GROUP01 – DEEP NEURAL NETWORK

Germán Cano Amaro, Sebastiano Monti, Valentina Tonazzo, and Elena Zoppellari
(Dated: March 19, 2022)

Deep learning is part of a broader family of machine learning methods based on artificial neural networks with representation learning. Deep-learning architectures such as **Deep Neural Networks** have been applied to fields including computer vision, speech recognition, natural language processing, machine translation, bioinformatics, drug design, medical image analysis, climate science, material inspection and board game programs, where they have produced results comparable to, and in some cases surpassing, human expert performance. The purpose of this review is to provide an analysis of the classification of a simple supplied dataset by means of a Deep Neural Network. By studying the latter and, in particular, the parameters that characterize it, it will be shown how it is possible to improve the forecast on given data and generalize to other datasets.

CONTENTS

The work is organized as follows: in the first part, the Net was studied with three modified versions of the initial dataset: reduced dataset, incremented dataset, and augmented dataset. Then, some gridsearches were implemented in order to improve more aspects and parameters of the model, and also it was checked if other data rescalings or initializations may improve the accuracy of the DNN. Finally, a new different dataset was studied and compered with the previous results. In addition, an alternative Neural Network architecture was taken into consideration. It was studied using the initial dataset and achieved results comparable with the previous ones.

INTRODUCTION

What are Deep Neural Networks? Neural networks are neural-inspired nonlinear models for supervised learning. In particular, neural nets can be viewed as natural, more powerful extensions of supervised learning methods such as linear and logistic regression and soft-max methods. The basic unit of a neural net is a stylized *neuron* i that takes a vector of d input features $x = (x_1, x_2, ..., x_d)$ and produces a scalar output $a_i(x)$. A neural network consists in many of such neurons stacked into layers, with the output of one layer serving as the input for the next. The first layer in the neural net is called the *input layer*, the middle layers are often called *hidden layers*, and the final layer is called the *output layer*. Modern neural networks generally contain multiple hidden layers (hence the *deep* in deep learning)[1].

Why do we study DNN? There are many ideas on why such deep architectures are favorable for learning. Increasing the number of layers, increases the number of parameters and hence the representational power of neural networks.

Scope of this review. Given a certain dataset and a DNN architecture, the main goal of this review is to use a typical Neural Network procedure to solve a supervised learning problem.

METHODS

Data The dataset is formed by a bi-dimentional set of 4000 samples representing the coordinates of points contained within a box of size $B = 100$, centered around the point $(0; 0)$. Points are randomly generated and classified with label $y = 1$ if they lay inside a triangle defined by Eq.1. Otherwise, they are classified with label $y = 0$. The generated points have been then rescaled in two different ways, in order to fit inside a box of dimensions $[-1; 1] \times [-1; 1]$ with the purpose of reducing calculations time and ease the classification. The first way consisted in simply dividing each sample's coordinates by 50, while the second, consisted in normalizing each sample's coordinates by the mean and the standard deviation of the entire set's coordinates. As can be seen in fig.1, the triangle formed by the data is incomplete, in sense that it does not form a complete figure, but just a partial picture of it. Two more datasets have been then generated in the same way as above, with the only difference in the number of contained samples. The reduced set contained a total number of 400 samples, wile the increased one, contained instead an amount of 10000 samples. Another last dataset has been instead generated using data augmentation technique. To do this, some small Gaussian random noise with zero mean and standard deviation of 0.7 has been added to each of the coordinates in the first

dataset. The analysis has finally been extended to a different non linear function, described by Eq.3, with a more irregular and complex point classification pattern. Such function can be observed in Fig.2. The main purpose of this project is to find an appropriate Neural Network for each dataset, capable of predicting data in order to complete the visual image efficiently. For this purpose, each dataset has been subdivided into a training set (80%) and a validation set (20%).

Neural Network Architecture After generating the points, neural network's architecture has been created using a function, in order to build the most general template possible for the network. This function receive as inputs the number of hidden layers composing the network, the number of neurons for each hidden layer, the type of activation function, the fraction of neurons used for the dropout and the kernel initializer. Since the task is classification, the chosen loss function for the architecture is binary crossentropy, which is the most widely used in these types of problems. In the last layer, only one neuron has been used, while, for the first hidden layer, two options have been chosen. For the majority of the analysis, the number of neurons in this layer was set equal to 2, as the supplied architecture suggested. However, for a further analysis, that parameter was replaced with a higher value, precisely: the number of neurons in the hidden layers, obtained through the optimization of the network parameters with the grid search. This was done in order to investigate whether a higher accuracy in classification could be achieved.

Grid search description After creating the above function, the next step consisted in generating various iterated grid searches, in order to find the best input parameters for the neural network's architecture that led to the higher classification accuracy. Grid searches are in fact algorithms that, receiving in input a training set of samples and different values of the parameters that define a network's architecture, stored into a dictionary, they tries all the possible combinations of parameters, finding the respective accuracy, and thus the hyperparameters' tuning that leads to the best accuracy. Some of the parameters used to perform a grid search, as for example, the number of hidden layers composing the network, the number of neurons for each hidden layer and the fraction of neurons used for the dropout, have been tuned through repeated tests. Once the structure of the network was fixed, it has been possible to study its behaviour, performing other grid searches that combined other important hyperparameters, related to the algorithms underlying the network, as for example the type of activation function, the kernel initializer and the optimizer function. From the wide range of hyperparameters, the ones which performed better on the preliminary studies were chosen to study how they behaved in combination with the other parameters. The best combination set was the one that conformed our final model. For completeness, in Table I are listed all the tested algorithms, identified by their names used in Python.

Optimizer	Kernel initializer	Activation function
SGD	uniform	selu
RMSprop	lecun_uniform	relu
Adagrad	normal	elu
Adadelta	zero	
Adam	glorot_normal	
Adamax	glorot_uniform	
Nadam	he_normal	
	he_uniform	

TABLE I. List of algorithms tested with grid search.

RESULTS

First non-linear function

The first and most general analysis was conducted on the data created using the first non linear function, which for every i-th row of the array of data (whose dimension was $N \times 2$) was defined as:

$$f(x,y) = \begin{cases} 1 & \text{if } x > -20 \wedge y > -40 \wedge x + y < 40, \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

The number N of samples which generate the points was $N = 4000$. In the very first analysis, each sample's coordinates were re-scaled by the maximum absolute value of the sample (50), therefore the re-scaled data lied into the range $[-1, 1]$ and in this way all the weights of the deep neural network are assured to be of a similar order of magnitude. The first neural network model created, assigned to each layer the same number of neurons *neurons/layer* except for the first hidden layer, whose value corresponds to $N_s = 2$. Using as parameters: 20 *neurons/layer*, 3 layers, ReLU (Rectified Linear Unit, defined as $ReLU(x) = max(0, x)$) as activation function, Adam as optimizer and a dropout probability of 0.2, the model was tested on the N data but also on the reduced, increased and augmented dataset, obtaining the results reported in Table IX.

Original	Reduced	Increased	Augmented
0.995	0.875	0.995	0.964

TABLE II. First model accuracy for different N.

In order to increase the accuracy of the model for all the four cases, a search was carried out using grid

searches to find better hyperparameters. The results are reported in Table III, providing an accuracy for the model of $\alpha = 0.996$. The number of layers found is really high (5), therefore they could have produced overfitting, on the other hand it should not be too consistent since the number of parameters is 2917. The best activation function is SELU (Scaled Exponential Linear Unit) which is defined as

$$f(x) = \begin{cases} x & \text{if } x \geq 0, \\ \lambda\alpha(\exp(x) - 1) & \text{if } x < 0, \end{cases} \tag{2}$$

where $\alpha \approx 1.6733$ and $\lambda \approx 1.0507$. This function shares with ReLUs and ELUs the capacity of not saturate for large inputs. The best optimizer is Nadam, which is a variation of Adam which incorporates Nesterov momentum, returning a better convergence quality than Adam. One last comment is made about He-Normal distribution for the initialization of weights. It draws samples from a truncated normal distribution centered in 0 with $\sigma = \sqrt{2/w}$ where w is the number of input units in the weight tensor.

Hyperparameters	Best value
Hidden layer/size	2,30,30,30,30
Batch size	50
Activation	*selu*
Optimzer	*Nadam*
Drop rate	0.1
Initializer	$He - Normal$

TABLE III. Hyperparameters for the data generated with the first non-linear function.

As described before, in this first model the first hidden layer had a number of neurons per layer equal to two. Aiming to try to further improve the accuracy, the model was changed imposing to every layer the same number of neurons. The resultant accuracy is $\alpha = 0.989$ for $N = 4000$, therefore, for the hyperparameters found, the previous model is better. This could be explained by the high number of layers chosen (5) which apparently works better with a first hidden layer with two neurons.

In an effort to supplementary improve the first model, two different re-scales of data have been studied. Firstly, they have been standardized to a Gaussian with zero mean and unit variance, secondly data have been scaled in range $[0, 1]$. The resultant accuracy are reported in Table IV, where it can be observed that the second one increases accuracy. Effectively, re-scaling in a range [0,1] can reduce the standard deviations of features.

The hyperparameters and the best re-scale found are applied to the reduced, increased and augmented data, providing for the model the accuracies listed in Table V, where it can be observed that all the values have increased except for increased data.

Gaussian re-scale	Re-scale in range [0,1]
0.980	0.997

TABLE IV. Model accuracy for different data scaling

Original	Reduced	Increased	Augmented
0.997	0.962	0.980	0.966

TABLE V. Final model accuracy for different data set.

In figure 1 the results from the model trained with the best hyperparameters and re-scale are shown.

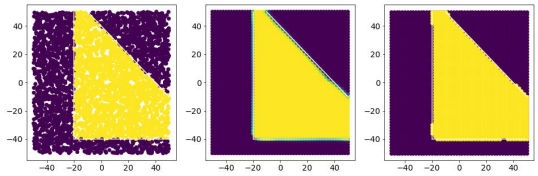

FIG. 1. Best model for the first function.

Second non-linear function

The second dataset was generated following the same structure as the first one but using a different function instead;

$$\begin{cases} 1 \;\; if \;\; sign(\sum x)sign(x)cos(\sqrt{\sum |x|^2})\frac{1}{2\pi} > 0 \\ \\ 0 \qquad otherwhise \end{cases} \tag{3}$$

As before, the first study was realized over 4000 samples generated between $[-50, 50]$. Firstly the model was trained normalizing data on $[-1, 1]$. With that re-scaling the study over hyperparameters that conformed the neural network started. First of all, a fixed set of neurons and layers of different sizes was established since neural network are very sensitive to them. Taking that into account, the combination of different sets of hyperparameters was studied, starting with the way we initialize the kernel, followed by the activation function of the neurons, the optimizer used for loss function optimization, the drop rate of neurons to avoid overfitting and the batch size also for optimizing the loss function. From this study the results listed in VI were obtained.

As can be observed, a priory, there is one clear value of neurons and layers that perform better with each hyperparameter individually, which coincide with the value with more layers between those tried. Usually more layers on our network allow to perform over more parameters and, in general, increase the representational power

Hyperparameters	Best individual value	Hidden layer/size
Batch size	10	$20, 20, 20, 20, 20$
Activation	$relu$	$20, 20, 20, 20, 20$
Optimizer	$Adam$	$20, 20, 20, 20, 20$
Drop rate	0.005	$20, 20, 20, 20, 20$
Initializer	$Glorot - Normal$	$200, 100, 50, 25$

TABLE VI. Description of individual study of hyperparamters.

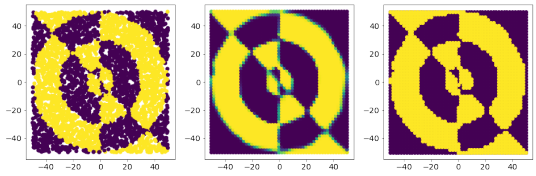

FIG. 2. Performance of the second model

of the network. Although, when taking into account the rest of hyperparameters conforming the model, it is possible to perform accurately over the data with less layers, as can be observed on the study of the best combination of parameters in Table VII.

Hyperparameters	Best value
Hidden layer/size	$200, 100, 50, 25$
Batch size	10
Activation	$selu$
Optimizer	$Nadam$
Drop rate	0
Initializer	$Glorot - Normal$

TABLE VII. Final hyperparameters of the second model.

It can observed how some hyperarameters perform better when combined with the others than individually, even though similar results were obtained for the values shown in VI. With this hyperparameters the final model was built and trained it again obtaining an accuracy of 96% over the test set and a performance that can be seen in Fig. 2.

After seeing the performance of the final set of hyperparameters, a study of their performance over the same dataset but with different sizes was made. For that three new datasets were generated, two of different sizes, other where noise was introduced and trained our model on them. The results obtained in VIII show how the hyperparameters perform with an accuracy higher than 85% over all the test sets, thus our model classify correctly the problem.

As it could be expected introducing noise on the dataset reduce the accuracy with respect to the others,

Original	Reduced	Increased	Augmented
0.966	0.987	0.953	0.869

TABLE VIII. Second model accuracies

as for the reduced dataset since the number of data is lower it is easier for the algorithm to fit them.

The last check on the performance of the final model pass by changing the normalization of data. Same re-scaling was applied as for the first model studied obtaining the following results, listed in Table IX.

Gaussian re-scaling	Re-scaling in range $[0, 1]$
0.910	0.956

TABLE IX. Accuracy for different data scaling

Close results to the first re-scaling were obtained but no better performance was observed. Machine learning algorithms are in general very sensitive with how data is scaled which can be observed on the differences between the scores obtained.

CONCLUSIONS

In this project, the aim was to create two deep neural networks models to classify two different datasets of points produced randomly and classified by different functions. The development of each model was carried out by the study of their hyperpameters and their performance in different situations as performing alone or in combination with other parameters. The accuracy of the models obtained following this procedure surpass in both cases the 95% of accuracy over the test set, which was a first indicator that both models correctly classify the problem without overfitting neither underfitting. In addition, several checks were made on the final models by studying their performance over datasets with different sizes, and different normalization on the data with an accuracy of at least more than 86%. With those results it can be stated that the found hyperparameters allow to create models that correctly classify the problem and generalize the performances.

REFERENCES

[1] P. Mehta, M. Bukov, C.-H. Wang, A. G. Day, C. Richardson, C. K. Fisher, and D. J. Schwab, "A high-bias, low-variance introduction to machine learning for physicists", Physics Reports **810**, 45–55 (2019).

APPENDIX

Organization of work:

- Germán Cano Amaro: Everything related to the second function study.
- Sebastiano Monti: Implementation of the function generating the neural network; preliminary trials for the grid search and writing of the methods section of the report.
- Valentina Tonazzo: Study of reduced, incremented and augmented datasets; introduction and preliminary explanations of report.
- Elena Zoppellari: Optimization of parameters for the first function study; weight initialization and re-scales study; writing of the first function's results in the report.