CupNet – Pruning a network for geometric data

Raoul Heese[1,2,*], Lukas Morand[3], Dirk Helm[3], Michael Bortz[1,2]

[1] Fraunhofer Center for Machine Learning
[2] Fraunhofer Institute for Industrial Mathematics ITWM
[3] Fraunhofer Institute for Mechanics of Materials IWM
[*] raoul.heese@itwm.fraunhofer.de

Abstract

Using data from a simulated cup drawing process, we demonstrate how the inherent geometrical structure of cup meshes can be used to effectively prune an artificial neural network in a straightforward way.

Keywords: regression, informed learning, pruning, network architecture, deep drawing

1 Introduction

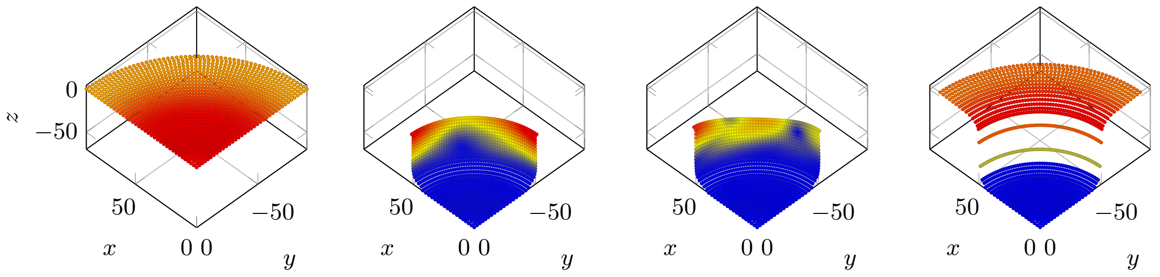
The optimization of production processes can benefit from machine learning methods that incorporate domain knowledge and data from numerical simulations [1]. Typically, such methods aim to model relations between process parameters and the resulting product. In this manuscript, we consider an example from the field of deep drawing, a sheet metal forming process in which a sheet metal blank is drawn into a forming die by mechanical action.

Specifically, we study the prediction of product geometries in a cup drawing process based on data from finite element simulations [2]. For each simulation, we choose randomized process and material parameters $\mathbf{p} \in \mathbb{R}^k$ with $k \equiv 9$ and observe the resulting geometry as a set of $m \equiv 1\,979$ mesh coordinates $\mathbf{x} \in \mathbb{R}^d$ with $d \equiv 3m = 5\,937$. Thus, the machine learning task is to predict

$$\hat{\mathbf{x}}(\mathbf{p}) : \mathbb{R}^k \longmapsto \mathbb{R}^d \tag{1}$$

based on the generated data. Such a predictive regression model can be considered as a short-cut for the actual simulation. In contrast to the simulation, it is faster and always numerically stable and therefore particularly suitable to solve optimization problems. On the other hand, the model predictions are less accurate than the simulation results, which corresponds to a trade-off between calculation speed and outcome precision.

The choice of parameters affects the resulting cup quality in the sense that we can infer good, defect and cracked cups (indicated by strong deformations) from the mesh geometries. In total, we ran 10 000 simulations, of which two failed (for numerical reasons). Of the remaining 9 998 parameter combinations, 3 991 lead to good cups, 5 075 lead to defect cups and 932 cause cracked cups, cf. Fig. 1.



a) Before deformation. b) Good cup. c) Defect cup. d) Cracked cup.

Figure 1: Typical cup geometries, each consisting of m points. For reasons of symmetry, it is sufficient to simulate the deformation of a quarter cup segment instead of the full cup. The colors indicate the distance of each point to the reference mesh, consisting of the average coordinates of all good cups. We use a different color scale for each subfigure: 0 ▬ a) 70, b) 2, c) 6, d) 49.

2 Method

We propose two artificial neural networks to model Eq. (1). Our first network architecture, which we call `CupNet`, particularly takes the geometrical structure of the data into account to effectively prune the network weights. Pruning is a technique that helps in the development of smaller and more efficient networks, see, e. g., Ref. [3] and references therein. That means, instead of changing the loss function as in e. g., Ref. [4], we use expert knowledge to change the network architecture itself.

The proposed network consists of an input layer of size k (i. e., it contains k units), which is fully connected to an initial layer of size d, which we call *frame*. We split the frame layer into three evenly sized segments (i. e., one for each dimension denoted by x, y, and z, cf. Fig. 1), which are each connected to the following layers in a special way. Specifically, we chose the forward pass

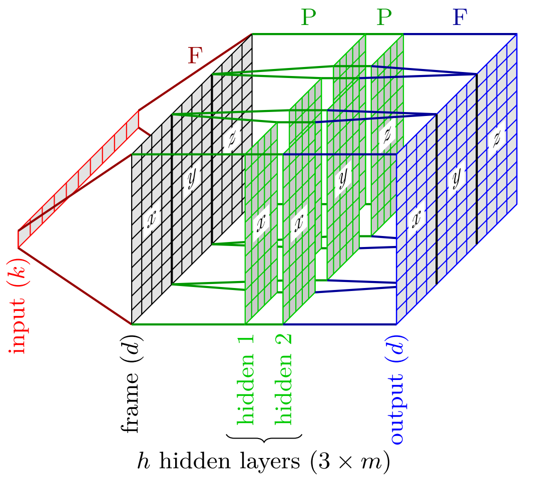
$$\mathbf{o} \equiv A([\mathbf{C}(\alpha) \odot \mathbf{W}]^\intercal \mathbf{i} + \mathbf{b}). \tag{2}$$

Here $\mathbf{i} \in \mathbb{R}^{m\times 1}$ and $\mathbf{o} \in \mathbb{R}^{m\times 1}$ represent the layer inputs and outputs, whereas $\mathbf{W} \in \mathbb{R}^{m\times m}$ and $\mathbf{b} \in \mathbb{R}^{m\times 1}$ stand for the (trainable) layer weights and biases, and $A(\cdot)$ represents the activation function. The symbol $\odot$ denotes the Hadamard product (element-wise multiplication). Moreover, we have introduced the symmetric pruning matrix $\mathbf{C}(\alpha) \in \{0,1\}^{m\times m}$ with elements

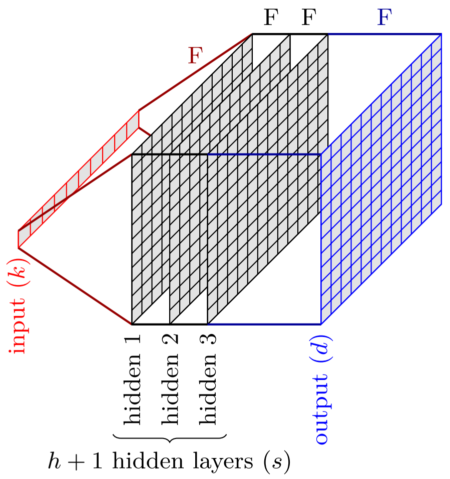
$$\mathbf{C}_{ij}(\alpha) \equiv \begin{cases} 0 & \text{if } \mathbf{D}_{ij} > \alpha \\ 1 & \text{if } \mathbf{D}_{ij} \leq \alpha \end{cases} \qquad \text{with} \qquad \mathbf{D}_{ij} \equiv ||\mathbf{x}'_\mathbf{i} - \mathbf{x}'_\mathbf{j}||_2 = \text{const.} \tag{3}$$

for $i, j = 1, \dots, m$. It is based on the symmetric distance matrix $\mathbf{D} \in \mathbb{R}^{m\times m}_{\geq 0}$, which contains the euclidean distances $||\cdot||_2$ between different mesh points $\mathbf{x}'$ of the undeformed geometry, Fig. 1a). Thus, the pruning matrix removes the influence of all weights for which the corresponding mesh points of the undeformed geometry have a distance beyond the user-defined pruning threshold $\alpha \geq 0$.

This special layer configuration is repeated h times and concludes with a fully-connected last layer merging the three previously splitted segments into the output layer. Summarized, we use the inherent geometrical structure of the data to prune a fully connected network in such a way that correlations between spatially related mesh points are preserved. The complete architecture is sketched in Fig. 2a).



a) Our proposed `CupNet` with $n_{\text{cup}}(h, \alpha)$ trainable parameters, Eq. (4), and dimensions x, y, and z.



b) The reference network `RefNet` with $n_{\text{ref}}(h, s)$ trainable parameters, Eq. (5).

Figure 2: Network architectures used to model Eq. (1). The k-dimensional simulation parameters $\mathbf{p}$ constitute the input, whereas the output corresponds to the d-dimensional mesh $\hat{\mathbf{x}}(\mathbf{p})$. Both architectures consist of a sequence of layers (number of units in brackets), which are fully connected (F) or partially connected, i. e., pruned (P). We apply a dropout after every inner layer for regularization.

As a reference we also use a second architecture, which we call `RefNet`. It has a similar structure and complexity as the `CupNet`, but does not take advantage of the geometrical structure of the data. Specifically, it consists of an input layer of size k, $h+1$ hidden layers of size s and an output layer of size m, all of which are fully connected. In order to obtain a comparable complexity of the two architectures we choose s in such a way that the number of trainable parameters is as close as possible. For that, we first determine the number of trainable parameters for the `CupNet` architecture

$$n_{\text{cup}}(h, \alpha) = (k \cdot d + d) + 3h(c(\alpha) + m) + 3(m^2 + m) \qquad \text{with} \qquad c(\alpha) \equiv \sum_{i,j=1}^{m} \mathbf{C}_{ij}(\alpha). \tag{4}$$

On the other hand, the number of trainable parameters for the `RefNet` architecture reads

$$n_{\text{ref}}(h, s) = (k \cdot s + s) + h(s^2 + s) + (s \cdot d + d) \tag{5}$$

so that the condition $n_{\text{ref}}(h, s) \overset{!}{=} n_{\text{cup}}(h, \alpha)$ leads to $s(h, \alpha) = (-u + \sqrt{u^2 - 4h(d - n_{\text{cup}}(h, \alpha))})/(2h)$, where $u \equiv k + h + d + 1$. We set the layer size s to the ceiling of this value for a given depth h and a given pruning threshold α. The complete `RefNet` architecture is sketched in Fig. 2b).

For both networks the activation function is chosen to be a ReLU for all but the output layer for which we use a linear activation. As a regularizer we apply a dropout after every inner layer, where we randomly set 20% of the units to zero.

3 Experiments

We test the performance of both network architectures, which we have implemented using Ref. [5]. For this purpose we split the standardized data into a training set with 9 000 elements and a test set with 998 elements stratified according to the three cup classes of good cups, defect cups, and cracked cups. We repeat this approach for 10 benchmark runs with different data splittings and different random seeds for the network initialization. For the training we use an Adam optimizer and a mean squared error for the loss. As a result, we obtain the R^2 scores shown in Table 1 and Fig. 3.

Table 1: R^2 score means and standard deviations over all 10 benchmark runs for different values of the network depth h and the pruning threshold α. The best mean results are highlighted in colored bold. A plot of the results is shown in Fig. 3.

network	$\alpha = 1.0$	$\alpha = 2.5$	$\alpha = 5.0$	$\alpha = 10.0$	$\alpha = 25.0$	$\alpha = 50.0$
`CupNet`, $h = 1$	**0.822 ± 0.042**	**0.823 ± 0.052**	**0.829 ± 0.040**	**0.832 ± 0.084**	**0.839 ± 0.044**	**0.819 ± 0.072**
`RefNet`, $h = 1$	0.818 ± 0.047	0.818 ± 0.066	0.815 ± 0.066	0.817 ± 0.045	0.818 ± 0.056	0.817 ± 0.053
`CupNet`, $h = 2$	**0.821 ± 0.038**	**0.850 ± 0.039**	**0.862 ± 0.046**	**0.851 ± 0.057**	**0.847 ± 0.047**	**0.831 ± 0.045**
`RefNet`, $h = 2$	0.804 ± 0.050	0.813 ± 0.061	0.809 ± 0.048	0.821 ± 0.036	0.803 ± 0.037	0.800 ± 0.070
`CupNet`, $h = 3$	**0.830 ± 0.039**	**0.849 ± 0.039**	**0.850 ± 0.053**	**0.861 ± 0.056**	**0.839 ± 0.053**	**0.829 ± 0.043**
`RefNet`, $h = 3$	0.811 ± 0.043	0.789 ± 0.085	0.826 ± 0.042	0.802 ± 0.064	0.810 ± 0.045	0.827 ± 0.029
`CupNet`, $h = 4$	**0.830 ± 0.040**	**0.838 ± 0.050**	**0.845 ± 0.064**	**0.849 ± 0.035**	**0.817 ± 0.114**	**0.806 ± 0.034**
`RefNet`, $h = 4$	0.797 ± 0.055	0.792 ± 0.044	0.800 ± 0.052	0.808 ± 0.041	0.779 ± 0.051	0.768 ± 0.068
`CupNet`, $h = 5$	0.478 ± 0.292	**0.835 ± 0.064**	**0.838 ± 0.048**	**0.839 ± 0.037**	**0.832 ± 0.036**	**0.805 ± 0.033**
`RefNet`, $h = 5$	**0.762 ± 0.060**	0.774 ± 0.041	0.789 ± 0.038	0.767 ± 0.050	0.778 ± 0.042	0.691 ± 0.126
`CupNet`, $h = 6$	0.052 ± 0.108	**0.832 ± 0.042**	**0.840 ± 0.046**	**0.832 ± 0.103**	**0.832 ± 0.045**	**0.787 ± 0.080**
`RefNet`, $h = 6$	**0.780 ± 0.041**	0.785 ± 0.039	0.769 ± 0.049	0.770 ± 0.045	0.765 ± 0.019	0.436 ± 0.129
`CupNet`, $h = 7$	0.053 ± 0.161	0.714 ± 0.191	**0.843 ± 0.049**	**0.857 ± 0.033**	**0.825 ± 0.051**	**0.781 ± 0.049**
`RefNet`, $h = 7$	**0.772 ± 0.034**	**0.761 ± 0.058**	0.755 ± 0.040	0.755 ± 0.031	0.592 ± 0.243	0.377 ± 0.203

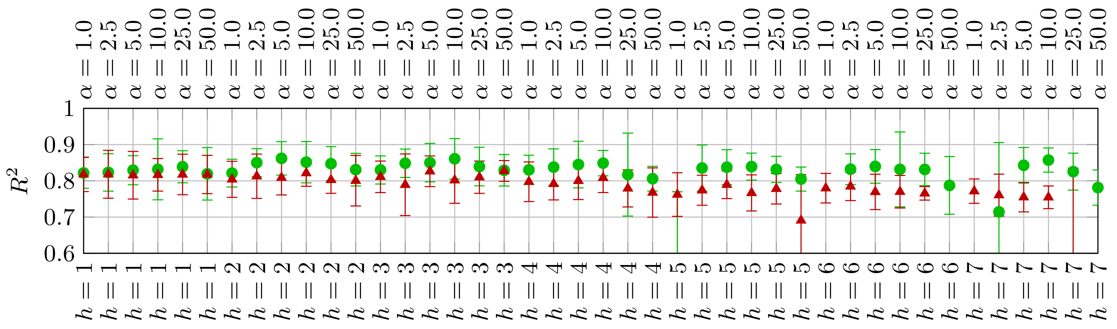


Figure 3: Visualization of the benchmark results from Table 1 for ● `CupNet` and ▲ `RefNet`, respectively.

We find that in most cases, `CupNet` is superior to `RefNet`. It performs worse only for low α and large h (i. e., strong pruning and deep nets). In these cases, the training process appears to not converge to a sufficiently good state. However, the mean scores of the two architectures are mostly within one standard deviation of each other. The best mean `CupNet` score of 0.862 is achieved for $h = 2$ and $\alpha = 5$, which corresponds to a network with $n_{\text{cup}} = 12\,277\,950$ trainable parameters. On the other hand, the best mean `RefNet` score of 0.827 is achieved for $h = 3$ and $\alpha = 50$, which corresponds to a much larger network with $n_{\mathrm{ref}} = 34\,898\,737$ trainable parameters. Thus, according to the respective optimal values for h and α, our pruning approach leads to smaller networks with a better expected score.

As an alternative approach we also test the performance of a Random Forest regressor (with 250 trees) on PCA-transformed features (with 100 components) using the implementation from Ref. [6]. This leads to a R^2 score of 0.736 ± 0.029 over 10 benchmark runs, which is worse than most results from Table 1.

4 Conclusion

Summarized, we find that our approach of pruning the network connections according to the spatial correlation of mesh points leads to a better expected performance in comparison with a reference network with similar structure and complexity. It would be an interesting point of origin for further studies to check whether this approach can also be applied to other geometrical data.

5 Acknowledgements

We would like to thank Alexander Butz for helpful discussions, Maria Baiker and Jan Pagenkopf for providing the simulation of the cup drawing process, and Boris Giba for data processing. This work was developed in the Fraunhofer Cluster of Excellence “Cognitive Internet Technologies” as well as the Fraunhofer lighthouse project “Machine Learning for Production”.

References

[1] Laura von Rueden et al. Informed machine learning – a taxonomy and survey of integrating knowledge into learning systems. *arXiv:1903.12394v2* 2020.

[2] Rodrigo Iza-Teran, Lukas Morand, Dirk Helm, and Jochen Garcke. Learning Product Properties with Small Data Sets in Forming Simulations. Submitted to: *NUMISHEET 2020: 12th International Conference and Workshop on Numerical Simulation of 3D Sheet Metal Forming Processes*, 2020.

[3] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning, 2018.

[4] Raoul Heese, Michał Walczak, Lukas Morand, Dirk Helm, and Michael Bortz. The good, the bad and the ugly: Augmenting a black-box model with expert knowledge. In Igor V. Tetko, Věra Kůrková, Pavel Karpov, and Fabian Theis, editors, *Artificial Neural Networks and Machine Learning – ICANN 2019: Workshop and Special Sessions*, pages 391–395, Cham, 2019. Springer International Publishing.

[5] Martín Abadi et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015. Software available from tensorflow.org.

[6] F. Pedregosa et al. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* **12**, 2825–2830, 2011.