Deep Learning Lab: Assignment #1

Francisco Rivera Valverde
Francisco.rivera@students.uni-freiburg.de

***Abstract*—This document summarizes the architectural network required to classify 28x28 human handwriting. The network employs 1 input layer, 2 fully connected layers (being it a ReLu and a Tanh) and a softmax output layer. Training with a negative likelihood loss yielded a 1.77 percentage error when using the whole test data.**

***Index Terms*—Deep Learning**

I. Introduction

The interpretation of human handwriting allows commercial institutions to provide ease of use services. For example, a bank can interpret a handwritten account, using this as a competitive advantage. The work presented here explains the design of a neural network to classify a 28x28 human handwriting. The network employs 1 input layer, 2 fully connected layers (being it a ReLu and a Tanh) and a softmax output layer.

The goal of the network is to reduce the training error, while using validation data, to be lesser than 1.6%. The section 2 describes the set of classes provided to the students. Section 3 justifies how the network was created. Section 4 describes the results obtained, while the section 5 describes conclusions and recommendations in the topic.

II. The Object Oriented Layer Architecture

The provided skeleton for a class base neural network, available in https://github.com/aisrobots/dl-lab-2018, allowed the creation of n-hidden layer neural networks, with tanh, sigmoid or ReLu activation. Figure 1 describes the architecture and denotes the complication induced by the object abstraction of the neuronal class with layers as objects. A NN (Neural network class for short), receives a group of layers, or class architecture, to learn how to decipher human handwriting. There are three types of layers: Input (which just propagates the inputs to the NN so the user can predict what number does the handwriting represent), Output Layer (softmax output with one hot encoding for easier training) and hidden layers (which are parametrizable objects that can learn through the weights and bias attributes of the class).

The number of layers was mandated to be between 2 -4 layers, not accounting for the input and output layers. As a design constrain, the author decided to use the minimum number of layers to accomplish the goal of 1.6% error over validation data. The next section justifies the parameter configurations selected in this document.

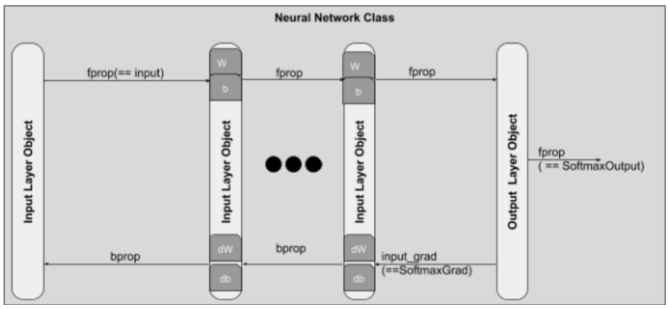

Fig. 1. Architecture of the neural network communicating layer objects between their forward and backward propagation methods

III. Designing the Architecture

A testbench was created to compile the permutations of different configuration parameters. This means, that several experiments where registered on an automated way while changing:

- Learning rate = {0.001, 0.005, 0.007, 0.01, 0.5, 0.7}
- Activation of Layer 1 = {relu, tanh, sigmoid}
- Units of Layer 1 = {50, 100, 200}
- Activation of Layer 2 = {relu, tanh, sigmoid}
- Units of Layer 2 = {50, 100, 200}
- Number of epochs = {20, 30, 40}

The goal of this testbench is to provide a direction of how to tweak the network parameters. It runs on validation data, not on test data for speed. Following data presented in this section base their underlaying reasoning in permutations of the above parameters, ran on a GPU based Python 3 kernel.

The first parameter of interest is the learning rate, where Fig2 shows that smaller learning rates significantly degrade the

This paragraph of the first footnote will contain the date on which you submitted your paper for review. It will also contain support information, including sponsor and financial support acknowledgment. For example, "This work was supported in part by the U.S. Department of Commerce under Grant BS123456".

The next few paragraphs should contain the authors' current affiliations, including current address and e-mail. For example, F. A. Author is with the National Institute of Standards and Technology, Boulder, CO 80305 USA (e-mail: author@ boulder.nist.gov).

S. B. Author, Jr., was with Rice University, Houston, TX 77005 USA. He is now with the Department of Physics, Colorado State University, Fort Collins, CO 80523 USA (e-mail: author@lamar.colostate.edu).

T. C. Author is with the Electrical Engineering Department, University of Colorado, Boulder, CO 80309 USA, on leave from the National Research Institute for Metals, Tsukuba, Japan (e-mail: author@nrim.go.jp).

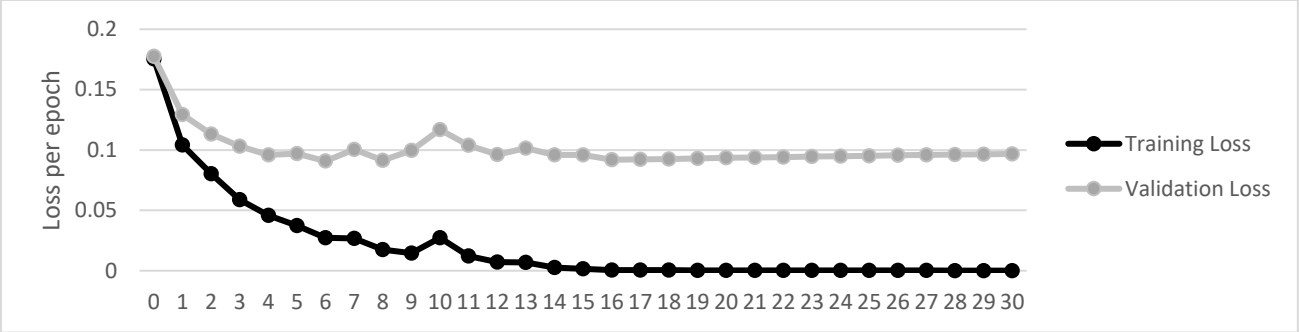

Fig. 3. Error rate registered for different activation layers and unit layers, when varying the learning rate between 0.001 to 0.7. Even though some outliers exist at 0.5 and 0.7 learning rates, higher error rates where more probable with 0.001 learning rates

error percentage. The author decided to use a learning rate of 0.5 as a fixed parameter for subsequent experiments because of the low error percentage registered as compared to lower learning rates.

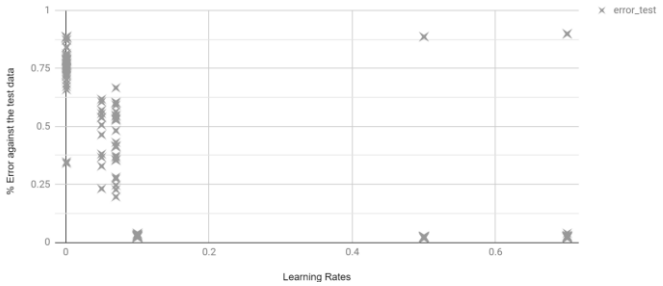

Fig. 2. Error rate registered for different activation layers and unit layers, when varying the learning rate between 0.001 to 0.7. Even though some outliers exist at 0.5 and 0.7 learning rates, higher error rates where more probable with 0.001 learning rates

Additionally, to decide the type of activation layers, one can add the validation error of each trial run, to get a guidance on what flavor of activation function to use. Particularly, the following table registers in column one the activation function used in layer one, while the following columns emulate the outcome of different second layer activation functions. Using a tanh activation function in second layer is a conclusive recommendation. For layer one, the table recommends to use a ReLu or a tanh. Because ReLu provides better results among subsequent runs (due to the sum of experiments), the author decided to use this activation function.

TABLE I
SUM OF ERRORS FOR DIFFERENT ACTIVATION LAYERS

	Second Layer		
First Layer	relu	sigmoid	tanh
relu	0.28410	0.26040	0.24000
sigmoid	6.41580	0.32890	0.28880
tanh	0.27460	0.27580	0.26150

Regarding the missing parameters, the best results was yielded by using 200 units in the first layer, 100 in the second, and 30 epochs.

IV. RESULTS

The product of the architecture selected. After 30 epochs, the network outputted a training error of 0 (the precision was set to 4 decimals), a validation data error of 0.0180 and an error on the test data of 0.0177 . Figure 3 plots the loss in an effort to give insights about the network behavior.

It is possible to use dropout to reduce the noticeable difference between training and validation loss. This was implemented as an advanced feature through the use of a binomial probability matrix, that kills or allows the output activation of layers to be propagated. It is important to note, that when doing backpropagation, the code accounts for binomial probability to prevent feedback from weights that did not contribute to forward propagation due to dropout. The figure 4 shows, on matplot, how the training loss is now perceived.

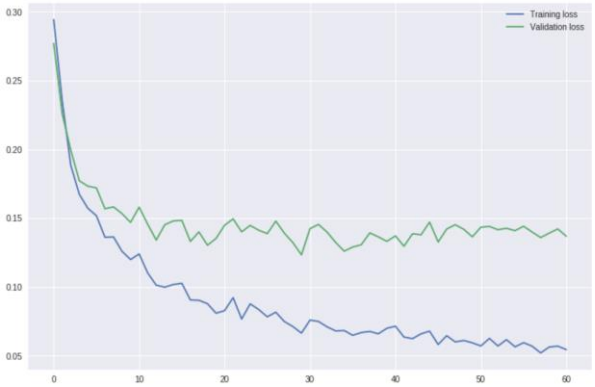

Fig. 4. Training and validation loss for 50% dropout included in both fully connected layers

V. CONCLUSIONS AND RECOMMENDATIONS

With just 2 layers, and 300 units total, it was possible to achieve a 1.7% error in the test data. For the implementation, one could notice that sigmoid degraded the results if used as activation. Further improvements by changing learning rate adaptatively can be achieved.

REFERENCES

Basic format for books:

[1] GoodFellow, Ian, et al. Deep learning. Cambridge: MIT press, 2016.