Applications of Neural Networks in Civil Engineering
Msc. Dissertation Summary

Paul Ionescu Constantin

October 5, 2018

The following document represents a summary of my master thesis defended at the Technical University of Civil Engineering of Bucharest in July 2018. Firstly, I would like to express my gratitude towards my supervisor, Professor Sorin Demetriu, for the encouragement and advice he has provided throughout my time as a student and during the research undertaken for this project.

Abstract

The thesis focuses on employing a class of artificial intelligence algorithms, namely neural networks in solving engineering problems. Hence it begins with a theoretical part (Chapters I-IV, from the full thesis) and continues (through Chapters V-VIII) with a series of four use cases for machine learning algorithms in solving civil engineering. The theory presented in the dissertation refers mainly to the working mechanism of feed-forward neural networks and to the validation techniques used to asses the model's accuracy. Furthermore, a brief description of recurrent neural networks was made, including also the subclass of LSTM cells (Long Short Term Memory) while preparing the ground for a problem regarding time series predictions.

1 Introduction

Artificial intelligence (AI) means briefly any form of human intelligence exhibited by a machine. For this, a programmer may define explicitly the rules through an algorithm, hence the responses could be determined prior to the execution of the code. On the other hand machine learning (ML) is a sub-field of AI in which the machine is provided with a series of inputs and example responses plus an algorithm to search for the rules corresponding to the example responses. Deep learning is a subclass of machine learning where the information is contained in a decentralized manner through a network of functional units called neurons. So instead of working with common type variables, say (x,y,z), the information in the model is defined by a series of operations (summations, multiplications, filter functions, etc.) applied to a set of numbers called weights, which can be represented as matrices.

A comprehensive work on applicability in engineering of this "paradigm shift" algorithms was done by H. Adeli [6] where there are depicted various subjects such as designing of steel beams or connections for structural steel elements, preliminary design of structures, design of concrete beams and slabs, etc. I. Flood et al. [8] approach in a gentle manner the inner workings of a neural network used to determine if yielding moment will develop on a cantilever steel beam under the action of a pair of forces. Inspired by these articles and the vast resources available in the machine learning field, I have built up an interest to further investigate how a civil engineer can make use of these methods to solve problems using open-source and modern software.

2 Deep Learning

Artificial neural networks have their architecture inspired from the natural neuronal networks found in the central nervous system of living beings. However, it is worth mentioning that they are a major simplification of their natural counterparts. Moreover, until now there doesn't exist a universally accepted proof for working similarity between them.

An artificial neuron consists in an input layer, a set of weights corresponding to each input plus a computing unit which executes a summation operation and a threshold function which will provide in the end a response/output. Usually, the first element from the input vector has the value of one and its corresponding weight is called bias. The adder operation sums up the element-wise product between the input vector and the corresponding weights. The obtained value, also known as the neuronal activity is provided to the threshold function. When dealing with classification problems, particularly when the output should be a categorical variable (i.e yes or no), the threshold functions employed outputs a response in a tightly bounded interval. Most commonly used for this problems is the sigmoid function also known as the logistic function. On the other hand, when solving regression problems, the threshold functions used are usually linear and unbounded. The most common for this case is ReLU (Rectified linear Unit) which outputs 0 for negative input and acts as the identity function for positive input values.

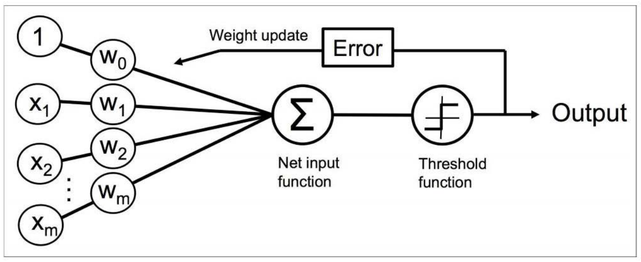


Figure 1: Graphical representation of an artificial neural neuron (perceptron)[16]

The main problem consists of finding the right weights so that, for an input, the neuron will provide the desired output/response to the considered problem. The idea is to find the right weights by having a large database of example input and output. Therefore the weights can be updated several times during iterations called training epochs until the model produces the desired accuracy. The working mechanism is described briefly below:

- the model is initialized by randomly drawing the weights, usually, from the standard normal distribution

- the database consisting of example input-output is split into training data and testing data (for example 75% + 25%)

- training will start by providing the sample input vectors from the database and collecting the output from the neuron

- a function called loss or sometimes cost function takes as inputs the neuron's output and the desired output from the example database and measures the model accuracy. Usually, the mean squared error (MSE) is used as a loss function

- each weight will be updated by adding a term which depends on the partial derivative of the loss function and a constant called learning rate. This method is called backpropagation

- the above procedure repeats for each input vector and for a fixed number of epochs or until the model has the desired accuracy on the test subset

- finally the model can be exploited on new inputs, outside the examples database

On the other hand, a neural network is made of densely connected neurons on multiple hidden layers within a carefully chosen topology that gives accurate results for the physical phenomena modeled. Detailed explanations about neural networks inner working mechanism can be obtained from dedicated books as [16],[22],[23], etc.

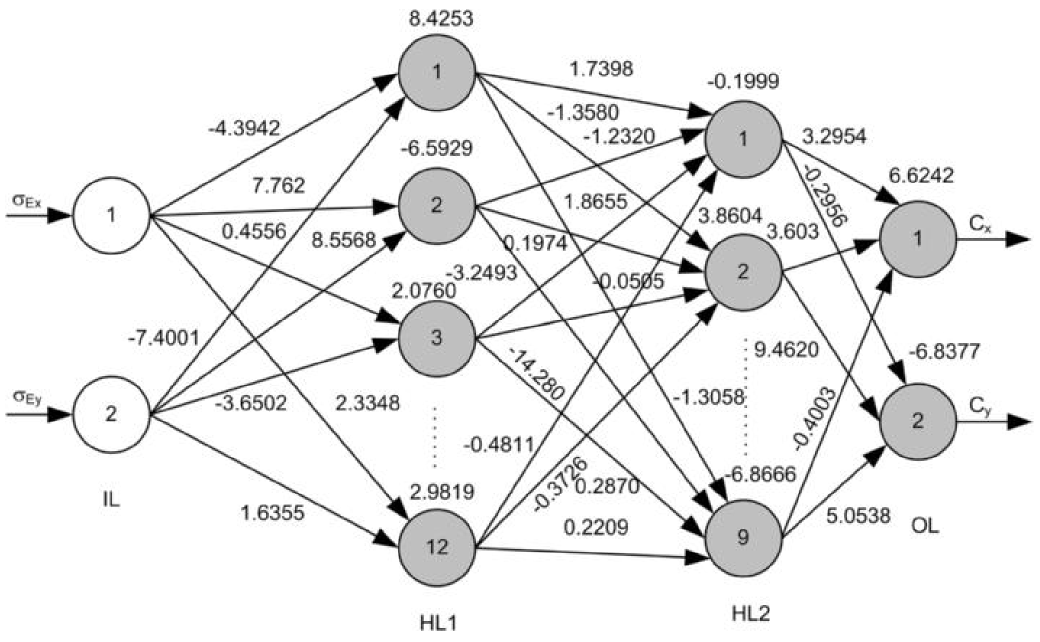


Figure 2: Example of a neural network with two input neurons, two hidden layers of 12 and respectively 9 neurons and two output neurons [19]

To improve the results, some of the techniques enumerated below are currently used:

- regularization of the weights - adding another term when updating them in order to reduce model complexity

- dropout - randomly removing neurons from the network by setting their neural activity equal to zero, thus increasing model generalization capability

- cross validation - splitting the database into train and test subsets multiple times and further train and test the model for each split. Thus the generalization capability of the model's topology and parameters can be evaluated. It also provides a better view of the network's accuracy

3 Experimental Setup

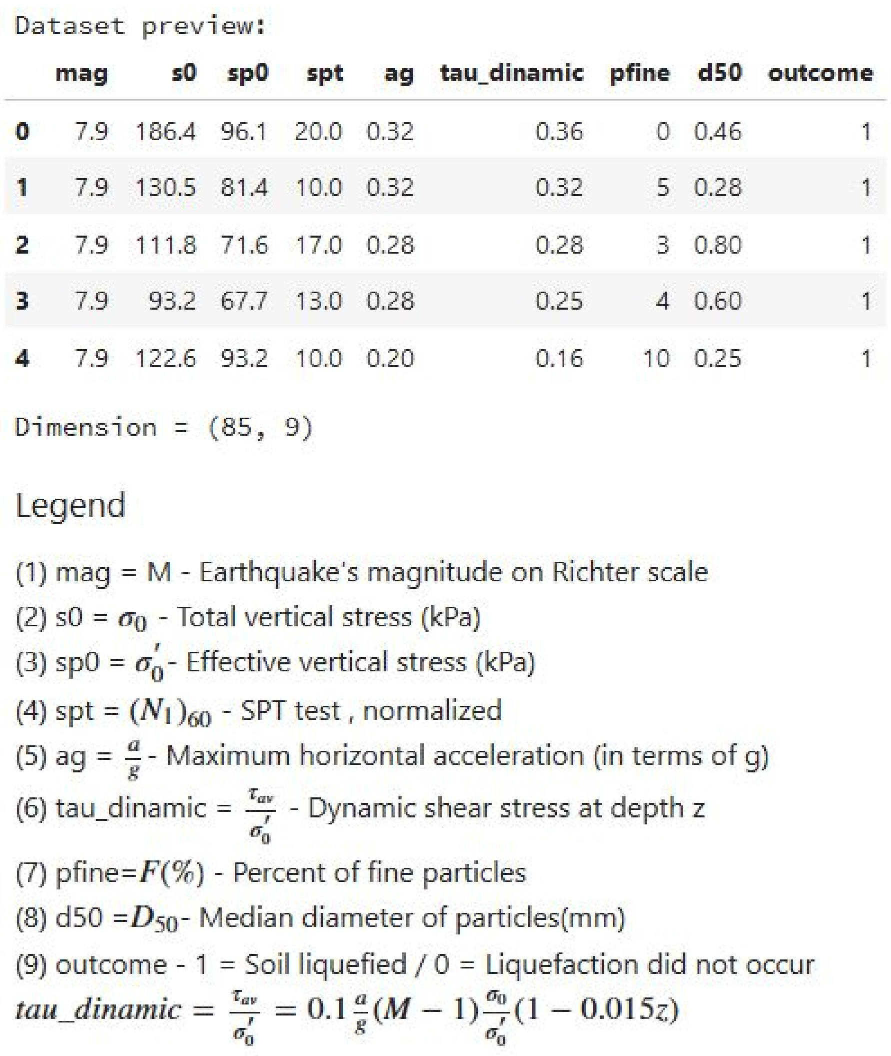
For developing the practical part of the thesis, Python programming language was used along with Jupyter Notebook and Jupyter Lab as environments to write and debug code. The open source libraries for Phyton most used in this thesis - which are worth mentioning - are:

- Numpy - scientific computing, parallel computing for matrices
- Pandas - data structures and data analysis tools
- Matplotlib & Seaborn - graph plots, exploratory data analysis
- Scipy.stats - statistical functions
- Scikit-learn (sklearn) - machine learning library
- Tensorflow - library for dataflow programming used for machine learning applications
- Tensorboard - visualization tools for Tensorflow
- Keras - high-level neural networks API used on top of Tensorflow

4 Applications

The practical part of the thesis consists in four applications comprising one problem concerning binary classification (soil liquefaction), two nonlinear regression problems (concrete compressive strength and deep beam capable shear force) and another one regarding time series predictions (sustained wind speeds).

Application I. Binary classification for soil liquefaction
In order to test the instruments employed, the first application replicates the results obtained in a study undertaken by another researcher. The database and the hyperparameters of the model were gathered from the following article **Seismic Liquefaction Potential Assessed By Neural Networks, A. T. C. Goh** [10]. The paper focuses on predicting soil liquefaction of terrain during earthquakes. In the end, the results form the study and the ones obtained in this application varied slightly due to the unknown random seed and method used by the author to generate the weights during network initialization.



```
Dataset preview:
```

	mag	s0	sp0	spt	ag	tau_dinamic	pfine	d50	outcome
0	7.9	186.4	96.1	20.0	0.32	0.36	0	0.46	1
1	7.9	130.5	81.4	10.0	0.32	0.32	5	0.28	1
2	7.9	111.8	71.6	17.0	0.28	0.28	3	0.80	1
3	7.9	93.2	67.7	13.0	0.28	0.25	4	0.60	1
4	7.9	122.6	93.2	10.0	0.20	0.16	10	0.25	1

```
Dimension = (85, 9)
```

Legend

(1) mag = M - Earthquake's magnitude on Richter scale

(2) s0 = σ_0 - Total vertical stress (kPa)

(3) sp0 = σ_0'- Effective vertical stress (kPa)

(4) spt = $(N_1)_{60}$ - SPT test , normalized

(5) ag = $\frac{a}{g}$- Maximum horizontal acceleration (in terms of g)

(6) tau_dinamic = $\frac{\tau_{av}}{\sigma_0'}$ - Dynamic shear stress at depth z

(7) pfine=$F(\%)$ - Percent of fine particles

(8) d50 =D_{50}- Median diameter of particles(mm)

(9) outcome - 1 = Soil liquefied / 0 = Liquefaction did not occur

$$tau\_dinamic = \frac{\tau_{av}}{\sigma_0'} = 0.1\frac{a}{g}(M-1)\frac{\sigma_0}{\sigma_0'}(1-0.015z)$$

Figure 3: Application I : Preview of the dataset [10]

Although the results of the current study are almost identical with the ones obtained in [10], it was observed that the same model will give different results when changing the training and testing subsets, hence obtaining variations in accuracy up to 15%. On the other hand, the random seed used to initialize the weights had also a considerable influence on the precision of the model.

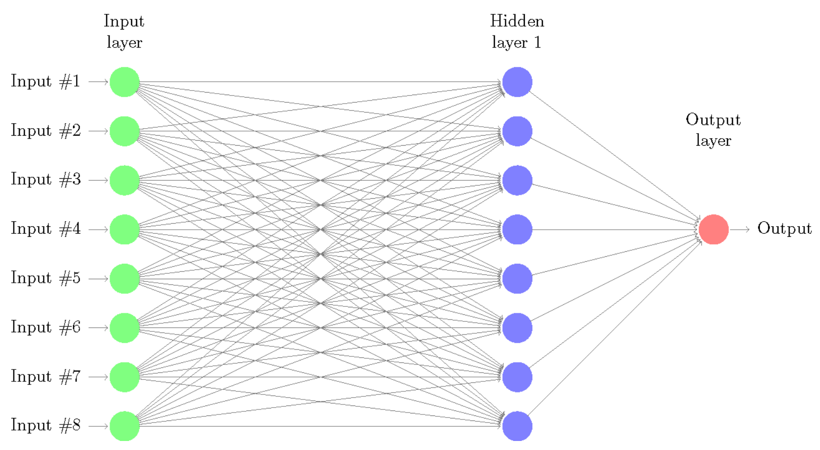


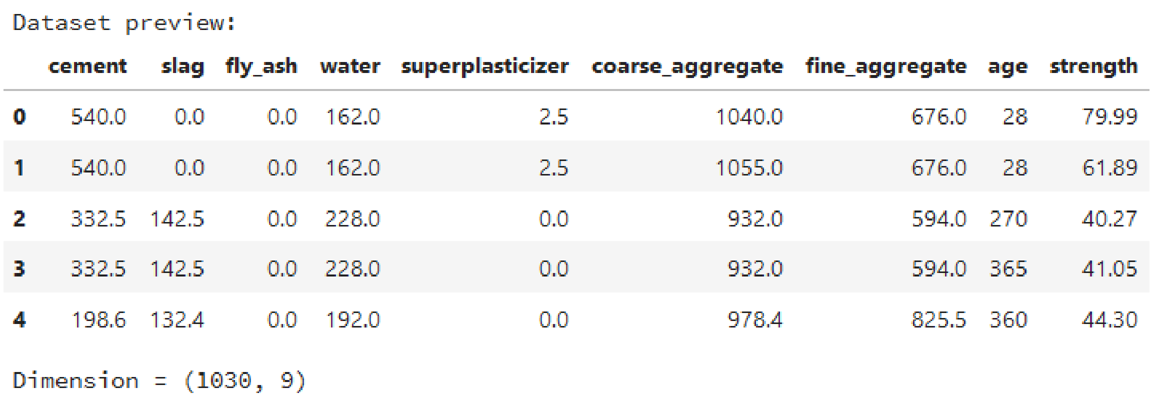
Figure 4: Application I : Network topology [10]

Accuracy	**Goh**	**Current study**
Training	97%	98.3%
Test	92%	92.3%
Overall	95%	96.5%

Table 1: Application I : Results

Application II.Nonlinear regression for concrete compression strength
This application approaches a common problem from the ML field, namely predicting concrete strength using the mixing recipe and the age as inputs. It is well known that the concrete compressive strength is a highly nonlinear function of age and ingredients.

The main focus consists of demonstrating the importance of features scaling. Because of working with input quantities that have different meanings and measuring units, for example time measured in days, strength measured in megapascals or cement content measured in kilograms while the neural networks are unit-less, a common practice is to use scaled input values (for example transforming them in standardized variables or by linear scaling in common intervals [-1,1], [0,1] etc.).


```
Dataset preview:
```

	cement	slag	fly_ash	water	superplasticizer	coarse_aggregate	fine_aggregate	age	strength
0	540.0	0.0	0.0	162.0	2.5	1040.0	676.0	28	79.99
1	540.0	0.0	0.0	162.0	2.5	1055.0	676.0	28	61.89
2	332.5	142.5	0.0	228.0	0.0	932.0	594.0	270	40.27
3	332.5	142.5	0.0	228.0	0.0	932.0	594.0	365	41.05
4	198.6	132.4	0.0	192.0	0.0	978.4	825.5	360	44.30

```
Dimension = (1030, 9)
```

Figure 5: Application II : Preview of the dataset [31]

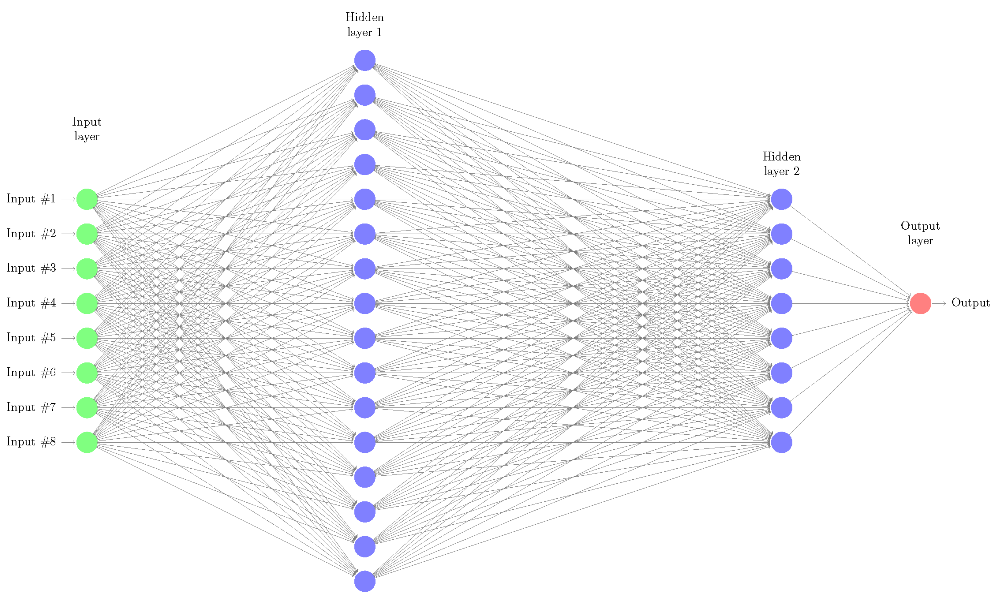


Figure 6: Application II : Network topology

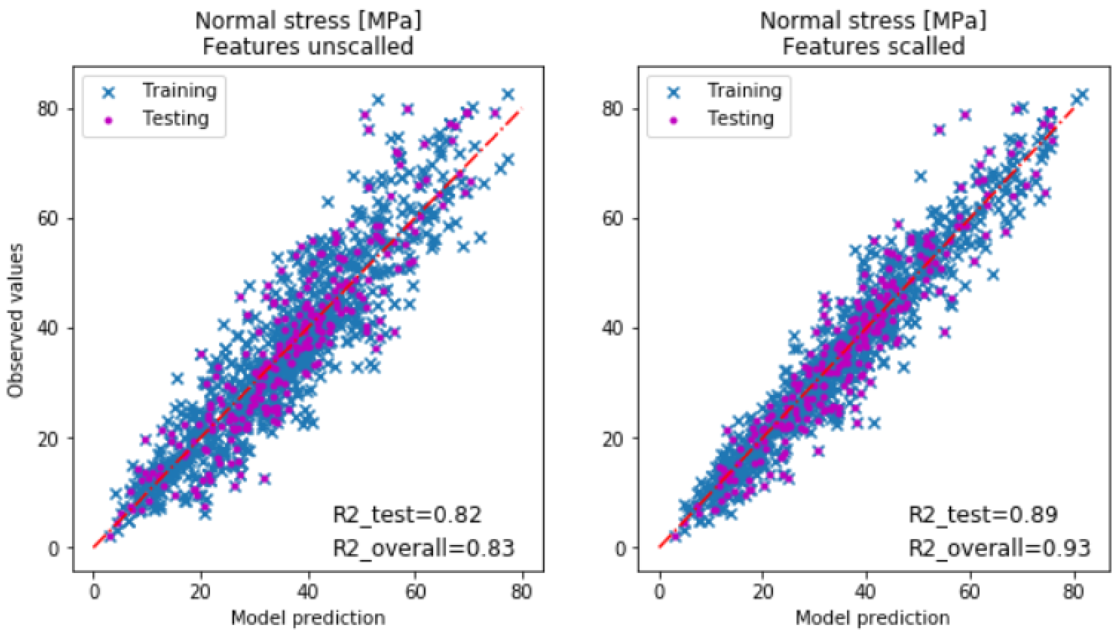
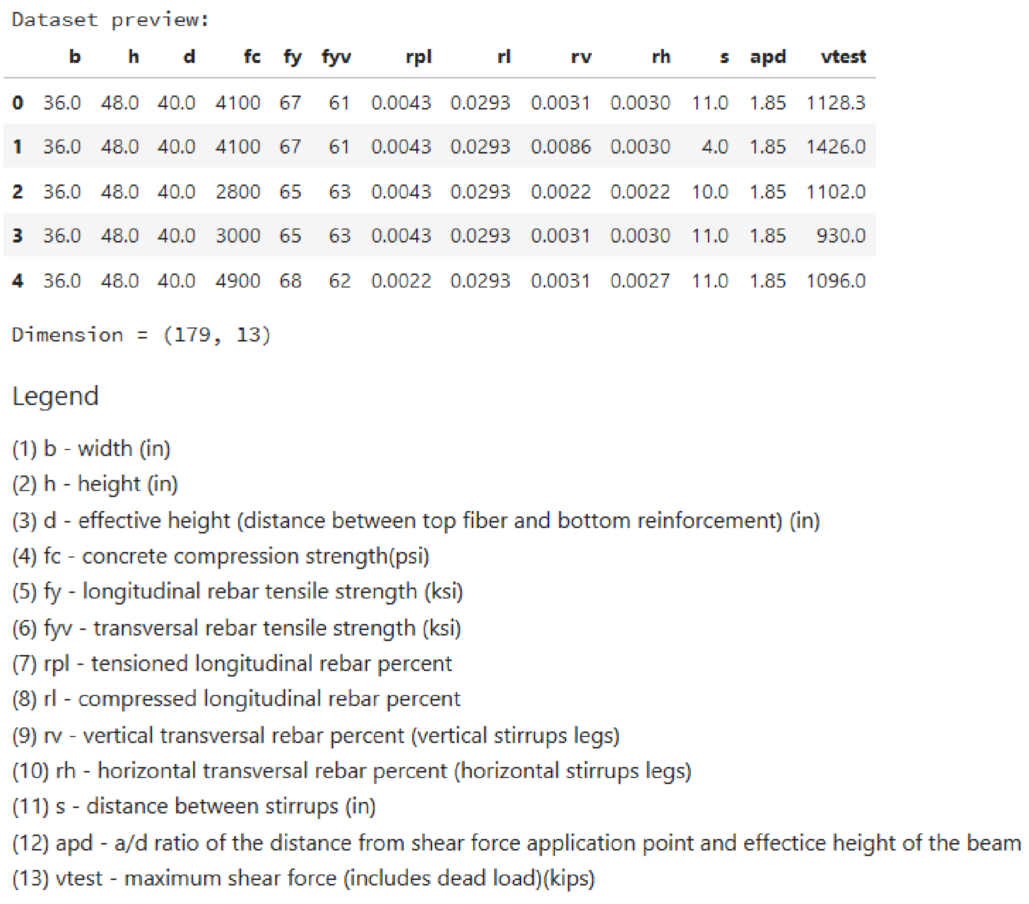


Figure 7: Application II : Results without/with feature scaling

It can be seen, as expected, that feature scaling plays an important role in the accuracy of the network. The model with scaled input also takes less time to converge. Due to the advantages of scaling the input, this procedure will also be applied to the following two applications.

Application III. Capable shear force prediction for deep beams
This application was inspired by the work of David Barra Birrcher in his PhD thesis, **Design of reinforced concrete deep beams for strength and serviceability** [32], which is also the source of the dataset used. The application focuses on finding a network topology that can provide better accuracy in predicting capable shear force developed in deep beams. The features taken as inputs are the geometry of the beam along with the type of concrete, reinforcement tensile strength and rebar percentages.

In order to find a desired network topology which can model the physical phenomena more accurately, a randomized search algorithm was employed. Thus, within this procedure up to 30000 neural networks were generated with different topologies containing one, two or three hidden layers and each of the hidden layers containing between 3 and 24 neurons. Each network was trained and tested on the same data sets for 1500 epochs. Then the best network in each category, namely the one with the lowest loss value was chosen and further trained until convergence.



```
Dataset preview:
```

	b	h	d	fc	fy	fyv	rpl	rl	rv	rh	s	apd	vtest
0	36.0	48.0	40.0	4100	67	61	0.0043	0.0293	0.0031	0.0030	11.0	1.85	1128.3
1	36.0	48.0	40.0	4100	67	61	0.0043	0.0293	0.0086	0.0030	4.0	1.85	1426.0
2	36.0	48.0	40.0	2800	65	63	0.0043	0.0293	0.0022	0.0022	10.0	1.85	1102.0
3	36.0	48.0	40.0	3000	65	63	0.0043	0.0293	0.0031	0.0030	11.0	1.85	930.0
4	36.0	48.0	40.0	4900	68	62	0.0022	0.0293	0.0031	0.0027	11.0	1.85	1096.0

```
Dimension = (179, 13)
```

Legend

(1) b - width (in)
(2) h - height (in)
(3) d - effective height (distance between top fiber and bottom reinforcement) (in)
(4) fc - concrete compression strength(psi)
(5) fy - longitudinal rebar tensile strength (ksi)
(6) fyv - transversal rebar tensile strength (ksi)
(7) rpl - tensioned longitudinal rebar percent
(8) rl - compressed longitudinal rebar percent
(9) rv - vertical transversal rebar percent (vertical stirrups legs)
(10) rh - horizontal transversal rebar percent (horizontal stirrups legs)
(11) s - distance between stirrups (in)
(12) apd - a/d ratio of the distance from shear force application point and effectice height of the beam
(13) vtest - maximum shear force (includes dead load)(kips)

Figure 8: Application III : Preview of the dataset [32]

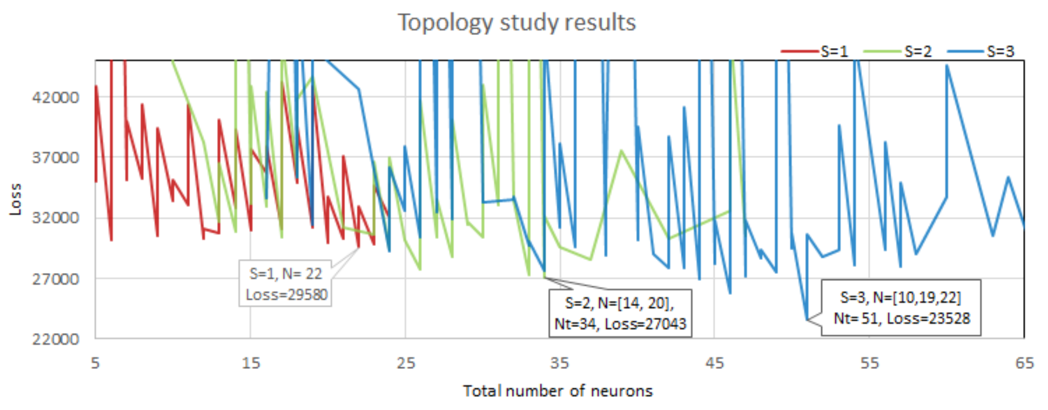


Figure 9: Application III : Parametric topology study results (red - one hidden layer, green - two layers, blue-three layers)

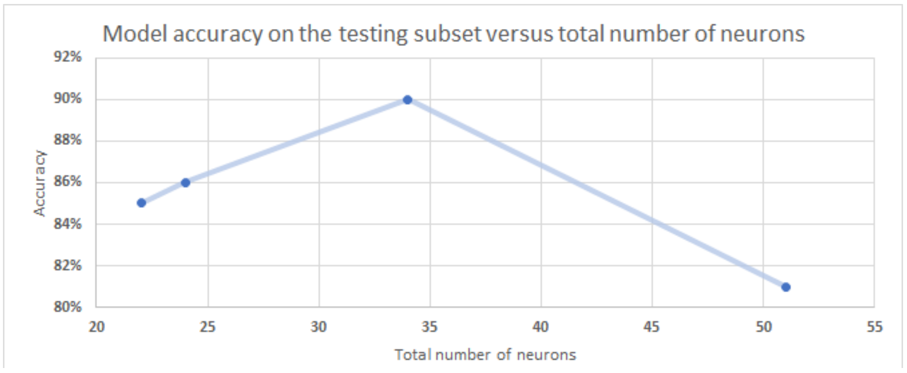


Figure 10: Application III : Model accuracy versus total number of neurons

In Figure 9 it can be observed that a large number of neurons doesn't conduct always to a greater accuracy of the network. When increasing the model's complexity, it will start overfitting the training values and will provide bad results for the new inputs from the testing subset. Also, the training duration will increase significantly. On the other hand, it can be observed in Figure 11 that the best topology will take considerable time to converge due to the fact that it doesn't encounter a local minimum in the early epochs of training. Another important finding is the ratio of the training time and the model's precision. Specifically to the studied problem, for a 1% increase in precision the network will need almost 10% more training time.

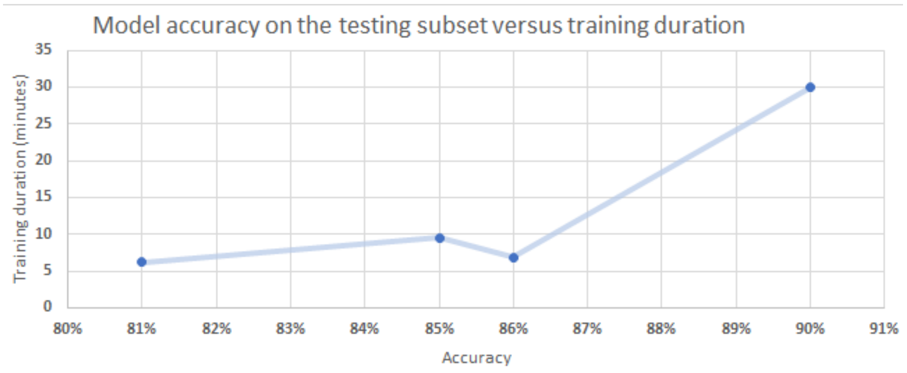


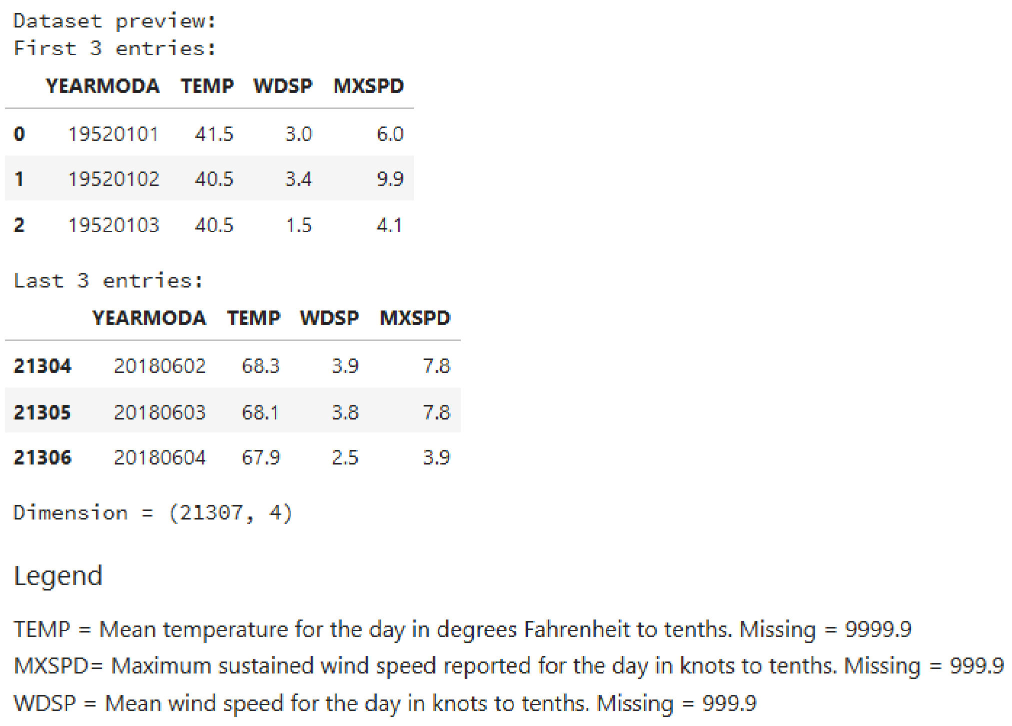
Figure 11: Application III : Model accuracy versus training time

Application IV. Sustained wind speed prediction using LSTM cells

This application focuses on employing LSTM (Long Short Term Memory) cells for predicting 10 future yearly peak values for the sustained wind speeds. This type of recurrent neural network (RNN) addresses the gradient vanishing problem by implementing two data pipelines for the input dataset: one for long-term dependencies and another one for the short-term ones. Weights are trained for both of them so that the network can retain relevant information from older inputs. Thus, they can influence predictions as much as the newer inputs, contrary to simple RNNs where newer inputs will mainly dictate the responses. The scope of this application does not consist of getting accurate predictions as long-term predictions for stochastic processes are impossible for now. Because of the reduced historical data available for annually sustained wind speeds, it wasn't justified to split the dataset into training and testing subsets. Nonetheless, the predicted values for sustained wind speeds will be used to compute the new quantile p=98% corresponding to the Gumbel extreme value distribution of maxima for the extended dataset (training data + 10 years prediction). The 98%quantile is important due to the fact that common practice consists in employing its value to determine the pressure exerted by the wind on buildings. Due to the limited computing capabilities, the neural networks were trained only with yearly values measured between 1973 and 2017.

The dataset contains three features for each entry: daily mean temperatures, daily maximum sustained wind speed and daily mean wind speed. The current application is divided into two parts:

- Developing a single feature LSTM neural network - will take as input only the maximum sustained wind speed
- Developing a multiple features LSTM neural network - will take as input also the mean temperatures and maximum sustained speeds (as yearly maximum values)



```
Dataset preview:
First 3 entries:
```

	YEARMODA	TEMP	WDSP	MXSPD
0	19520101	41.5	3.0	6.0
1	19520102	40.5	3.4	9.9
2	19520103	40.5	1.5	4.1

```
Last 3 entries:
```

	YEARMODA	TEMP	WDSP	MXSPD
21304	20180602	68.3	3.9	7.8
21305	20180603	68.1	3.8	7.8
21306	20180604	67.9	2.5	3.9

```
Dimension = (21307, 4)
```

Legend

TEMP = Mean temperature for the day in degrees Fahrenheit to tenths. Missing = 9999.9
MXSPD= Maximum sustained wind speed reported for the day in knots to tenths. Missing = 999.9
WDSP = Mean wind speed for the day in knots to tenths. Missing = 999.9

Figure 12: Application IV : Preview of the dataset [32]

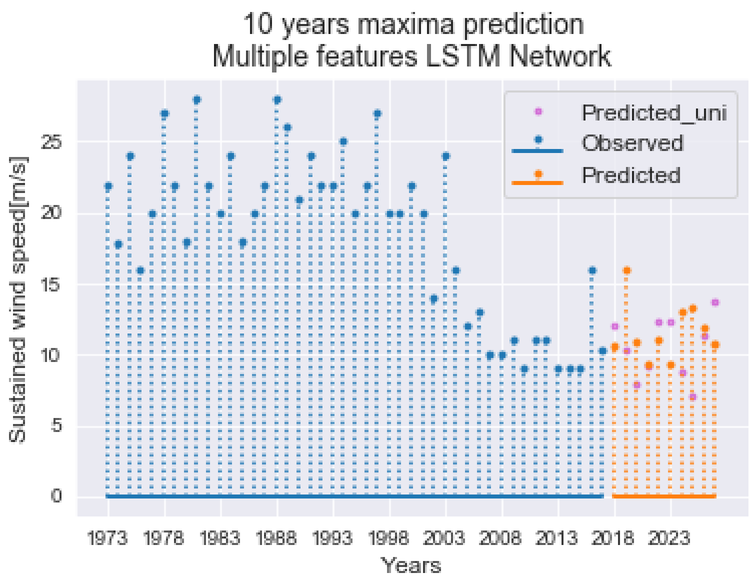


Figure 13: Application IV : Results. Prediction of sustained wind speeds for 10 years (Comparison between the two models)

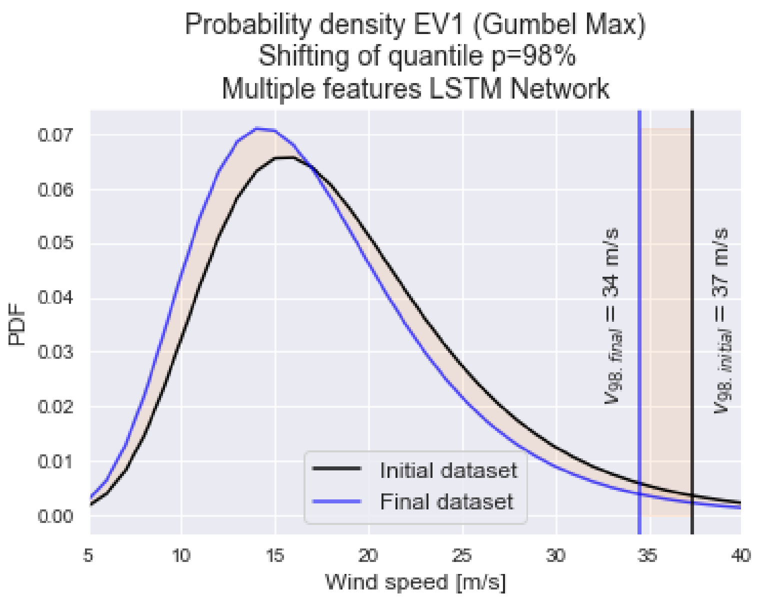


Figure 14: Application IV : Shifting of the p=98% quantile due to the extension of the dataset

As seen in Figure 9, the sustained wind speed corresponding to the 98 quantile decreased by almost 10% when computing it for a larger dataset. This type of approach can be used to study possible scenarios by simulating future values of a random process.

Source code
The source code and annotations for all the application can be previewed in a web browser or the Jupyter notebooks files can be downloaded from the following links:

- Application I. Binary classification for soil liquefaction
- Application II.Nonlinear regression for concrete compression strength
- Application III. Capable shear force prediction for deep beams
- Application IV. Sustained wind speed prediction using LSTM cells
- Link to the GitHub repository containing additional documentation

5 Conclusion

The use of neural networks is justified when having at disposal large quantities of historical or experimental data. This thesis stands as an example of employing with ease highly developed tools in solving engineering problems. Using methods for controlling the over-fitting and the model errors like topology search, cross-validation, dropout or regularization techniques one can find satisfactory results when dealing with highly non-linear phenomena for which physical models haven't yet been developed.

I believe the future holds great changes in the field of engineering. Probably after a short period as twenty years the activity this industry will be different altogether. Currently, the design and detailing of structures are still mainly driven by people employing machines to ease their volume of work. While the design problem is on the brink of being solved by programming explicitly the rules (i.e. CSi Etabs Design Modules), the detailing jobs consist, yet, mainly in manual labor of draftsmen. Being much harder to write down the detailing rules explicitly and having to deal with geometrical modeling which must work on unseen before or complex shapes, I believe that detailing of structure makes also a great candidate for implementing Machine Learning algorithms. Ultimately this changes will be transforming the designers in supervisors of machine output.

References

Corresponding to the full master thesis

[1] M. Nielsen, Cap 4, http://neuralnetworksanddeeplearning.com/chap4.html [visited on 04.06.2018]

[2] Universal approximation theorem, http://neuralnetworksanddeeplearning.com/chap4.html [visited on 04.06.2018]

[3] Technological singularity https://en.wikipedia.org/wiki/Technological_singularity [visited on 07.06.2018]

[4] Silicon Brains https://www.humanbrainproject.eu/en/silicon-brains/ [visited on 02.05.2018]

[5] Moravec's paradox https://en.wikipedia.org/wiki/Moravec%27s_paradox [visited on 05.06.2018]

[6] H. Adeli, Neural Networks in Civil Engineering: 1989-2000, Computer-Aided Civil and Infrastructure Engineering, vol. 16, pp. 126-142, 2001

[7] B. Benahmed, M. Hamoutenne, Use of the Artificial Neural Networks to Estimate the DRF for Eurocode 8, Periodica Polytechnica Civil Engineering, 06 04 2015

[8] I. Flood, N. Kartam, Neural Networks in Civil Engineering. I: Principles and Understanding, Journal of Computing in Civil Engineering, vol. 8, nr. 2, 1994

[9] I. Flood, N. Kartam, Neural Networks in Civil Engineering. II: Systems and Application, Journal of Computing in Civil Engineering, vol. 8, nr. 2, 1994

[10] A. T. C. Goh, Seismic Liquefaction Potential Assessed By Neural Networks, Journal of Geotechnical Engineering, vol. 120, nr. 9, September 1994

[11] GT 064:2011 - Ghid privind echiparea construcțiilor hidrotehnice de retenție cu aparatură de măsură și control - AMC, 2011 (Romanian design normative)

[12] A. Pandelea et al., Applications of Artificial Neural Networks in Civil Engineering, International Conference for PhD Students in Civil Engineering and Architecture - CE-PhD, 2014

[13] A. Pandelea, Diagnosticari in Domeniul Constructiilor Utilizand Retelele Neuronale Artificiale, Teza de doctorat, Iasi, 2017

[14] I. Caluianu, Cresterea productivitatii energetice a panourilor fotovoltaice, Teza de doctorat, Bucuresti, 2011

[15] F. CHOLLET, Deep Learning with Python, Manning Publications Co., 2018

[16] S. Raschka, V. Mirjalili, Python Machine Learning 2nd Edition, Packt, 2017

[17] https://askabiologist.asu.edu/neuron-anatomy, [visited on 01.06.2018]

[18] S. SHARMA, Activation Functions: Neural Networks, 6.09.2017, https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6 [visited on 08.06.2018]

[19] A. B. Raj et al., Intensity feedback-based beam wandering mitigation in free-space optical communication using neural control technique

[20] N. Lewis, Deep Learning Made Easy with R, 2016

[21] C. Zaharia, A. Cristea, Cristea - Algoritmi genetici si rețele neuronale, Editura Academiei Romane, 2002

[22] S. Haykin, Neural Networks A Comprehensive Foundation, Pearson Prentice Hall, 1999

[23] I. Goodfellow, Y. Bengio și A. Courville, Deep Learning, MIT Press, 2016

[24] D. Dumitrescu, H. Costin, Rețele neuronale. Teorie și aplicații, Teora, 1996

[25] S. Hochreiter, J. Schmidhuber, Long Short-Term Memory, Neural Computation, vol. 9, nr. 8, 1997

[26] C. Olah,Understanding LSTM Networks, http://colah.github.io/posts/2015-08-Understanding-LSTMs/ [visited on 01.06.2018]

[27] N. Srivastava, et. al., Dropout: A Simple way to Prevent Neural Networks from Overfitting, Journal of Machine Learning Research 15, 2014

[28] K.-L. Du, M. Swamy, Neural Networks and Statistical Learning, Springer, 2014

[29] A. Karpathy, 2018, http://cs231n.github.io/

[30] I.-C. Yeh, MODELING OF STRENGTH OF HIGH-PERFORMANCE CONCRETE USING ARTIFICIAL NEURAL NETWORKS, Cement and Concrete Research, vol. 28, nr. 12, p. 797–1808, 1998

[31] Concrete Compressive Strength Data Set, https://archive.ics.uci.edu/ml/datasets/Concrete+Compressive+Strength [visited on 14.03.2018]

[32] D. B. Birrcher, DESIGN OF REINFORCED CONCRETE DEEP BEAMS FOR STRENGTH AND SERVICEABILITY, The University of Texas at Austin, 2009

[33] Climatic Data OnLine (CDO), https://www7.ncdc.noaa.gov/CDO/country

[34] D. Britz, WildML, Recurrent Neural Networks Tutorial, Part 3 – Backpropagation Through Time and Vanishing Gradients, http://www.wildml.com/2015/10/recurrent-neural-networks-tutorial-part-3-backpropagation-through-time-and-vanishing-gradients/, [visited on 17.06.2018]