The Fourteenth International Conference on
Computational Structures Technology
23–25 August, 2022 | Montpellier, France

Proceedings of the Fourteenth International Conference on
Computational Structures Technology
Edited by B.H.V. Topping and J. Kruis
Civil-Comp Conferences, Volume 3, Paper 4.3
Civil-Comp Press, Edinburgh, United Kingdom, 2022, doi: 10.4203/ccc.3.4.3


# Using neural networks for dimensioning and certification of mechanical systems: model precision and accuracy

**Y. El Assami and B. Gely**

**Technology & Engineering Center, Capgemini Engineering**
**Blagnac, France**

## Abstract

Neural networks show impressive performance in lots of domains to handle problems of high complexity. They are universal approximators and can, in principle, be used to learn any type of model. Their use would be of a great benefit as they can be intended to automate major tasks within an engineering project (such as system dimensioning, certification, and criteria verification). However, it is not yet customary to use these technics for lack of competitiveness against forward engineering calculations. One major issue is the robustness and the difficulty to ensure high precisions for deterministic predictions. In this work, we investigate the ability of neural networks to be used to approximate engineering models and their performance in terms of precision and accuracy per target relative error. Increasing accuracy requires understanding how these models work in a deeper way. Applications on use-cases of mechanical structures are used to understand the behaviour of neural networks for this type of problems and illustrate the encountered constraints.

**Keywords:** model approximation, machine learning, neural networks, deep learning, model precision, mechanical structures.

## 1 Introduction

Even though neural networks are used for very impressive applications and the numerical framework to manage machine learning models is at its best, many domains based on deterministic calculations are in lack of use-cases [1], [2]. These new technics may seem not to be practical since each engineering system is unique, and it

is not obvious to distinguish common patterns from different problems. Also, resources needed to implement a machine learning model are generally more costly than what it takes to implement a single engineering model itself, as datasets are to be prepared for model training.

In fact, using machine learning models to simulate engineering systems may be interesting for problems that share similar inputs. This interest is more obvious for engineering problems in need of high numerical resources with consequent human intervention for modelling or post processing. We can think of direct use of these methods on examples of system dimensioning or certification and criteria verification.

Neural networks, for instance, have multiple advantages in terms of performance and flexibility [3], [4]. It has been demonstrated that neural networks are universal approximators [5]–[7]. They seem to be adequate to mimic engineering models and predict results without knowing the real details of a model [8]. Each perceptron calculates a linear combination of its inputs. The combination of many linear relations, with the use of activation functions, allows the calculation of a suitable approximation of a highly nonlinear problem.

Despite the advantages of these networks, there remain certain constraints linked to the non-interpretable nature of internal characteristics ("black box" [9]). Likewise, the number of parameters and hyper-parameters needed to define and compile such models is large, with no straightforward way to make optimal choices.

To make deep learning based expert engineering systems, model precision is the main issue that should be dealt with. To evaluate the feasibility of high precision learning, mechanical structures of different behaviours are defined as use-cases. These examples are chosen to be simple enough to make an intuition on the behaviour of neural networks in terms of prediction precision and complex enough to present challenging nonlinear relations between characteristics. These use-cases can be generalized to wider range of problems in different fields, as long as coherent datasets are established.

## 2 Methods

We would like to predict results of mechanical models using neural networks. These models are only use-cases and methods inspected hereby can be generalized to any alike situation. We target the quantification of resistant effort of several mechanical systems (system of bars under tension, fittings in aeronautics, …). Each system is defined with the geometry of its components, a material and, incidentally, a loading case. These mechanical models are used to generate synthetic datasets. But the trained neural networks suppose that the original models are not known or difficult to establish, which would always be the case.

Data we dispose of is numerically perfect (without noise). The aim is to train neural networks to reproduce this data with high precision. It is less likely to have problems of overfitting under these conditions and we have a large flexibility for fine tuning.

Safety is a major issue while dimensioning (expressed for example in terms of safety criteria or reserve factors). To meet safety conditions, relative error on each prediction needs to be controlled. Thus, it is more convenient to use relative errors to evaluate the quality of a neural network. More exactly, the accuracy $s_e$ of relative error e will be used as main metric.

$$s_e = \frac{\text{number of test data with relative error} < e}{\text{total number of test data}} \quad (1)$$

In the process of model approximation, different architectures are used. They are based on dense networks (DNN), with rectified linear activation function for several layers. The size of a model depends on the variance of the mechanical problem and the number of parameters. The choice of an architecture is based on experience and validated by several tests of performance. Gradient descent is realized with moment algorithm Adam [10] with different sets of parameters used for fine tuning.

In the framework of this study, the main purpose is to achieve high accuracies, even if calculations last longer. For datasets with very reduced noise, large number of iterations is allowed. Sometimes, training is executed more than once, with different initial learning rates, to seek higher numerical stability and bring the cost function as close as possible to a local minimum. In what follows, we present and discuss results of mechanical model approximation with neural networks in terms of accuracy per target relative error.

## 3 Results

To make an intuition about the performance of a neural network, let's apply it on a linear problem. For example, $N_{\mathrm{Rd}} = b_1 h f_{\mathrm{y}} + b_2 h f_{\mathrm{y}} + b_3 h f_{\mathrm{y}}$, where $N_{\mathrm{Rd}}$ is the normal resistance of a system of parallel bars, $h b_i$ the area of bar $i$ ($b_i$ variable) and $f_{\mathrm{y}}$ the limit of elasticity (constant). This model can obviously be represented with a single perceptron (linear regression). This problem is idealized, and the cost function has one global minimum. Theoretically, the weights of the perceptron should be equal to the constant parameters $h f_{\mathrm{y}}$. But as the optimization is done with gradient descent, there will always be a numerical error related to the chosen hyper-parameters.

When a high precision is required, the number of iterations is a parameter of high importance. And even with sophisticated optimizers, it helps to run training more than once with refined initial learning rates. The present model shows perfect scores for target relative errors higher than 10-3 ($s_{0.1\%} = 100\%$). This score decreases for smaller target relative errors (e.g. $s_{0.01\%} = 92\%$). The precision achieved here can also be deduced from the weights and the bias, calculated with a relative error of the order 10-4 (Table 1). Values of the tolerance, used to state the convergence (and thus affects the number of epochs), make a difference on the results when coarse. But there is a limit under which its importance is forsaken (Table 1).

In a more general case, if one was not aware of this linear relation, a neural network with a bigger capacity could have been used. This increase of capacity can sometimes result in lower performance [11], [12] as more local minima may appear.

Nevertheless, this simple calculation allows quantifying the best case expected precision of a model under a given configuration.

| *Tolerance* | *Pearson score* | *Accuracy for relative error of 0.1%* | *Accuracy for relative error of 0.01%* | *Mean relative error on model weights* | *Relative error on bias* | *Number of epochs* |
|---|---|---|---|---|---|---|
| | $R^2$ | $s_{0.1\%}$ | $s_{0.01\%}$ | $\frac{\langle w_i \rangle_{i \neq 0} - hf_y}{hf_y}$ | $\frac{w_0 - hf_y}{hf_y}$ | |
| $10^{-3}$ | 0.69274327 | 0.5% | 0.00% | 5.6e-1 | 4.1e-1 | 2794 |
| $10^{-5}$ | 0.99994436 | 47.8% | 5.91% | 7.4e-3 | 5.6e-3 | 8022 |
| $10^{-7}$ | 0.99999978 | 100% | 74.9% | 4.7e-4 | 3.5e-4 | 8418 |
| $10^{-9}$ | 0.99999992 | 100% | 93.1% | 2.8e-4 | 2.1e-4 | 8797 |
| $10^{-11}$ | 0.99999993 | 100% | 91.6% | 2.7e-4 | 2.1e-4 | 8798 |

Table 1: Evolution of relative error with respect to tolerance and number of epochs.

The approximation of a model with nonlinear relations surely needs the use of neural networks with bigger capacities. Several architectures are tested, and fine tuning applied to achieve the highest precisions. A perfect score can be obtained for a relative error up to 5%. But at $e = 1\%$, $s_{1\%} = 96\%$ at most (Figure 1).

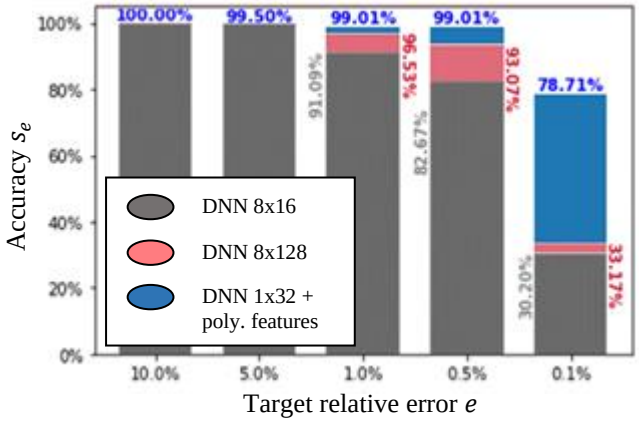


Figure 1: Evolution of accuracy w.r.t target relative error for dense neural networks of 8x16 units, 8x128 units and 1x32 with polynomial features.

Improving the performance of a model becomes more and more costly. One of the solutions to overcome this limit is second order features (square and cross products). Such approaches are used for simpler models but become also justified for neural networks to achieve better performance ($s_{0.1\%} > 99\%$) with architectures of much lower capacity (Figure 1).

## 4 Conclusions and Contributions

Neural networks are underused as universal approximators for engineering models due to the lack of competitivity against numerical models and the assumption that each system is unique. However, from another angle, they can make many studies easier, especially for multi-dimensional problems that share similar characteristics. If

the complexity of engineering systems varies a lot for each problem, the process of training a neural network remains the same.

Using neural networks to predict deterministic results might appear to be odd, as they are mostly evaluated from a statistical point of view. But the nature of the applications justifies this use. One difficulty that can be encountered though and that this paper tries to evaluate is the precision of predictions. We notice that even with parameter optimization, capacity variation of networks and fine tuning, there is always a numerical limit to the precision that becomes difficult to overcome.

As shown here, quantifying the relative error for each test case may lead to conclusions different from when the reasoning is over usual metrics such as distance indicators (MSE) or correlation indicators (R2). And as it can be expected, relative errors are higher for results with lower amplitudes. If the studied system requires high precision that cannot be provided by a neural network, the robustness of this network used for approximation should be quantified and taken into account (in the safety coefficients for example).

Polynomial features are among the ways that were proven to easily increase efficiency of machine learning models. Such technic is usually not convenient for learning models with large numbers of inputs (computer vision or time series), but it can be justified for problems with few parameters like the use-cases described in this work.

Despite all the technics used to improve the models, most of the time, it is not yet possible to achieve perfect accuracy for small target relative errors, even for problems with known global minima. Reaching perfect accuracy is a feedforward goal that would encourage considering neural networks as a way to automate solving complex physical models.

# References


[1] J. Willard, X. Jia, S. Xu, M. Steinbach, and V. Kumar, 'Integrating Scientific Knowledge with Machine Learning for Engineering and Environmental Systems', *ArXiv200304919 Phys. Stat*, Jul. 2021

[2] S. Lathuilière, P. Mesejo, X. Alameda-Pineda, and R. Horaud, 'A Comprehensive Analysis of Deep Regression', *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 9, pp. 2065–2081, Sep. 2020, doi: 10.1109/TPAMI.2019.2910523.

[3] C. Sharpe, T. Wiest, P. Wang, and C. C. Seepersad, 'A Comparative Evaluation of Supervised Machine Learning Classification Techniques for Engineering Design Applications', *J. Mech. Des.*, vol. 141, no. 12, Oct. 2019, doi: 10.1115/1.4044524.

[4] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[5] K. Hornik, M. Stinchcombe, and H. White, 'Multilayer feedforward networks are universal approximators', *Neural Netw.*, vol. 2, no. 5, pp. 359–366, Jan. 1989, doi: 10.1016/0893-6080(89)90020-8.

[6] D.-X. Zhou, 'Universality of Deep Convolutional Neural Networks', *ArXiv180510769 Cs Stat*, Jul. 2018

[7] A. Kratsios and E. Bilokopytov, ‘Non-Euclidean Universal Approximation’, *ArXiv200602341 Cs Math Stat*, Nov. 2020
[8] J.-A. Goulet, *Probabilistic Machine Learning for Civil Engineers*. Cambridge, MA, USA: MIT Press, 2020.
[9] X. Cheng, B. Khomtchouk, N. Matloff, and P. Mohanty, ‘Polynomial Regression As an Alternative to Neural Nets’, *ArXiv180606850 Cs Stat*, Apr. 2019
[10] D. P. Kingma and J. Ba, ‘Adam: A Method for Stochastic Optimization’, *ArXiv14126980 Cs*, Jan. 2017
[11] Y. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, and Y. Bengio, ‘Identifying and attacking the saddle point problem in high-dimensional non-convex optimization’, *ArXiv14062572 Cs Math Stat*, Jun. 2014
[12] H. Kwak and B.-T. Zhang, ‘Understanding Local Minima in Neural Networks by Loss Surface Decomposition’, Feb. 2018