Neural Networks Network

Marco Tuccio - Progetto Computazionale per l'esame di Reti Neurali del Prof. M. Osella - Università di Torino
https://github.com/MarcoTuc/NNN_Project
July 15, 2021

Abstract

The current neuron model used in MLP neural networks is an "integrate and fire" collapsed point in space: the perceptron. Human neuron's dendrites are spatially distributed and have been shown to perform very complex nonlinear operations over synaptic inputs, rendering the neuron more than just a point in space.
If one wants to take inspiration from Mother Nature in the design of Artificial Neural Networks, the perceptron could therefore be an over-simplification. In this project I investigated the performances of a neural network equipped with "smart neurons" and compared its performances with equivalent MLP neural network architectures.

1 Introduction

Dendritic arborizations are one of the most fundamental building blocks of neurons in the human brain. It has long been assumed that the summation going on at dendritic intersections can instantiate only OR and AND operations. By following these assumption, the perceptron model [2] for a neuron makes perfect sense. However, this assumption has largely been based on the fact that most of our knowledge about neurons has been acquired by studying the rodents' brain.

The canonical example of the limitations of neurons (as being modeled by perceptrons) is the incomputability of the XOR operation, which has long been thought to be feasible only as a network distributed computation. Fromherz and Gaede (1993) theorized the possibility for a single neuron to compute the XOR by signal annihilation on the collision of action potentials at dendritic intersections [4]. Gidon et al. (2020), have shown that a single human pyramidal neuron, which exhibits the most complex dendritic structures, is really capable of performing the XOR operation over its inputs [3].

In order to reproduce their complexity, human pyramidal neurons, have been modeled by Poirazi et al. (2003) as a two layer MLP neural network in which dendrites are subunits acting as perceptrons [1]. By updating the architecture of the neuron to an independent neural network, it is possible for it to compute the XOR function on its own. Figure 1 shows the simplest known MLP architecture capable of performig the XOR, this will be the minimal possible cell of the model here developed, henceforth referred to as the NNN (Neural Networks Network)

2 MLP Inception

The NNN Model can be thought of as the MLP brother of Google's Inception CNN [6], based on the Network in Network architecture of Lin et al [5]. Multiple convolutional filters, differently sized, are applied in parallel to the same input and their output is concatenated and fed to the successive layer.

In the NNN Model, something similar happens, each layer is composed of neurons made of three three main parts: synapses, dendrites and the axon. The synapses are the interface of the neuron, taking inputs from the previous layer outputs, dendrites are the hidden layers of the neuron and the axon is the output layer, sending signals to successive layer neurons.

NNN is an MLP Inception: Neurons are MLPs and if considered as points they are forming a wider MLP architecture. Theoretically, this gives each neuron the freedom to implement its own activation function, because of the universal approximation property of the MLP.

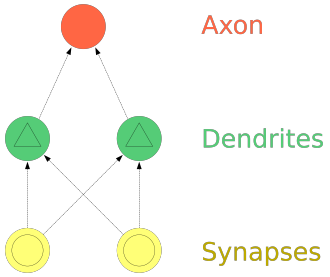

Figure 2: ***Neuron as dendritic network***
synapses: *receive signals from previous layer's axons*
dendrites: *internal hidden layer(s) of the neuron*
axon: *output of the neuron*

3 Forward Propagation

For each NNN cell, its output can be written as:

$$a_j^l = \gamma({}_j^l\zeta^{\Theta})$$

$$z^{l+1} = \omega^{l+1}a^l$$

with γ the axon activation function, l layer index, j neuron index, Θ is the number of dendritic layers of the neuron, ζ is the dendritic equivalent of z in an MLP network.
Equations for internal forward propagation for each cell are:

$${}_j^l\zeta^{\theta} = {}_j^l\alpha^{\theta}\ {}_j^l d^{\theta-1}$$

$${}_j^l d_{\beta}^{\theta-1} = \delta({}_j^l\zeta_{\beta}^{\theta-1})$$

with δ being the dendritic activation function, α being internal weights of the neuron. The synaptic interface equation is:

$${}_j^l d_{\beta}^{0} = \delta(z_j^l)$$

with z_j^l incoming to neuron synapses.According to these equations, the NNN model is fully connected, each synapse sees the whole input of previous layer and there is no sparsity implemented.

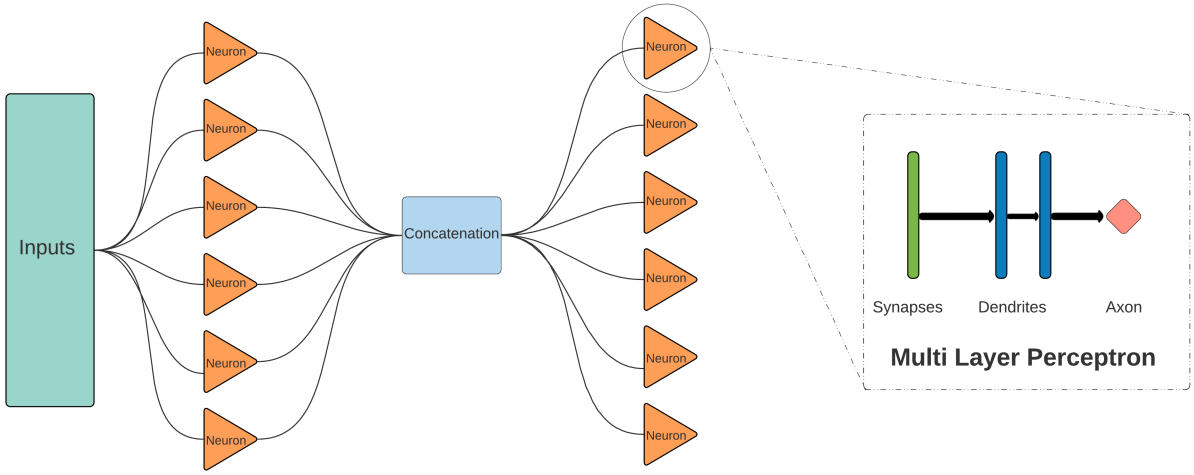

Figure 1: NNN Architecture as inplemented in Tensorflow

4 Errors Backpropagation

In order to train the model, one has to be able to calculate the gradient of the output loss function with respect to weights. In the following are the analytically derived backpropagation equations for the individual neuron by derivating loss function J over internal neuron dendritic weights α

$$\frac{\partial J}{\partial {}^l_j\alpha^\theta_{rs}} = \frac{\partial J}{\partial {}^l_j\zeta^\theta_r}\frac{\partial {}^l_j\zeta^\theta_r}{\partial {}^l_j\alpha^\theta_{rs}} = {}^l_j\square^\theta_r \; {}^l_j d^{\theta-1}_s$$

${}^l_j\square^\theta_r$ is the error of dendrite r at dendritic layer θ of the neuron and can be backpropagated as usual by taking as the "original error" the of the axon's one as the sum of synaptic errors Δ of the next layer:

$${}^l_j\square^\Theta = (\sum_\mu \Delta^{l+1}_\mu \omega^{l+1}_{\mu j}) \;\; \gamma'({}^l_j\zeta^\Theta)$$

The synaptic errors can be normally backpropagated between the two dense layers in which the l layer is represented by its axons and the successive layer $l+1$ is represented by the synapses (hence the sum is over $\mu \in synapses^{l+1}$).

5 Results: NNN vs MLP

Different NNN architectures have been tested on a categorical classification task on the Fashion MNIST database. Depht and wideness of neurons (and of network) have been tweaked in search of performance changes. All the other hyperparameters have been kept constant as:

1. Optimizer: RMSProp (standard tensorflow values)
   (a) Learning Rate = 0.001
   (b) $\rho = 0.9$
   (c) momentum = 0.0
   (d) $\epsilon = 10^{-7}$
2. Activation function: ReLU
3. Kernel Initialization: Glorot Uniform
4. Bias Initialization: Zero

Model	Accuracy	Time/epoch	Overfitting
Simple	0.8833 (0.8903)	230s (7s)	low (low)
Big Neurons	0.7316 (0.8903)	240s (7s)	- - -
Pyramidal	0.9200 (0.8858)	730s (7s)	very low (very high)
Deep	0.8913 (0.8818)	1080s (10s)	some (a lot)

Table 1: MNIST experiments on fully connected NNN compared to equivalent MLP performances (in brackets), everything has been trained for 30 epochs. For evaluating overfitting see the Appendix

The architectures have been implemented using the Tensorflow library and processed on an NVIDIA RTX 2070 graphics card and are described in [Table 2]

By looking at results in [Table 1], NNN is very slow if compared to an MLP, due to unefficiency in parallelization of the process: each neuron gets computed serially and then the output of all axons gets concatenated for completing a layer computation and sending values to the next layer. This is of course very slow when compared to a simple tensor multiplication for a simple MLP layer.

In early epochs, the accuracy of the NNN starts at very low values if compared to an MLP: better initialization is needed for the NNN. Accuracy manages to reach good levels nonetheless, but with no outstanding results compared to equivalent MLP architectures.

The only good result the NNN has been able to obtain

Model	Layers	Neurons	Neurons Shape	Params
Simple	2	20	[2,2,2,1]	33.050
Big Neurons	2	20	[200,100,100,10]	5.194.410
Pyramidal	3	784, 112, 28	1, [7,7,5,1], [4,4,3,1]	1.199.858
Deep	4	50	[5,5,5]	248,210

Table 2: Various tested NNN architectures.

is better overfitting performances, but only in the deep architecture and expecially in the pyramidal architecture.

The Big Neurons architecture has performed very poorly, there is huge loss of information happening inside neurons, because they weren't able to pass on this information through only 10 output axons. The fact that this model performed very poorly when compared to the simple one, which implemented the same network architecture but much simpler neurons, tells us that a certain balance between neurons size and network size has to be maintained.

6 Results: Convolutional NNN

A different testing round has been executed on Convolutional Neural Networks by comparing the performances of LeNet-5 in three cases:

- Normal LeNet-5
- LeNet-5 with NNN "tiny neurons" final layers (LNNN5(a)):
  1. 120 neurons of size [6,6,4,1]
  2. 84 neurons of size [4,4,3,1]
- LeNet-5 with NNN "medium neurons" final layers (LNNN5(b)):
  1. 120 neurons of size [30,30,20,10,1]
  2. 84 neurons of size [20,20,15,8,1]

Tests have been conducted on the same database (Fashion MNIST) and on the same hyperparameters of the previous test except for the optimizer: here I used Adam which is the optimizer of choice for LeNet-5.

A good result has been achieved by LNNN5(a) which managed to reach the same accuracy levels of the plain LN5 but increased the non-overfitting performances of the network. LNNN5(b) on the other hand has been overfitting a lot. Moreover on this particular network I needed to tweak the learning rate from 0.01 (TF standard) to 0.001 in order for it to converge. With the standard learning rate, this network indeed managed to reach good accuracy/loss levels on the first epochs, but around epoch 10 it started getting worse and worse with very high loss and very low accuracy.

7 Conclusion

The NNN model has not shown outstanding performances when compared to already existing models. The tests here conducted however have been very limited to the extent of fully testing the performances of the NNN model. Indeed one has to first write a better computational implementation of the model and perform many more experiments by going deeper in the trial process of network/neuron hyperparameters. It has to be noted that all other parameters have been kept constant in my experiments, such as learning rate, initializations and regularizations (which haven't been used). Activation functions have been ReLUs all over the network and all over inside neurons, it could be interesting to see what happens if one goes into using different activation functions on the various neuron layers. Another interesting experiment to be conducted is the application of dropout and connections sparsification at neurons and at network level. In the end it is also important to note that there are many other tasks for Neural Networks to be performed and image classification could not be the task in which NNN performs better than other models.

References

[1] Poirazi, Brannon, Bartlett, (2003), Pyramidal Neuron as Two-Layer Neural Network, (Neuron, Vol.37, 6, 989)

[2] Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. (Psychological Review, 65(6), 386–408.)

[3] Gidon, Zolnik, Fidzinski et al. (2020), Dendritic action potentials and computation in human layer 2/3 cortical neurons, (Science 03, Jan 2020 : 83-87)

Model	Ls/Acc	t/epc	oos L/A	Params
Plain LN5	.3076 .8985	7s	.4954 .8780	107,786
LNNN5(a)	.3604 .8806	875s	.3925 .8790	621,554
LNNN5(b)	.031 .9889	540s	.6479 .8985	3,318,470

Table 3: CNN vs "CNNN" performances

[4] P. Fromherz, V. Gaede, (1993), Exclusive-OR function of single arborized neuron, (Biological Cybernetics 69, March 1993 : 337-344)

[5] M. Lin, Q. Chen, S. Yan, (2014), Network in Network, CoRR, abs/1312.4400

[6] C. Szegedy et al. (2015), Going Deeper with Convolutions, proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 1-9

8 Appendix

Here I report plots from all the experiments for a quantitative/visual view of results. every model has been evaluated through plotting its loss and accuracy over training and validation sets and through a confusion matrix heatmap. (sorry for dark/light plots)

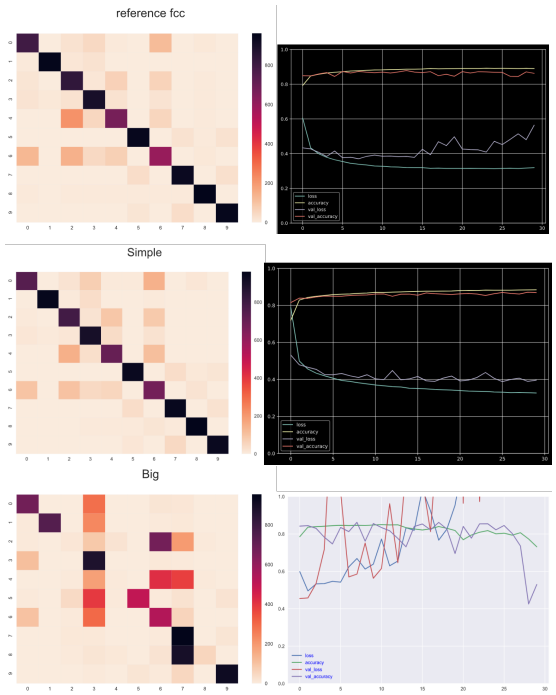

Figure 3: Performances of Simple and Big NNN architectures compared to reference

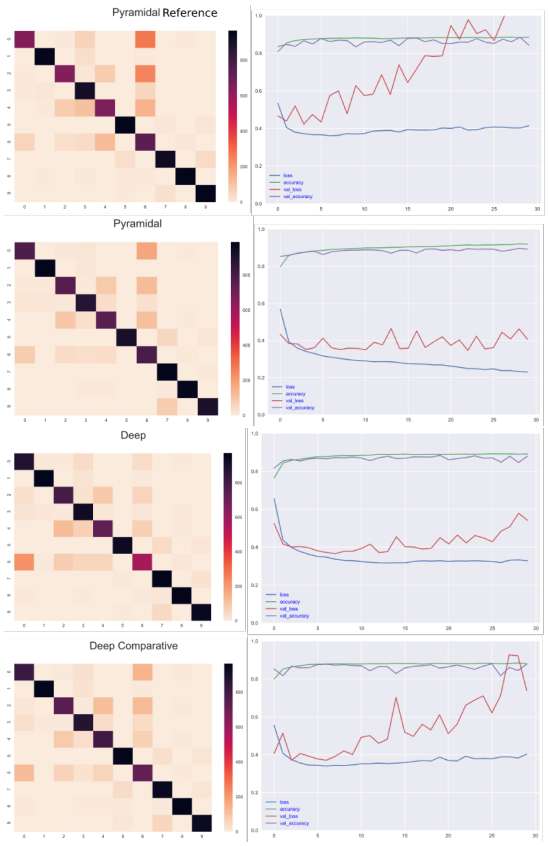

Figure 4: Deep and Pyramidal performances confronted to their reference

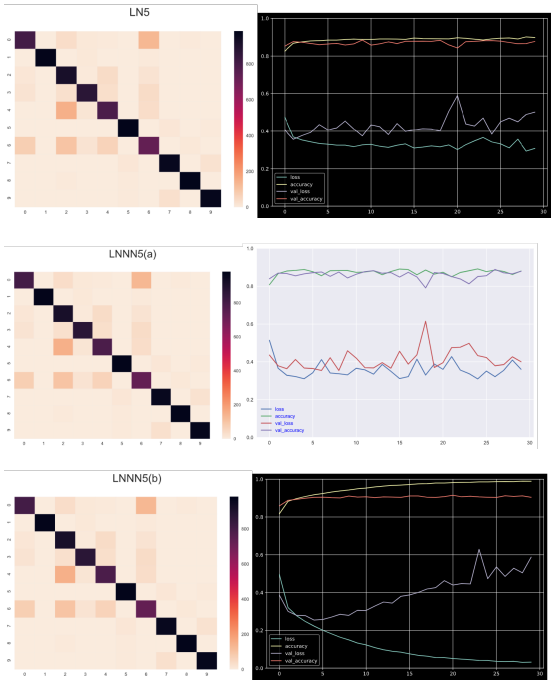

Figure 5: CNN vs "CNNN" experiments performances