Deep Learning Solution to Age Estimation from Facial Image

Necip Yildiran
Computer Engineering Department
Middle East Technical University
Ankara, Turkey
necip.yildiran@ceng.metu.edu.tr

***Abstract*—The purpose of the work that is reported in this paper is to identify the effects of changing certain hyper-parameters on a deep neural network, such as number of epochs, learning rate, number of hidden layers and number of neurons in hidden layers. To achieve so, an Age Estimation (AE) system is constructed using the PyTorch framework. Holding experiments with varying hyper-parameters, neural networks are trained, which leads to discussion on the effects of altering hyper-parameters with the comparison of their loss and accuracy history on training and validation set. As a result, an example set of hyper-parameters that gives satisfying results is obtained.**

I. Introduction

Deep Neural Network (DNN) is a specific multi-layer architecture for deep learning, which stands to represent data with complex non-linear functions. The parameters of a DNN represents the specific model. In order to develop its data specific parameters, it is subjected to a training process, which requires right choice of hyper-parameters to achieve good representation of the model. In the "Method" section, brief information about the implementation that is used for evaluation process could be found. In "Evaluation" section, the statistics from experiments with different set of hyper-parameters could be found. In addition, the effect of changing each hyper-parameter is discussed. In "Conclusion" section, the final set of hyper-parameters that are chosen to give the best results are shared by citing the outcome of the evaluation process. Moreover, the system is tested with my self portrait and the results of it are shared.

II. Method

The implementation of the AE system is mainly based on the deep learning framework, PyTorch. For plotting concerns, python plotting library, Matplotlib is employed. A parametrized structure is followed while implementation to ease changing the set of hyper-parameters. The tensor operations are held on GPU thanks to the CUDA support of PyTorch, which provided acceleration. While hyper-parameters used in the training process have accountably distinctive effects on the results, they cannot be set independent of each other. For example, decreasing the learning rate might require increasing the number of epochs, or, increasing the number of hidden layers might require a new setting for number of neurons inside these hidden layers. Therefore, it would not be realistic to choose a value for a hyper-parameter independent of others. Although the effect of each hyper-parameter on the accuracy is examined with this approach in the following sections, the final setting of hyper-parameters is obtained gracefully, by taking the effect of each hyper-parameter on others into consideration.

III. Evaluation

A. Number of Hidden Layers and Number of Neurons in Hidden Layers

As known priori, both number of hidden layers and hidden layer sizes control how complicated functions we could express with the neural network. As we increase the number of layers and the number of neurons of these hidden layers, we could achieve a model that expresses the training data much better. However, it might be problematic in our case and cause over-fitting since we are not employing any regularization. In other words, the capability of expressing complicated functions of our network might proceed to too much relying on the training data, which loosens the generality of our model and makes it prone to errors arising from noise in training data. In addition, the complexity of the function to represent depends on the application. Even if we applied regularization, using a unnecessarily big number of hidden layers and layer sizes would not make contribution to and would cause to decline in performance while training. Knowing the above-mentioned issues, I held experiments with varying number of hidden layers and varying hidden layer sizes. While doing so, I kept other hyper-parameters constant, which are number of epochs and learning rate.

As could be seen from Fig.1, Fig.2, Fig.3 and Fig.4 respectively, as the number of layers increase, the loss on both training and validation data decreases. However, when the number of hidden layers is increased from 3 to 4, which could be seen by comparing Fig.4 and Fig.5, the loss on both training and validation data increased by a little amount, which could be explained with the continuous peaks and troughs on the loss figure. However, they are not caused by the number of layers. It is an expected result arising from the learning rate, about which you could find my comments in "Learning Rate" subsection. When I concentrate on the hyper-parameter I am trying to get a good value for, which is number of layers, I chose 3 hidden layers appropriate because of two main reasons. Firstly, decreasing it increases the loss, which is not

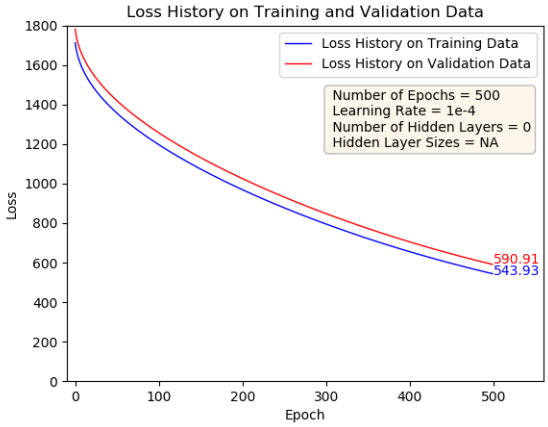

Fig. 1. Loss History on Training and Validation Data

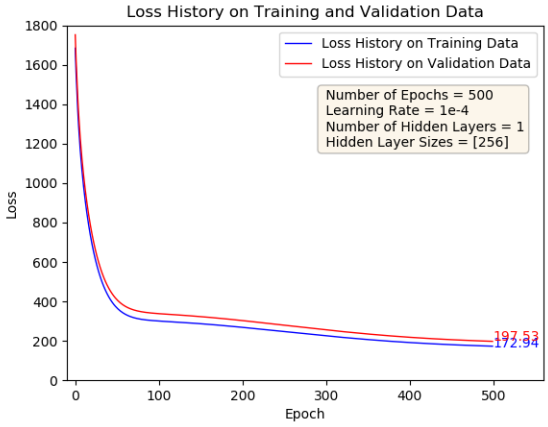

Fig. 2. Loss History on Training and Validation Data

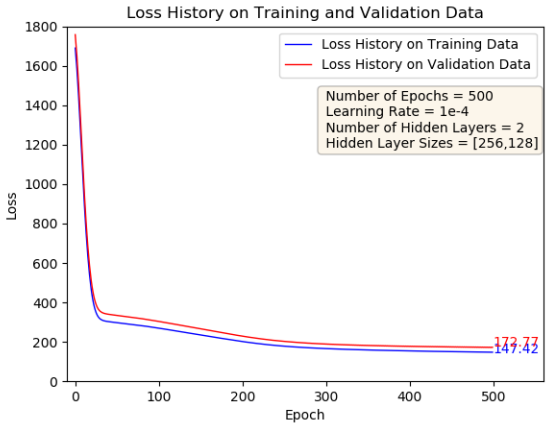

Fig. 3. Loss History on Training and Validation Data

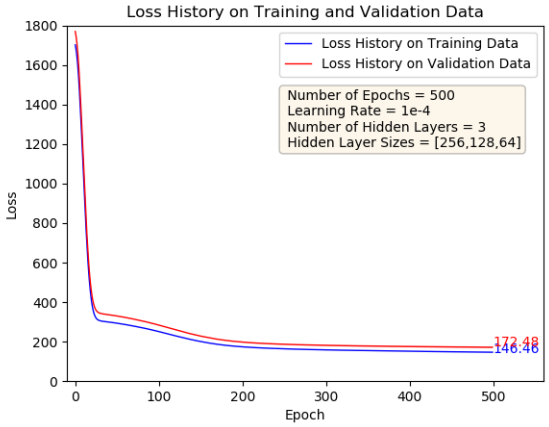

Fig. 4. Loss History on Training and Validation Data

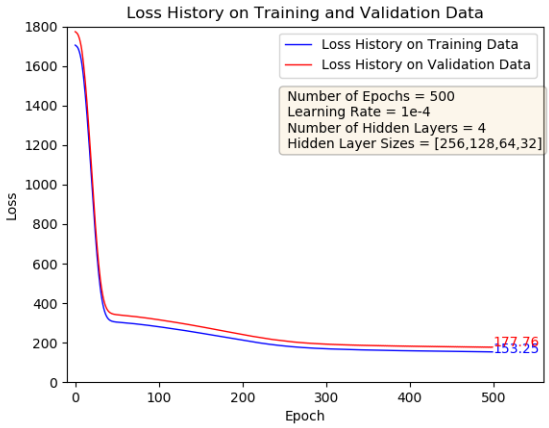

Fig. 5. Loss History on Training and Validation Data

desired. The reason is that a smaller number of hidden layers than 3 does not give enough space to represent the function of the model we try to create. Secondly, larger number of hidden layers than 3 does not make contribution, which causes decline in training performance in terms of time.

When I increased the number of neurons in the hidden layers from [256,128,64] to [256,256,256], loss on both validation and training data are decreased as could be seen from Fig.6 and Fig.7. This was an expected result since increasing the layer sizes allows the neural network to develop a better fit to the desired model.

B. Learning Rate

When Fig.8 and Fig.9 compared, the hyper-parameter settings change only by their learning rates. The losses in Fig.8 makes continuous peaks and troughs with a high frequency each epoch while the losses in Fig.9 do not. This could be explained with the fact that large learning rates makes it harder to reach to local minima by yielding an indecisive learning process. On the other hand, small learning rates could achieve this task better by being slower but much more precise while

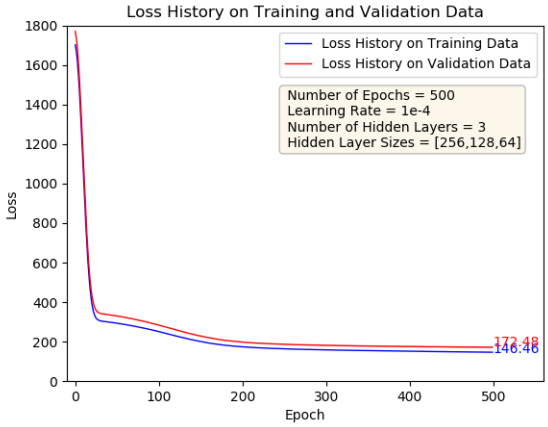

Fig. 6. Loss History on Training and Validation Data

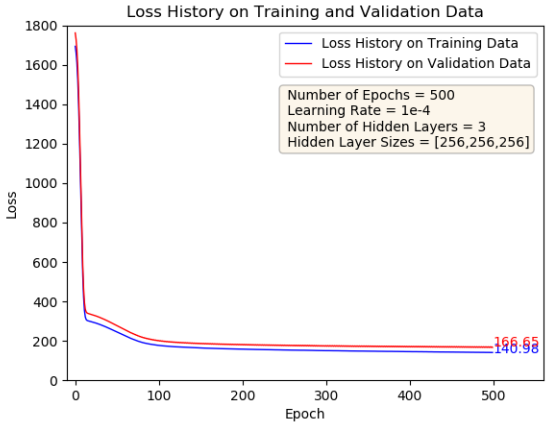

Fig. 7. Loss History on Training and Validation Data

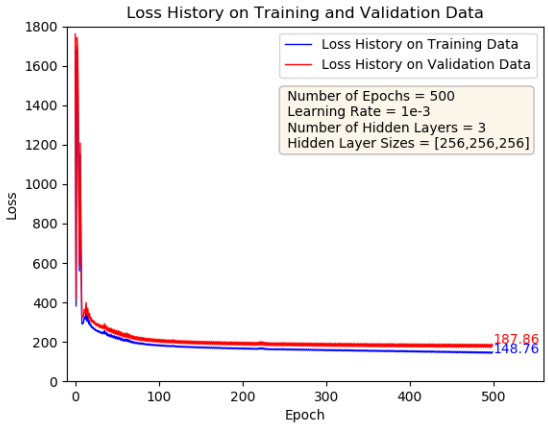

Fig. 8. Loss History on Training and Validation Data

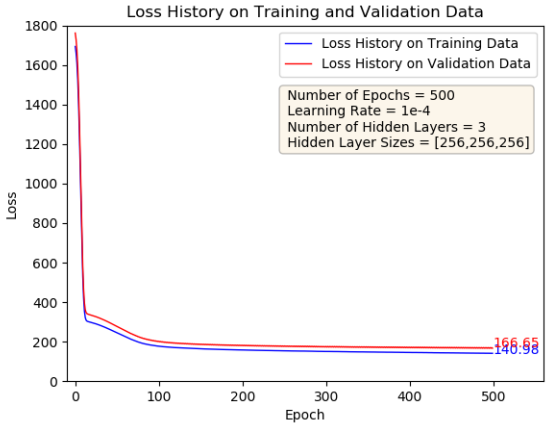

Fig. 9. Loss History on Training and Validation Data

training. However, how large or small the learning rate should be depends on the use case.

C. Number of Epochs

While choosing the number of epochs, I needed to decide when the training converges to its local minima. Choosing a small number of epochs than required leads to premature models, which are not trained enough. The result of this could be seen by examining Fig.9 since the loss at 50^{th} epoch is larger than that of at 500^{th}. On the other hand, choosing a large number of epochs than required does not harm the wellness of the model but wastes the computational power by making unnecessary training rounds.

IV. Conclusion

Having detected the approximate values for hyper-parameters, I needed to do fine adjustment to obtain the final hyper-parameters that give the best results. I chose the number of hidden layers as 3 and hidden layer sizes as [256, 256, 256] as mentioned in the "Evaluation" section. On top of these hyper-parameters, I chose the learning rate as $1e-5$ and the number of epochs as 4000. I noticed that this learning rate avoids difficulty in reaching a local minima by following a decisive path and requires an acceptable number of epochs. 4000 as the number of epochs has been a good choice because it is the minimum number of epochs required to achieve the local minima.

Fig.10 shows the loss history and Fig.11 shows the accuracy history with the final hyper-parameter settings.

To further evaluate the accuracy of the system and test the behavior of it under changing circumstances, I used the age estimator on my self portraits. For the images in Fig.12, Fig.13 and Fig.14, I obtained the age estimations 37, 34 and 38, respectively. Although the age estimator did not give accurate results on my self portraits since my age is only 23, it estimated consistent ages for images of the same person with changing scene circumstances.

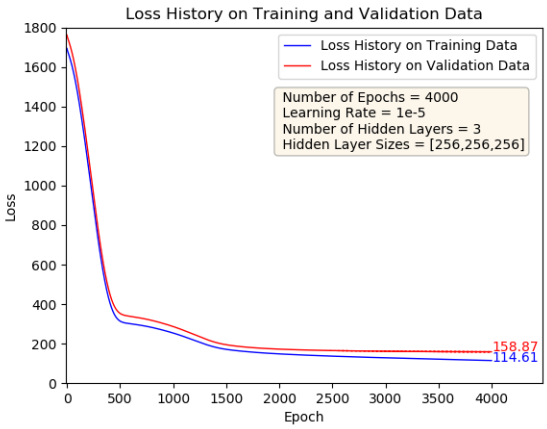

Fig. 10. Loss History on Training and Validation Data with Final Hyper-Parameter Setting

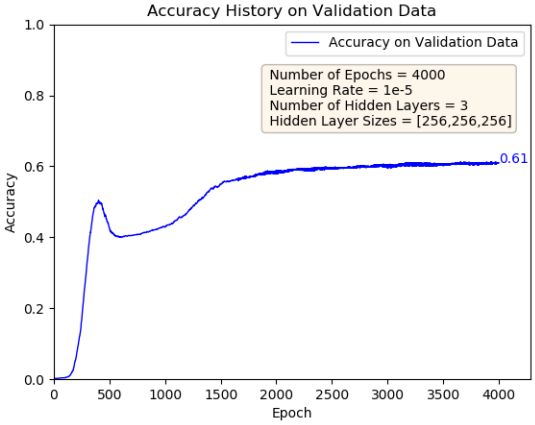

Fig. 11. Accuracy History on Validation Data with Final Hyper-Parameter Setting

Fig. 12. Self-portrait in Clear Scene Conditions

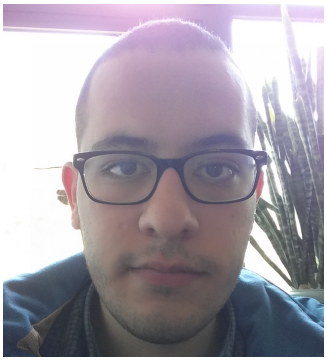

Fig. 13. Self-portrait in Scene with Unperfect Lighting

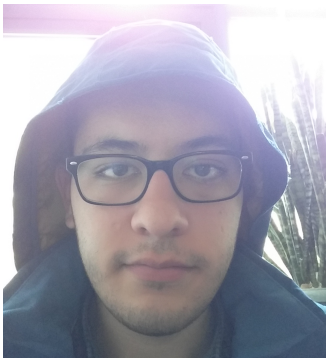

Fig. 14. Capped Self-portrait in Scene with Unperfect Lighting