Advanced Topics in Machine Learning Challenge 2

Piero Pettenà

January 2, 2024

Abstract

This challenges proposes a particular parametrization of the ReLU activation function for a neural network classifier. The goal is to learn the parameters of this formulation and understand what their final values mean. Implementation of the project can be found at [1].

1 Introduction

The proposed parametrization of the ReLU function for a specific layer is the following:

$$ReLU_\alpha(x) = ReLU(x) - (1-\alpha)ReLU(-x) \tag{1}$$

where $\alpha \in [0,1]$. When $\alpha = 1$ the resulting activation function is simply $ReLU_1(x) = ReLU(x)$, however, when $\alpha = 0$, the activation functions results in $ReLU_0(x) = x$. The various αs are collected in a vector which will be denoted by $\boldsymbol{\alpha}$.

In order to learn the parameter $\boldsymbol{\alpha}$, the loss function has to be updated as follows:

$$\hat{\mathcal{L}} = \mathcal{L} + \lambda ||\boldsymbol{\alpha}||_1 \tag{2}$$

where $\mathcal{L}$ is the chosen initial criterion, i.e. the Cross Entropy Loss for this challenge, $\lambda \in \mathbb{R}$ is a hyperparameter for the loss function.

2 Architecture of the neural network

The architecture of the classifier neural network is based on the example of section 10.2.7 of [2], which is a classifier network for the MNIST dataset. The same number and type of layers are used, i.e. 3 convolutional layers followed by a fully connected layer. The hyperparameters of the convolutional layers are shown in table 1.

The output of the third layer is reshaped into a vector of size 36, which is mapped to a 10 dimensional vector representing the output via the fully connected layer. The $ReLU_\alpha$ functions are placed after each convolutional layer, for a total of 3 activation functions and $\boldsymbol{\alpha}$ of size 3.

Layer	Input channels	Output channels	Kernel size	Stride
Layer 1	1	15	2	2
Layer 2	15	15	2	2
Layer 3	15	1	2	1

Table 1: Caption

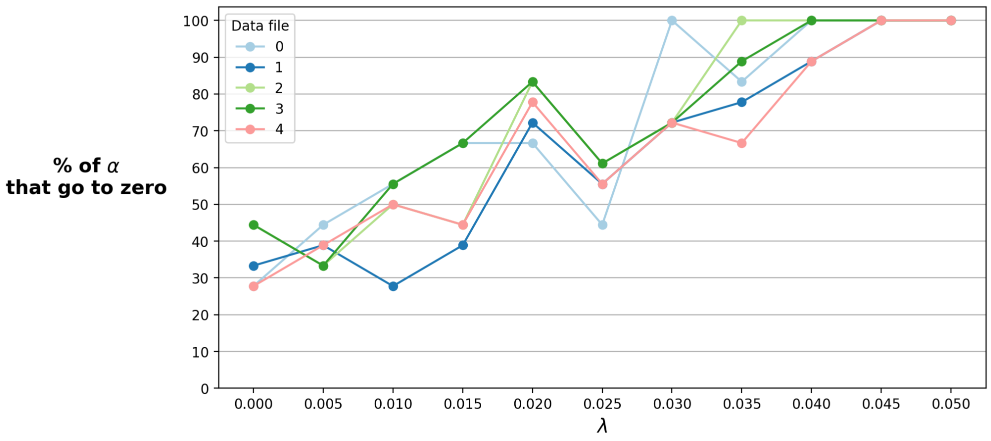

Figure 1: Percentage of αs that go to zero for each λ

3 Methods

Training has been performed on subsamples of 10000 images from the popular Fashion MNIST dataset [3]. The batch size is 64 and the number of epochs for training is 10, with a learning rate of 0.01 while the momentum is null. Multiple networks have been trained with different values for λ in the arbitrary interval $[0, 0.05]$ with a step of 0.005. The interval was chosen because higher values for λ appeared to always converge $\boldsymbol{\alpha}$ to zero. Initial values for each element of $\boldsymbol{\alpha}$ have been randomly picked from the set of numbers given by the division of interval $[0, 1]$ in steps of size 0.05.

4 Results

While resulting plots are not easy to interpret, the dependence on λ seems to be the most clear one.

Figure 1 indicates the percentage of αs that go to zero for each value of λ, for a specific training dataset. The plotted values are obtained calculating the percentage of 6 runs for each λ value, each with random initial $\boldsymbol{\alpha}$. Every α_i is assumed to be "converged to zero" if $\alpha_i \leq 0.05$ after each epoch has been used. An example of 6 runs can be seen in figure 2.

Figure 1 shows that, in general, the higher values for λ force more alphas to collapse to zero. This is coherent with the loss formula (2), where a higher λ brings a higher penalization for the values of $\boldsymbol{\alpha}$. The data in figure 1, shows that there are common patterns for the different datasets. In particular, every datasets seems to decrease the percentage after $\lambda = 0.020$. The cause of this suspicious behaviour has not been identified.

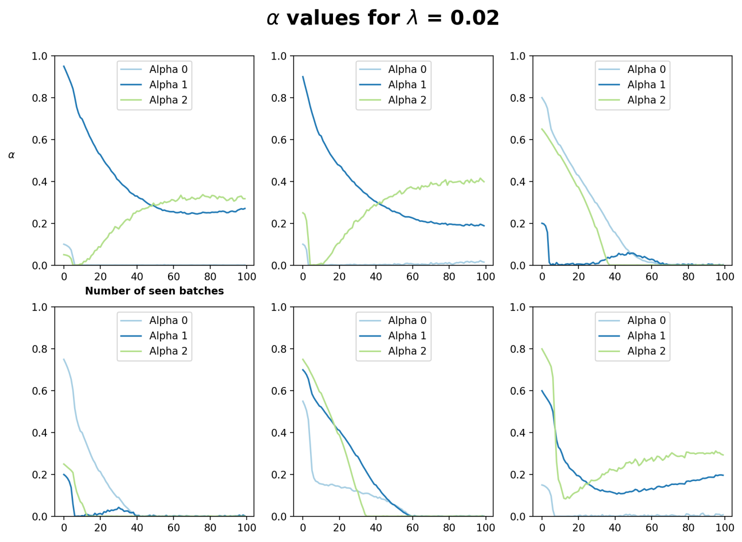

Figure 2: Trajectory of αs from 6 trained networks with $\lambda = 0.02$

Results show that only seldom, and for the smaller values of λ, the components of $\boldsymbol{\alpha}$ tend to converge to some intermediate value between 0 and 1. Figure 3 illustrates this. It can be noticed that α_0 rarely settles to 1 (only for $\lambda = 0.0, 0.005$). Since this is the parameter controlling the ReLU function after the first convolutional layer, this suggests that in some rare scenarios a linear regression is preferred over a classic activation function in that position of the network. Of course, only the lower values of λ allow this. Similar reasoning can be made for α_1 and α_2.

From the plots that have been reported as an example in figure 2, there is not an evident relationship between the starting point of $\boldsymbol{\alpha} \in [0, 1]^3$ and the convergence of its components.

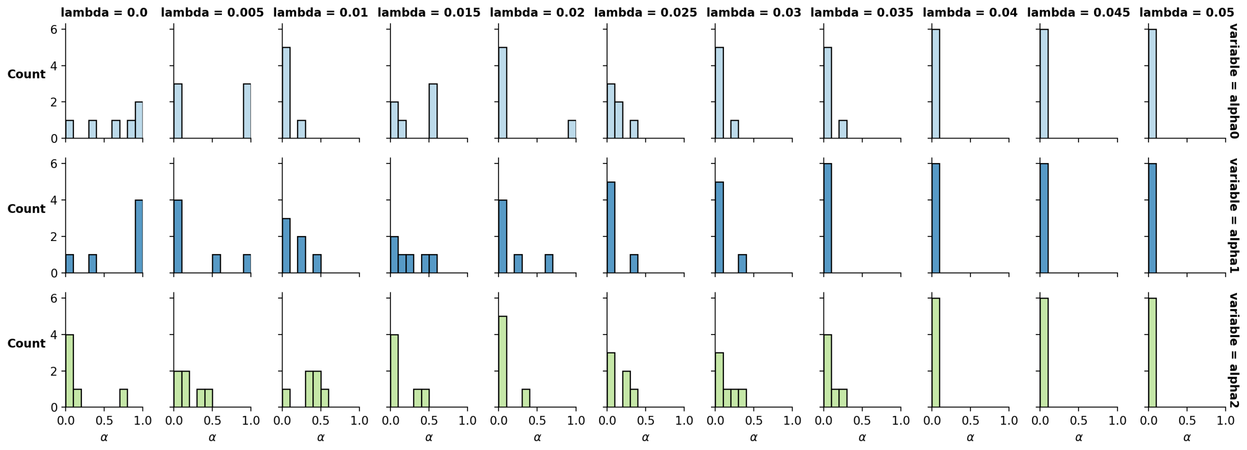

Figure 3: Histograms of α final values

Figure 4 shows the distribution of the final $\boldsymbol{\alpha}$ values with respect to λ. It shows that there is not a clear enough division of the points to perform clustering on them. It is important to

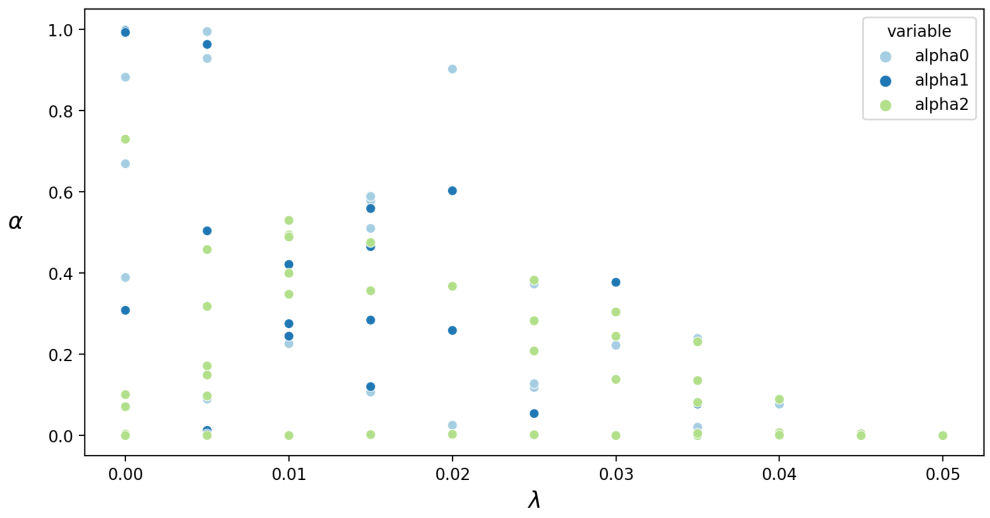

Figure 4: Scattered distribution of final alpha values with respect to λ

notice, however, that an increase in the number of epochs could determine different results as the final values might settle and create natural clusters. This was not tested because of computational and time constraints.

That being said, figure 4 shows that the further along this specific network, the less likely α goes to 1, i.e. the activation function is modified to a compromise between a linear function and the classic ReLU function. This can be noticed in the figure as α_2 never collapses to 1.

References

[1] Piero Pettenà. *Parametrized ReLU classifier*. URL: https://github.com/pettepiero/parametrized-relu-classifier. (accessed: 19/12/2023).

[2] Simon J.D. Prince. *Understanding Deep Learning*. MIT Press, 2023. URL: https://udlbook.github.io/udlbook/.

[3] Zalando. *Fashion MNIST*. URL: https://www.kaggle.com/datasets/zalando-research/fashionmnist.