First Assignment Report

Student: Isac Pasianotto

2024-05

0 Requirements

The aim this assignment is to (try to) reproduce the experiment reported in *"Figure 1."* of the paper [1].

That consist in implementing a small neural network to solve the following classification task: given a 6-bit long binary number as input, determine if this is palindromic number or not.

1 Network definition

The network described in the paper was composed by a neuron in the input layer for each digit of the number (i.e. six neurons in total), two neurons in the unique hidden layer and just one neuron in the output layer.
The authors provided also a figure (reported in 1a) of the network, which we can redraw with a more conventional representation as 1b.

Moreover, the authors provided the following additional information about the network:

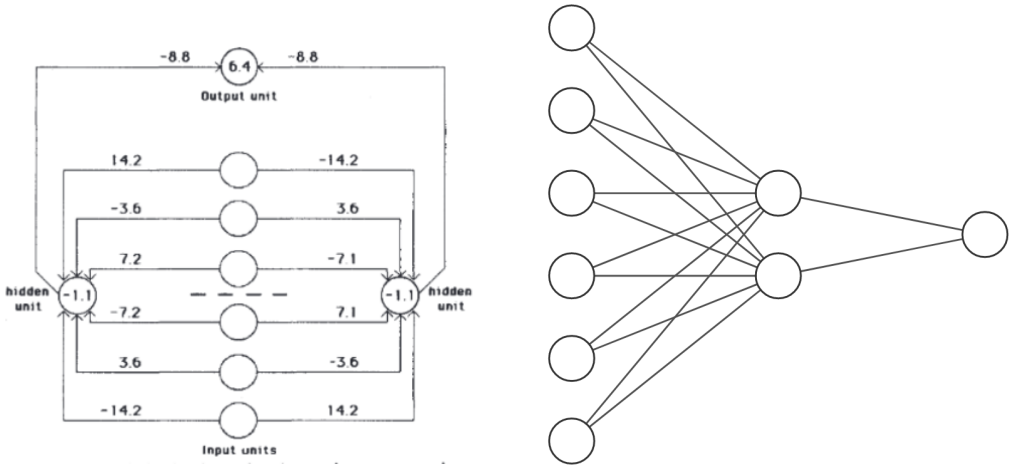
- The initial weights of the network were randomly chosen from a uniform distribution in the range $[-0.3,\ 0.3]$.
- They use the whole dataset (64 combinations of 6-bit binary numbers) to train the network.
- The used loss function was $E = \frac{1}{2} \sum_c \sum_j (y_{i,c} - d_{i,c})^2$.
- The learning rate was set to $\varepsilon = 0.1$ and the number of epochs the train lasted was 1,425.

What is missing in the paper is:

- The activation function used in the hidden and output layers. I assumed that the authors used the sigmoid function,since it was the only one mentioned in the paper.
- The initializations of the bias terms. I assumed that they were initialized with zero.

2 Results

Performing the experiment, trying to still as close as possible to the original formulation didn't lead to a very promising result, as shown in the figure 2.
I blame the unbalanced dataset for the unsatisfactory results, since the number of palindromic numbers is much smaller of the non-palindromic ones. In fact even if the accuracy seems to be



(a) Network representation presented in the paper.

(b) Network representation with a more conventional layout.

Figure 1: Network used in the experiment: 6 input neurons, 2 hidden neurons and 1 output neuron.

good, this is due to the fact that the network ended up into being a dummy classifier, always predicting the most frequent class.
Considering a balanced dataset (repeating the palindromic numbers more than once to match the number of non-palindromic numbers) lead to a much better result, as shown in the figure 3.

3 Variations on the original experiment

Since we were encouraged to try different variations of the experiment, I tried to train the network in several ways, which can be summarized in the table 1.

description	balanced dataset	learning rate	hidden neurons	loss Function	activation Function	optimizer	figure
Original experiment	no	0.1	2	E	sigmoid	GD	2
Balanced dataset	yes	0.1	2	E	sigmoid	GD	3
double learning rate	no	0.2	2	E	sigmoid	GD	4
halved learning rate	no	0.05	2	E	sigmoid	GD	5
4 hidden neurons	no	0.1	4	E	sigmoid	GD	6
softmargin loss	no	0.1	2	softmargin	sigmoid	GD	7
ReLU activation	no	0.01	2	E	ReLU	GD	8
Adam optimizer	no	0.01	2	E	sigmoid	Adam	9

Table 1: Summary of the experiments.

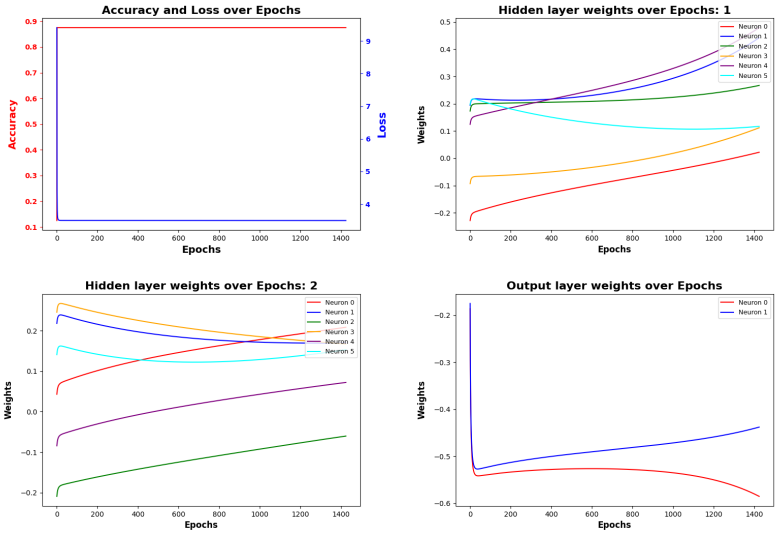


Figure 2: Original experiment results.

References

[1] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. "Learning representations by back-propagating errors". In: *Nature* 323 (1986), pp. 533–536. URL: `https://api.semanticscholar.org/CorpusID:205001834`.

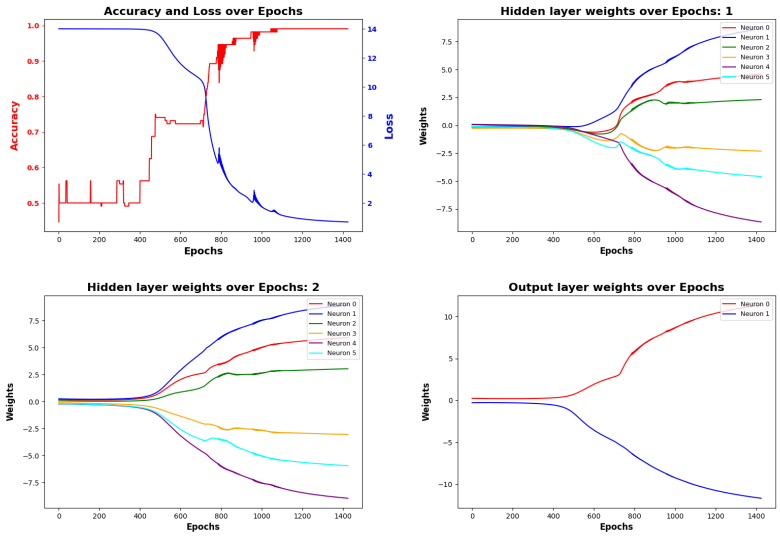


Figure 3: Original network, but with a balanced dataset.

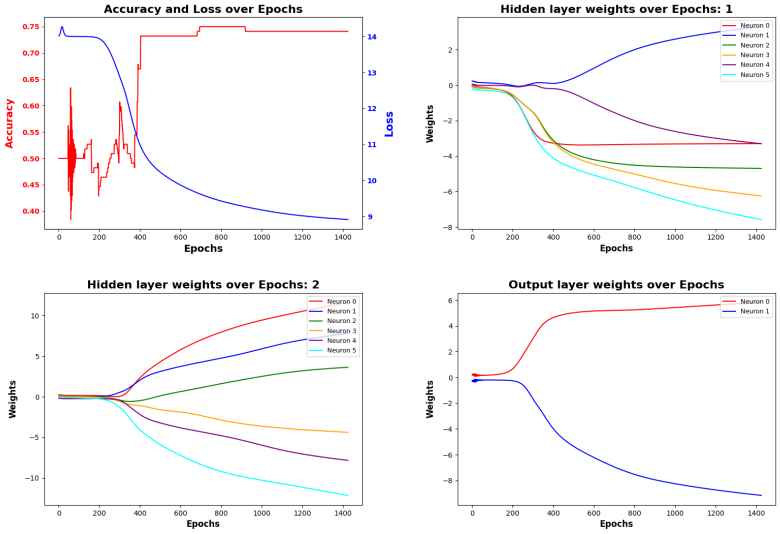


Figure 4: Original network, but with doubled learning rate.

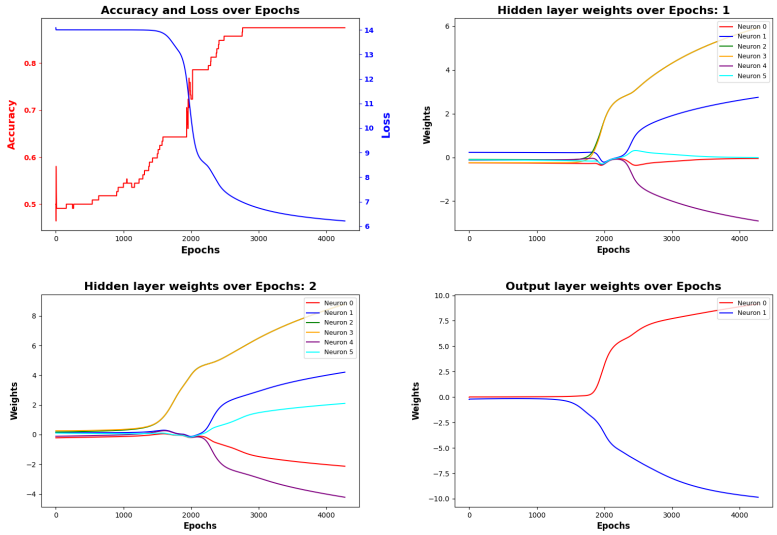


Figure 5: Original network, but with halved learning rate.

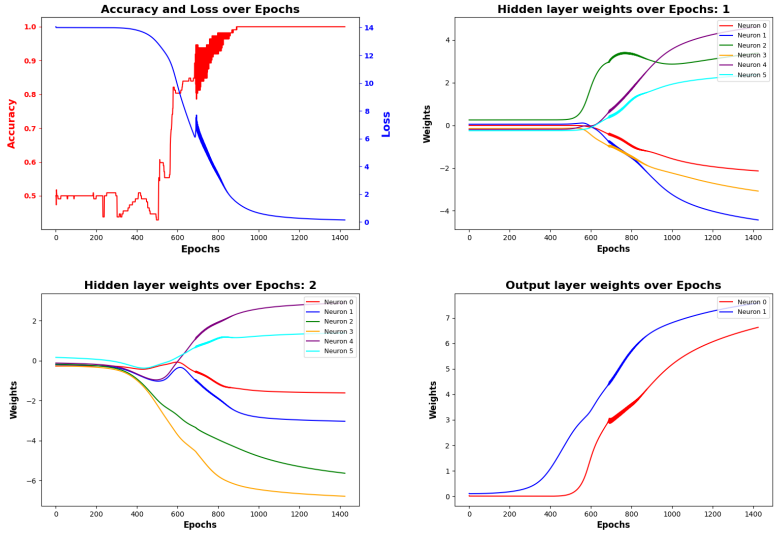


Figure 6: Network with 4 hidden neurons. In the plot are reported only the hidden layer weights of the first 2 neurons.

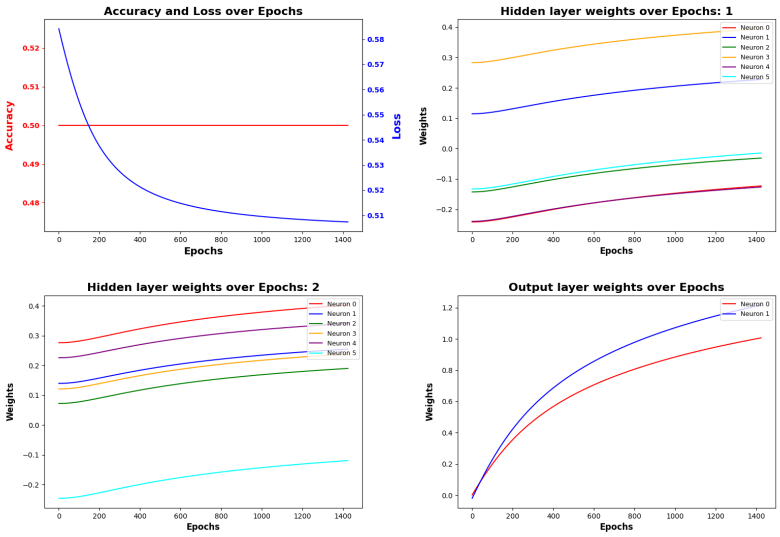


Figure 7: Original network, but using the softmargin loss function.

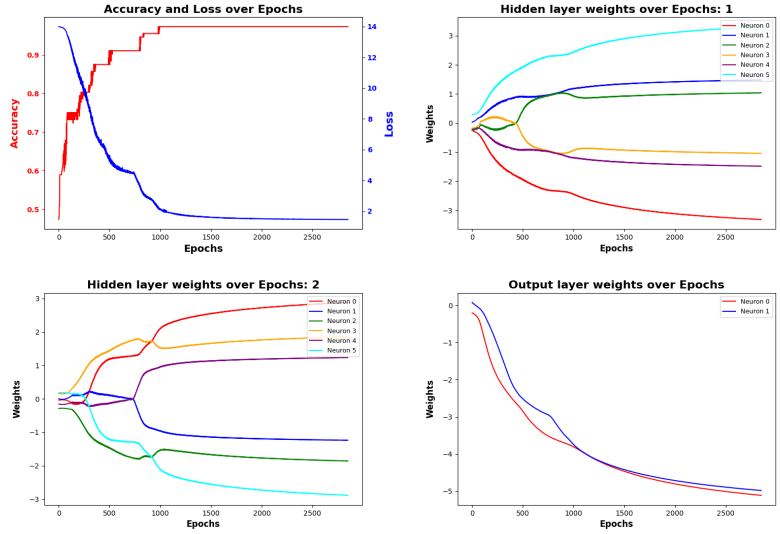


Figure 8: Network in which the hidden layer has the ReLU activation function.

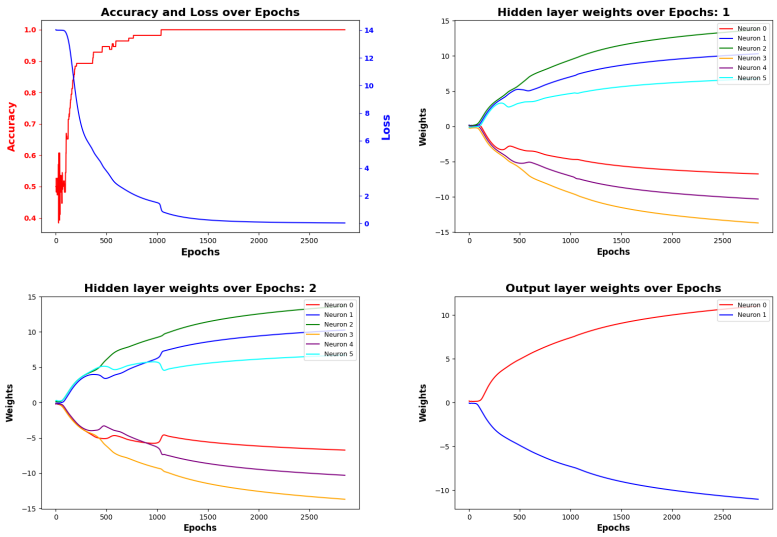


Figure 9: Original network architecture, but using the Adam optimizer.