TDL Report: On the Optimization of Deep Networks: Implicit Acceleration by Overparameterization

Guillaume LEVY Clement WANG

February 2024

This report is based on the paper: **On the Optimization of Deep Networks: Implicit Acceleration by Overparameterization** by Sanjeev Arora, Nadav Cohen and Elad Hazan [1]. The code can be found here.

1 Introduction

In deep learning, it is common knowledge that adding depth to a model will increase its expressiveness, albeit at the expense of introducing challenges in optimization, such as the vanishing gradient problem. However, this paper suggests a counterintuitive idea that increasing depth will result in accelerating the convergence. To effectively assess this proposition, various models with differing depths will be examined. To isolate the effect on the convergence while maintaining the same expressiveness, these models will not have any activation function and thus will be equivalent to a neural network with only one layer. This report will delve into the theoretical underpinnings and experimental validations of the advantages offered by this overparameterization.

2 Theoretical Aspects

First, let's see the properties that the overparametrization has. As explained in the introduction, we consider a depth N neural network. The weights of each layer are noted W_i for i from 1 to N. We use the identity function for the activation. This causes the network to be equivalent to a neural network with only one layer. The setup is resumed on figure 1.

The difference with a model with only one layer lies in the optimization of this model. Since the proof is a bit too long and complicated for the scope of this report, we encourages the reader to refer to the original paper for the details. To obtain the result, the model is required to be updated using

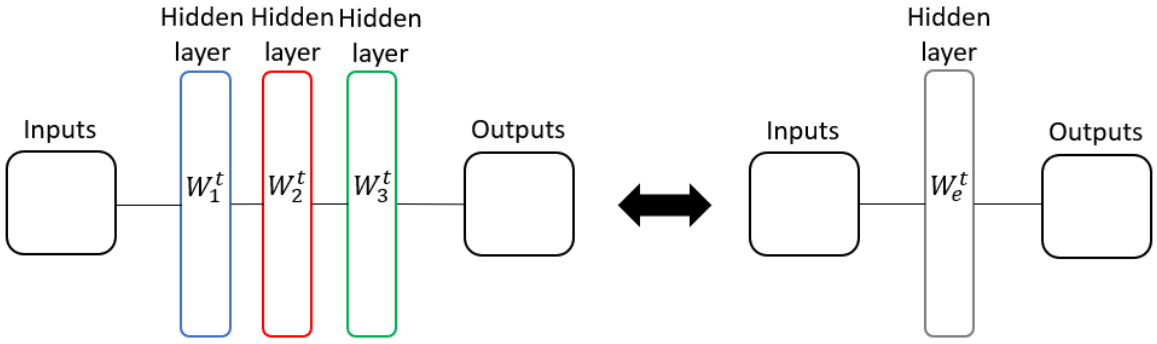

Figure 1: The architecture used for the model. On the left, the model as it is optimized and on the right the equivalent model when used. W_1^t,W_1^t and W_3^t are the weights of the three layers respectively after the t update while W_e^t is the weight of the equivalent model: $W_e^t = W_1^t W_2^t W_3^t$

the stochastic gradient descent (SGD) with or without regularization. The update for each layer is the following.

$$\dot{W}_j^t = -\eta\lambda W_j^t - \eta\frac{\partial L_N}{\partial W_j}(W_1^t, W_1^t, \cdots, W_N^t) \quad \forall j \in [1, N]$$

We also assume that the initial values of the weight matrices W_1, W_2, ..., W_N follows:

$$W_{j+1}^T W_{j+1} = W_j^T W_j$$

The update of the model can then be written as:

$$\dot{W}_e = -\eta\lambda N W_e - \eta\sum_{j=1}^{N}[W_e W_e^T]^{\frac{j-1}{N}} \cdot \frac{dL^1}{dW}(W_e) \cdot [W_e^T W_e]^{\frac{N-j}{N}}$$

with the equivalent matrix:

$$W_e^{(t)} = W_1^{(t)} W_2^{(t)} \cdots W_n^{(t)}$$

The time dependence is not specified for readability but is present. For simplicity of the formula, we consider the case of a single output. The updates then becomes:

$$\boxed{W_e^{(t+1)} \leftarrow (1-\eta\lambda N)W_e^{(t)} - \eta\|\boldsymbol{W_e^{(t)}}\|_{\mathbf{2}}^{\mathbf{2}-\frac{\mathbf{2}}{\boldsymbol{N}}} \cdot \left(\frac{dL^1}{dW}(W_e^{(t)}) + (N-1)\cdot \boldsymbol{Pr}_{\boldsymbol{W_e^{(t)}}}\left\{\frac{dL^1}{dW}(W_e^{(t)})\right\}\right)}$$

As we can see, this optimization holds two interesting properties, highlighted in the update equation. It has an adaptive learning rate with the term: $\|W_e^{(t)}\|_2^{2-\frac{2}{N}}$ and the update of the layers is in the direction of the gradient of the equivalent matrix. These enhancements cannot be brought up by regularization techniques and therefore are the specificity of this architecture. We will see in the next section if these benefits the model.

3 Experiments

3.1 Experimental setup

Through different experiments, we found out that the results were highly dependent on the initialization and the training setup. To have a better understanding, we decided to do our experiments on two very different datasets.

- Gas Sensor Array Drift Dataset at Different Concentrations [3]:
  - Predicting the concentration of the ethanol. This dataset was used for the experiments of the original paper.
  - 2565 samples, 128 features
- Abalone length dataset [2]:
  - Predicting the age of abalones.
  - 4177 samples, 8 features

As we are studying the speed of convergence of the neural networks, we want to be independent of the choice of the learning rate. Therefore, all experiments were run with a grid search on the learning rate: [0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01]. We selected the run with the best training loss at the last epoch. Also, as we are only interested in the convergence, all the metrics concern the training set. Except when it is clearly stated otherwise, we used the SGD optimizer of Pytorch to train our model with batch gradient descent on the mean square error loss (MSE).

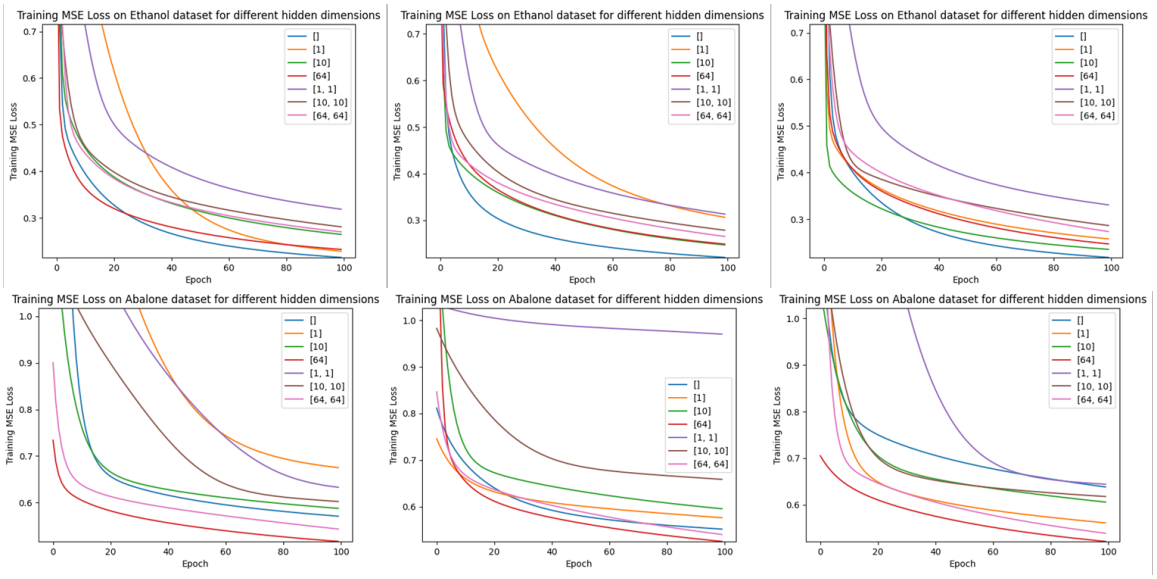

Figure 2: Learning curves of linear nets without activation for different hidden layers.

3.2 Linear nets without activation

The first experiment consisted of training randomly initialized neural networks with 1, 2, or 3 layers just as in the original paper. As stated before, we were surprised by the stochasticity of the results. To illustrate it, we ran three times the exact same experiments on both datasets (Figure 2).

The 1-layer linear regression is consistently better on the Ethanol dataset while 2-layer and 3-layer networks consistently do better on the Abalone dataset. However, we can see that the results heavily depend on the width of the layers while the paper suggested that it did not have an impact on the learning dynamics. Moreover, the same conclusions could not be drawn from both datasets.

3.3 Towards a better initialization

As mentioned in the previous section, some conditions must be verified by the weights at the beginning of the training so that the mathematical formula of the previous section holds. We have to have for all $1 \leq j \leq N-1$:

$$W_{j+1}^T(t_0)W_{j+1}(t_0) = W_j^T(t_0)W_j(t_0)$$

This is in particular verified for the "near zero" and "near identity" initializations. "near zero" initialization corresponds to having close to zero weights at the start of the training and "near identity" is having semi-orthogonal matrices. We implemented both settings and compared the learning curves.

There is much less stochasticity when using a particular initialization process. Sometimes in the over-parametrized setting, we get some unexplained bursts in the loss. In the over-parametrized setting, "near identity" initialization seems to converge faster. However, we still cannot draw any conclusion on a better convergence speed of deeper regression neural networks.

3.4 Comparison with a different optimizer

Theoretically, having a deeper regression neural network is akin to adding an adjusting learning rate and a momentum term so we decided to compare directly with Adam optimizer.

It seems like Adam gets better metrics for all architectures on both datasets. The over-parametrization seems to make convergence slightly faster in our experiments with Adam.

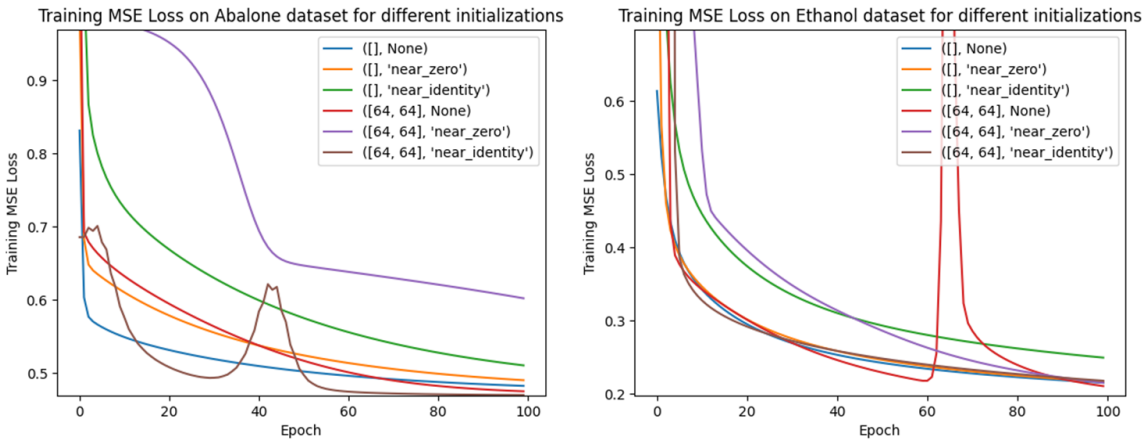

Figure 3: Learning curves of linear nets with specific initializations for different hidden layers.

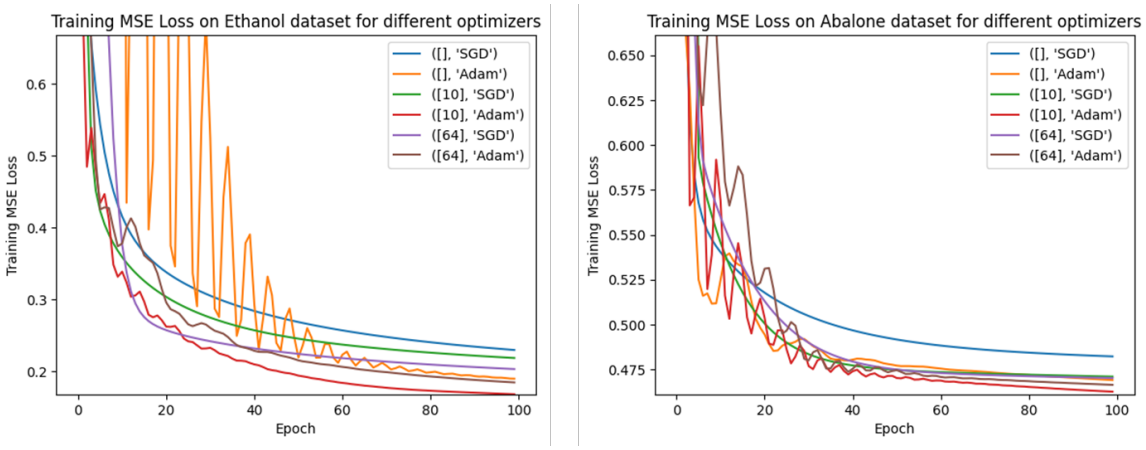

Figure 4: Learning curves of linear nets with near identity initialization for different hidden layers and trained with different optimizers.

4 Conclusion

This paper demonstrates that stacking multiple linear layers changes the update of the model. Theoretically, this introduces a learning rate regularizer and momentum. However, experimental findings diverge from anticipated outcomes. The optimization seems dependent on the dataset and on the initialization of the model which therefore differs from the conclusion of the paper.

References

[1] Sanjeev Arora, Nadav Cohen, and Elad Hazan. On the optimization of deep networks: Implicit acceleration by overparameterization, 2018.

[2] Sellers Tracy Talbot Simon Cawthorn Andrew Nash, Warwick and Wes Ford. Abalone. UCI Machine Learning Repository, 1995. DOI: https://doi.org/10.24432/C55C7W.

[3] Alexander Vergara. Gas Sensor Array Drift Dataset at Different Concentrations. UCI Machine Learning Repository, 2013. DOI: https://doi.org/10.24432/C5MK6M.