# CENG483 HW2

## Constructing an Age Estimation System

Murat Topak
Department of Computer Engineering
Middle East Technical University
Ankara, Turkey
murat.topak@metu.edu.tr

***Abstract*—This report explains the work carried out and presents results about constructing an Age Estimation System based on deep learning methods. Effect of the size of the neural network, and hyper-parameter choosing on the system is explored throughout the document.**

***Keywords*—*Age Estimation; Deep Learning***

## I. Introduction

The main purpose of the Age Estimation (AE) systems is to determine the age of the person in a query image. The problem is the semantic gap between the images and their representations. In this study, to overcome this difficulty, images are described as featured vectors which are higher level representations than collection of pixels. With these feature vectors age estimation can be formulated as a learning problem to match an image representation with the age of a person.

In the experiments, a fully connected network with rectified linear unit as nonlinearity function between layers is constructed. The network is trained with RMSprop optimizer and mean squared error loss function (1) is used to minimize the difference between the reality and the estimation.

$$L = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 \qquad (1)$$

For evaluation, 0-1 loss (2) is used with threshold 10. This means an estimate only in the margin of 10 is considered successful.

$$L = \frac{1}{n}\sum_{i=1}^{n} l_i, \qquad l_i = \begin{cases} 1 & |\hat{y}_i - y_i| > \theta \\ 0 & |\hat{y}_i - y_i| \leq \theta \end{cases} \qquad (2)$$

## II. Implementation Details

PyTorch with no GPU support has been used to train the network. The train dataset consists of 5000 images while the validation set has 2000 images.

For the sake of consistent testing, random numbers have been seeded. This allowed the experiments to be repeatable.

## III. Results

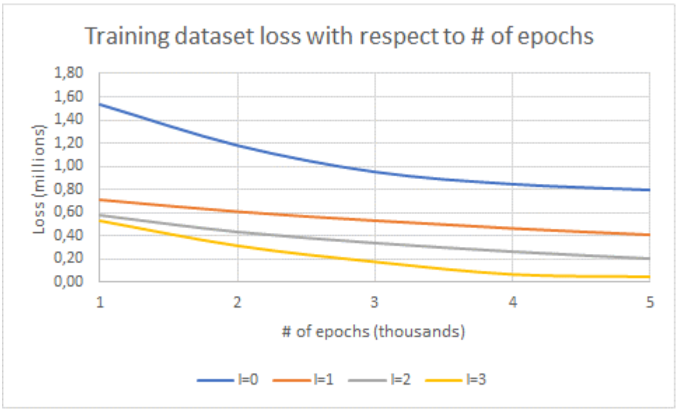

Fig. 1: Training dataset loss history

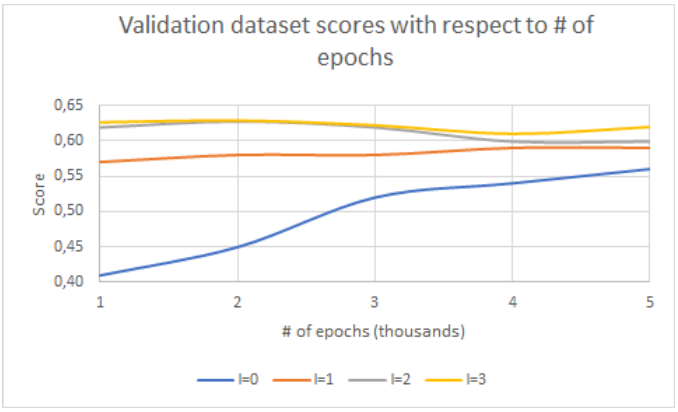

Fig. 2: Validation dataset scores

*Number of Layers*

In Fig. 1, training dataset loss history is presented. The learning rate is $10^{-4}$, and the number of neurons in every hidden layer is 512 throughout this figure. The vertical axis indicates the loss values in millions. The horizontal axis is the number of epochs in thousands. There are four lines representing 0,1,2,3 layers.

Having explained that, it is clearly seen, at any epoch, the loss is less with more layers in the network. This makes sense because with the increased number of layers, the number of neurons is also increased. With more neurons, the model can store more examples and effectively reduce the lost during the training. However, the speed is sacrificed in turn. One epoch takes more time when the number of layers is increased.

It is inferred from Fig. 2 that complex networks (in terms of number of layers) performed better. Hence, the conclusion on the number of layers is that the capacity of the network increases with the number of layers. That is, the space of representable functions grows, and more complicated functions can be learned. This is great but always open to overfitting [1].

### *Hyper-parameters*

In the experiments, the best learning rate was $10^{-4}$. With the higher learning rates, the loss was exponentially decaying but stuck at worse values. This is because there is too much "energy" in the optimization and the parameters are bouncing around chaotically. With the lower learning rates, the improvement was linear.

For the size of the neural network, there are no formal methods to follow. Try and Error with forward approach in [2] has been used to determine the optimal size. That is, number of hidden neurons is increased until training and testing is improved. Of course, there are some rule of thumbs here such as arranging the number of neurons so that it lies between the size of the input layer and output layer.

Number of epochs was also determined by trial and error. For example, in the cases where the loss was stuck due to high learning rate, it was no good to continue to train the network. The same applies also when the score converged to a value. Therefore, the number of epochs was chosen such that it respects the time passes, loss, and score values.

### *Estimates*

Three of the worst estimates of the best model is shown in Fig. 3, Fig. 4, and Fig. 5. The problem with the first two is that they have their ground truth values defined wrong. For example, the ages defined for these are 111 and 81 respectively while the trained model predicts them as 31 and 22. The third picture on the other hand is really handled poorly by the model. The person is at the age of 89 according to the ground truth dataset but the model predicts him at 39. The reason might be the low quality because the picture is so smooth the person looks like he does not even have any wrinkles at all. Perhaps there is nothing like this in the training dataset, so the function failed to predict correctly.

On the other hand, the old man in Fig. 6 is predicted perfectly by the model. There are other examples like this such as 'v718.jpg', 'v1224.jpg', and 'v1545.jpg' where the prediction is almost perfect. One thing common in all these images is that they don't resemble the old man in the black and white photo. Therefore, the model cannot be blamed for its low accuracy on 'v192.jpg' since it has not been trained with that kind of samples.

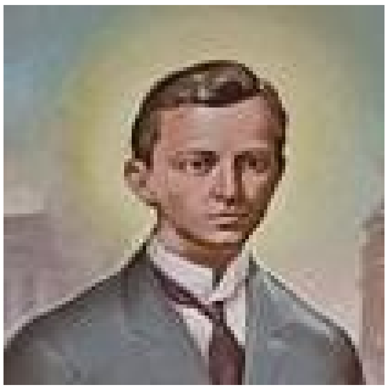
Fig. 3: 'v43.jpg'

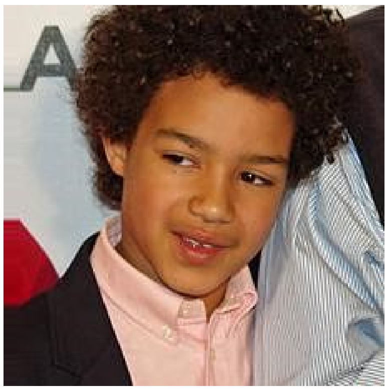
Fig. 4: 'v1213.jpg'

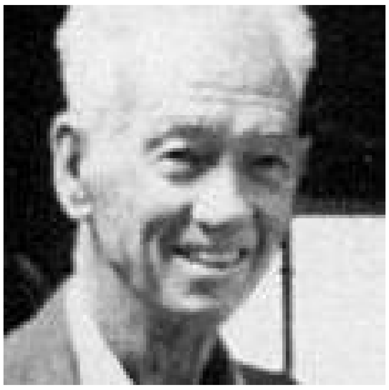
Fig. 5: 'v192.jpg'

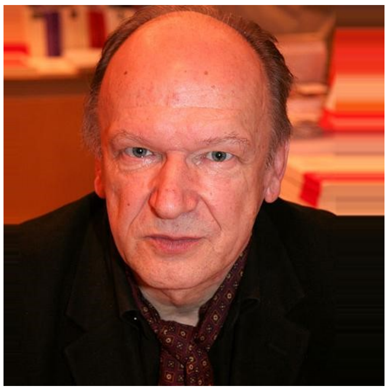
Fig. 6: 'v327.jpg'

*Self-portrait*

I must have had funnier pictures but these two were quick to find. My best network tells I am 33 and 30 in Fig.7 and Fig. 8 respectively. At the time these photos were taken I was like 20. The deal here is that the accuracy improved with a beanie. When the train dataset is browsed through, one encounters people with various hats and helmets. Maybe my feature vectors are more like those than the others.

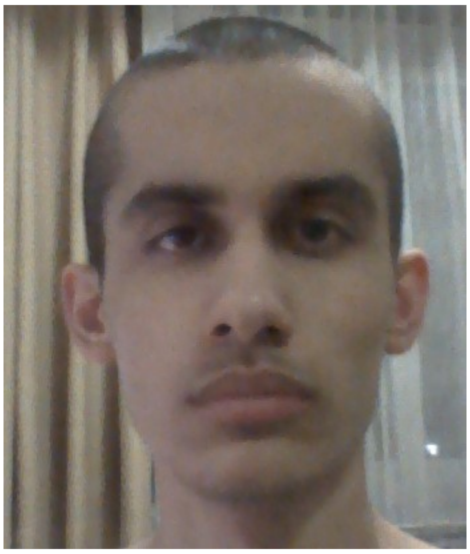

Fig. 7: Me without a beanie

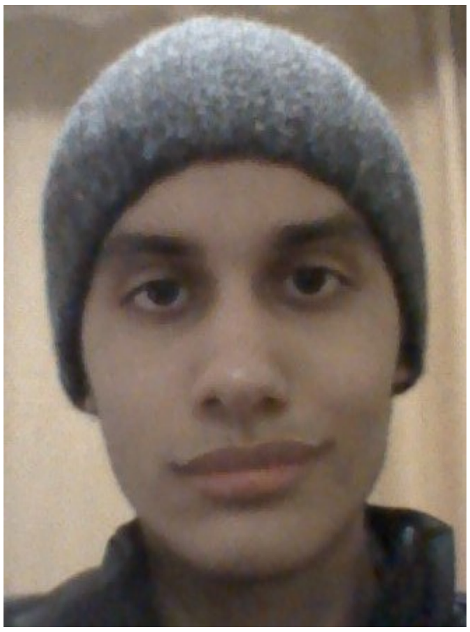

Fig. 8: Me with a beanie

## References

[1] Saurabh Karsoliya, 2012, "Approximating Number of Hidden layer neurons in Multiple Hidden Layer BPNN Architecture", *International Journal of Engineering Trends and TechnologyVolume3Issue6-*

[2] F. S. Panchal and M. Panchal. "Review on Methods of Selecting Number of Hidden Nodes in Artificial Neural Network". In: *International Journal of Computer Science and Mobile Computing* 3.11 (2014), pp. 455–464.