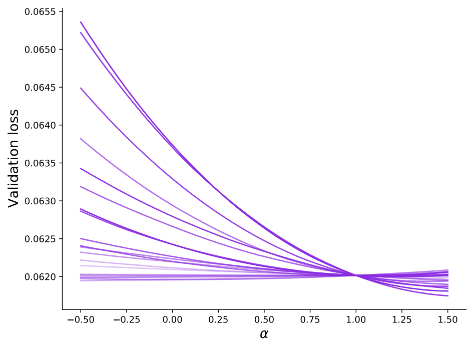

Figure 4. `conv1`

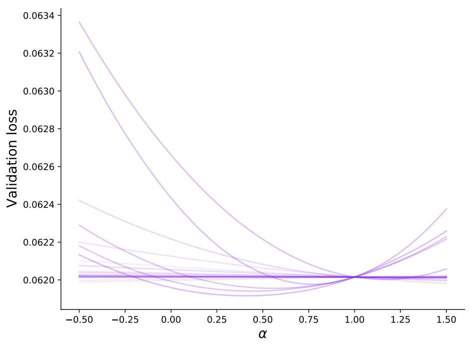

Figure 5. `conv2`

Figure 6. Parameters of the convolutional layers. The distance that the parameters have travelled is represented by the opacity of the color. The more opaque, the further the parameter has travelled.

accuracy until the interpolation coefficient reaches a value around $\alpha = 0.1$, where the accuracy acquires its almost final value. The accuracy even overtakes the accuracies of less important layers during the interpolation of parameters.

The other layers also have an impact on the final performance when their parameters are changed, but not as big as `conv1` and `fc1` has. According to the observations, these layers could be classified as ambient.

6. Visualizing Training Process on Parameters

In this experiment, the individual parameters are examined. Each layer is examined with a randomly chosen set of parameters. The `fc1` layer was examined more thoroughly as the previous experiment has indicated that this layer does have the biggest impact on the performance. The number of parameters to examine in `conv1` layer was increased after finding out that the parameters of this layer are changing by quite large values.

The final parameters Θ_f are loaded in the model. With Θ_i set as Θ_0 and Θ_f set as Θ_1 the parameters Θ_α are calculated for each value of interpolation coefficient α and the interpolated value of examined parameter replaces its corresponding value in the parameters loaded in the model. The performance of the model is calculated while having the examined parameter modified.

The results of the experiment for the convolutional layers are shown in Figure 6. Their impact on the performance is perceptible. The parameters of the second convolutional layer have a similar effect on the performance as the parameters of `conv1`. The loss function progress in both convolutional layers is corresponding with the shape of the loss function presented in Figure 1.

The first fully connected layer has the biggest impact on the final performance when changing all of its parameters. Closer examination has shown that the individual parameters of the `fc1` layer do not have a significant impact on the performance.

Similarly to the parameters of `fc1`, the parameters of `fc2` and `fc3` have only undergone only a negligible change in their values and their impact on the performance of the model is relatively small as is shown in the Figure 10. However, the interpolation shows that when changing some of the parameters, the value of the loss function worsens. The individual parameters of the output layer `fc3` have similar behavior as the parameters of `fc2` have.

7. Conclusions

The visualization of progress of the training using the linear path is successfully reproduced in this work. This has shown that the loss function is well behaved along this cross-section. A single line would do a good job of the training if the final parameters were known.

The interpolation of the parameters on the level of layers has made it possible to examine the training process with a more detailed look. In this experiment, the robust and ambient layers have been found and identified.

The second experiment examines the training on the level of parameters. The experiment unveils how much each parameter impacts the performance of the model and provides a detailed look at the training progress on the level of individual parameters.

The paper proposes a tool for examining the training process of the neural network model on a low level, which provides a look at individual parameters and layers. It can identify robust and ambient layers and can help to identify which parameters are changing slowly

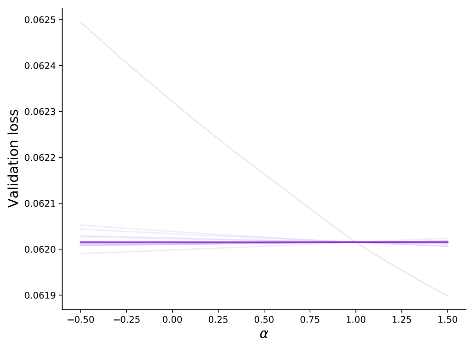

Figure 7. `fc1`

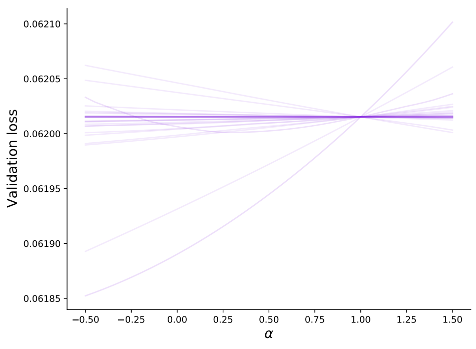

Figure 8. `fc2`

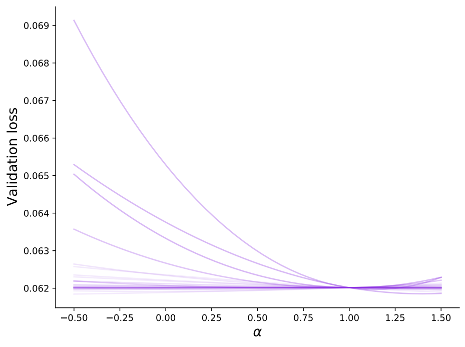

Figure 9. `fc3`

Figure 10. Parameters of the fully connected layers. The parameters of the fully connected layers have changed by a negliblible value. Each parameter individually is impacting the performance of the model in a small way. Most of the progresses are a descending lines. Some of the progresses are ascending. The ascending validation loss progresses are a optima for the examined parameter. Together with others parameters, the loss function probably has greater value in the concerned point of progress and thus the optimization algorithm never reaches this optima.

during the training or which are changing significantly.

The examination in more dimensions is to be developed. The results of this paper open a discussion about the significance of parameters or whole layers in a trained model. The tool can help identify robust and ambient layers and to gain more insight into the training progress.

Acknowledgements

I would like to thank my supervisor Ing. Karel Beneš for his mentoring and help during creation of this work.

References

[1] Y. LeCun, L. Bottou, G. Orr, and K. Muller. Efficient backprop. 1998.

[2] Ian J. Goodfellow and Oriol Vinyals. Qualitatively characterizing neural network optimization problems. *CoRR*, abs/1412.6544, 2015.

[3] Chiyuan Zhang, Samy Bengio, and Yoram Singer. Are all layers created equal? *arXiv*, 2019.

[4] Sandro Skansi. *Introduction to Deep Learning: From Logical Calculus to Artificial Intelligence*. Undergraduate Topics in Computer Science. Springer, Zagreb, 1st edition, 2018.

[5] Hao Li, Zheng Xu, Gavin Taylor, and Tomm Goldstein. Visualizing the loss landscape of neural nets. *arXiv*, 2018.

[6] Jerome Le Ny. Introduction to stochastic approximation algorithms. online, `https://www.professeurs.polymtl.ca/jerome.le-ny/teaching/DP_fall09/notes/lec11_SA.pdf`, Cited 2021-03-28.

[7] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.