Knowledge Injection via ML-based Initialization of Neural Networks

Lars Hoffmann[1], Christian Bartelt[1] and Heiner Stuckenschmidt[2]

[1]University of Mannheim, Institute for Enterprise Systems, L15, 1-6, 68131 Mannheim, Germany
[2]University of Mannheim, Chair of Artificial Intelligence, B6, 26, 68131 Mannheim, Germany

Abstract

Despite the success of artificial neural networks (ANNs) for various complex tasks, their performance and training duration heavily rely on several factors. In many application domains these requirements, such as high data volume and quality, are not satisfied. To tackle this issue, different ways to inject existing domain knowledge into the ANN generation provided promising results. However, the initialization of ANNs is mostly overlooked in this paradigm and remains an important scientific challenge. In this paper, we present a machine learning framework enabling an ANN to perform a semantic mapping from a well-defined, symbolic representation of domain knowledge to weights and biases of an ANN in a specified architecture.

Keywords

Knowledge Injection, Neural Networks, Initialization, Machine Learning

1. Introduction

Despite the substantial achievements of artificial neural networks (ANNs) driven by high generalization capabilities, flexibility, and robustness, the training duration and performance still highly depend on several factors, such as the network architecture, the loss function, the initialization method, and most importantly on the available training data. However, in many real-world applications, e.g., in safety-critical systems, there are various issues regarding data collection and generation. In these scenarios, the capabilities of exclusively data-oriented approaches to train an ANN are limited.

To tackle these domain-specific challenges, the concept of integrating or injecting existing domain knowledge into the generation process of ANNs becomes increasingly attractive in research and practice, indicated by several survey papers for machine learning (e.g., [1, 2, 3, 4]) as well as deep learning in particular (e.g., [5, 6, 7]). Furthermore, this paradigm bares the potential to mitigate general weaknesses of ANNs, like slow convergence speed, high data demands and the risk of getting stuck at local minima or saddle points [8].

Consequently, there is a great variety of approaches in this field focusing on different elements within the ANN generation process. One prominent category targets the learning process by adding domain-specific constraints or loss terms to the cost function, such as [9], [10], [11] and [12]. However, there is little to no research on initializing the weights and biases of an ANN based on domain knowledge. Such knowledge can act as a pointer towards a promising starting point in the optimization landscape. The resulting "warm start" of the learning process can reduce the required training time as well as improve the overall performance. This effect shall be exploited efficiently by the framework presented in this paper.

With this goal in mind, the existing collection of network initialization techniques were analyzed. Aguirre and Fuentes [13] define three groups. "Data-independent" methods are based on randomly drawing samples of different distributions, e.g., *LeCun* [14], *Xavier* [15] and *He* [16]. "Data-dependent" approaches, such as *WIPE* [17], *LSUV* [18] and *MIWI* [19], additionally take statistical properties of the available training data into account. Approaches within the third group, like [20, 21, 22, 23, 24], apply the concept of "pre-training". Their goal is to learn an ANN on a related problem (with sufficient availability of high-quality data) and use it as an initialization for the primary task. Consequently, they are not limited to the actual training data, and thus to some extent independent to task-specific data issues. Although, none of these approaches explicitly considers domain knowledge, they could be adapted to pre-train an ANN on synthetic data encoding domain knowledge; also proposed by Karpatne et al. [4]. These data can be generated, for instance, by simulations or querying a domain model. But this comes

KINN@CIKM'21: Proceedings of CIKM Workshop on Knowledge Injection in Neural Networks, November 1, 2021, Online Virtual Event
hoffmann@es.uni-mannheim.de (L. Hoffmann);
bartelt@es.uni-mannheim.de (C. Bartelt);
heiner@informatik.uni-mannheim.de (H. Stuckenschmidt)
https://www.uni-mannheim.de/ines/ueber-uns/wissenschaftliche-mitarbeiter/lars-hoffmann (L. Hoffmann);
https://www.uni-mannheim.de/ines/ueber-uns/wissenschaftliche-mitarbeiter/dr-christian-bartelt (C. Bartelt);
https://www.uni-mannheim.de/dws/people/professors/prof-dr-heiner-stuckenschmidt/ (H. Stuckenschmidt)
0000-0002-9667-0310 (L. Hoffmann); 0000-0003-0426-6714 (C. Bartelt); 0000-0002-0209-3859 (H. Stuckenschmidt)

with several disadvantages. First, the data samples must be efficiently generated to fully represent the domain knowledge, and second, the ANN must be able to learn the contained knowledge. On top of this potential loss of information, this entire process must be repeated every time the domain knowledge or the target network architecture changes.

Replacing this indirect, data-based knowledge transfer from an already existing domain model into an ANN with a direct mapping or transformation can potentially solve these problems. Although they do not relate to knowledge injection, several authors engineered explicit mapping algorithms for different representations. A prominent example are Decision Trees (DTs), because they also have a graph-based structure. Early work was already performed in the 1990s (e.g., [25, 26, 27, 28, 29]), but this topic is recently becoming more attention again (e.g., [30, 31, 32, 33]). Nevertheless, there are two major shortcomings if applied to knowledge injection with initialization. On the one hand, they are model-specific and hard to engineer, which makes them impractical considering the diversity of knowledge representations. On the other hand, they cannot map to arbitrary ANN architectures, which may restrict an ANN's ability to discover new characteristics in the subsequent optimization. This becomes increasingly critical as the gap between the expressed knowledge and the entire task complexity widens.

Instead of engineering such mappings by hand, this paper introduces a machine learning (ML) framework capable of training an ANN to become a semantic mapping from a well-defined, algebraic representation of domain knowledge to a network's weights and biases. We call such a mapping "Transformation Network" or $\mathcal{T}$-Net. This data-driven framework can be applied to various model algebras, such as DTs or polynomials, with only slight adaptions. Thereby, it tackles the challenge of variability in domain knowledge representations by transferring the complex mapping generation from humans to machines. Furthermore, an arbitrary network structure as $\mathcal{T}$-Net output can be selected, which achieves an independence between the complexity of the domain model and the target ANN.

2. Framework and Approach

In this section, we give a brief introduction on (1) how the proposed framework trains an ANN to become a semantic mapping ($\mathcal{T}$-Net) from the internals of a given algebraic model to a network's weights and biases, and (2) how to utilize its capabilities for knowledge injection via ANN initialization.

2.1. $\mathcal{T}$-Net Generation

Before a given task-specific function approximation, also referred to as domain model, can be injected, a suitable $\mathcal{T}$-Net must be generated once with the proposed ML framework. This can be done completely with synthetic data. The overall objective is to maximize the transformation fidelity between the input function and the predicted ANN. A schematic overview of the framework consisting of three main steps is shown in Figure 1.

2.1.1. Algebra Selection and λ-Function Generation

At first, a diverse set of functions Λ in the same algebra as the given task-specific domain model is created. If such a domain model is not already defined, a well-suited algebra for representing the existing domain knowledge needs to be selected and the model generated. This can be done implicitly by pre-training or explicitly by an expert. However, each function $\lambda_i \in \Lambda$ must operate on the same solution space defined by the overall task to be solved, for instance, a binary classification or regression problem. In addition, each function λ_i requires a set of N representative examples $\mathcal{D}_{\lambda_i}$ as

$$\left\{\mathcal{D}_{\lambda_i} = \{(x_{i,j}, y_{i,j})\}_{j=1}^{N}\right\}_{i=1}^{|\Lambda|},$$

where $x_{i,j} = (x_{i,j_1}, x_{i,j_2}, \ldots, x_{i,j_D})$ denotes one data point of dimensionality D and $y_{i,j} = \lambda_i(x_{i,j})$ is the result after applying the function λ_i to $x_{i,j}$. How to generate Λ and set N depends on the given context.

2.1.2. Data Preparation for $\mathcal{T}$-Net Training

Before the $\mathcal{T}$-Net training, the data and λ-functions must be put in the correct shape, i.e., numeric vectors for ANNs. Therefore, an encoding method, denoted as Enc_λ, is required. Similarly, a decoding method Dec_μ enables the translation of the returned network weights and biases to an executable ANN μ_i. The dataset required for the $\mathcal{T}$-Net training is defined as

$$\mathcal{D}_\mathcal{T} := \{(Enc_\lambda(\lambda_i), \mathcal{D}_{\lambda_i})\}_{i=1}^{|\Lambda|},$$

where each example is a tuple of the encoded function λ_i and its representative samples $\mathcal{D}_{\lambda_i}$. For clarification, these samples are not the target output of the $\mathcal{T}$-Net, but are required for the loss calculation. This is described in the next step.

2.1.3. $\mathcal{T}$-Net Training

After the preparations, the $\mathcal{T}$-Net is trained. Its weights and biases are adjusted based on the backpropagated prediction error over the training dataset $\mathcal{D}_\mathcal{T}$. This error

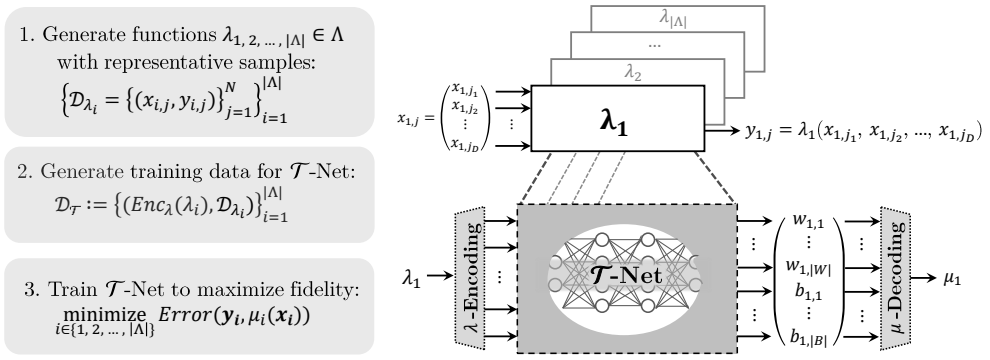

Figure 1: Overview of proposed machine learning framework for training a transformation ANN ($\mathcal{T}$-Net) in three main steps: function generation, data preparation and $\mathcal{T}$-Net training.

measures how different each input function λ_i is compared to the currently predicted ANN counterpart μ_i. To quantify this difference, a traditional task-specific measure ($Error$), e.g., categorical cross-entropy, is applied to the true values $\boldsymbol{y_i} = \lambda_i(\boldsymbol{x_i})$ and the predictions $\mu_i(\boldsymbol{x_i})$ given the vector of all input samples $\boldsymbol{x_i}$. The overall optimization goal can be formally described as

$$\underset{i \in \{1,2,\dots,|\Lambda|\}}{\text{minimize}} \; Error(\boldsymbol{y_i}, \mu_i(\boldsymbol{x_i})).$$

Thus, the $\mathcal{T}$-Net training aims to maximize the transformation fidelity. By that, we want to enable the $\mathcal{T}$-Net to generalize to previously unseen λ-functions, making it a capable mapping for this family of functions.

2.2. Knowledge Injection via $\mathcal{T}$-Net Execution

After the one-time effort of generating a suitable $\mathcal{T}$-Net, it is able to instantly initialize ANNs for all possible domain models within the trained function algebra. Therefore, we just need to pass the encoded representation to the $\mathcal{T}$-Net and let it predict the initial weights and biases. In the current state, one $\mathcal{T}$-Net maps to ANNs with a pre-defined specification, i.e., architecture and activation functions. To achieve a high fidelity, it must be assumed that ANNs with this specification are capable of accurately approximating the input functions. However, if changes to the network specification are required, only the $\mathcal{T}$-Net training must be repeated with an adapted output layer and/or μ-Decoding.

3. Evaluation

In this section, we want to briefly show that our framework shows promising results in practice in terms of the $\mathcal{T}$-Net mapping fidelity as well as the effects of applying it for ANN initialization. Therefore, we conducted experiments on polynomials as symbolic domain models to support solving random regression problems including two variables. The $\mathcal{T}$-Net was trained on 10,000 polynomials with orders between 0 and 8 to find the closest ANN approximations with one hidden layer of 225 neurons.

Without extensive hyperparameter optimization, the $\mathcal{T}$-Net could achieve on average a mapping fidelity quantified by the coefficient of determination (R^2) of 0.77 (± 0.26) over representative samples on a set of 2,500 test polynomials. Despite the noticeable distance to a perfect mapping ($R^2 = 1$), it significantly proves the learning capability of the proposed ML framework.

To investigate the impact of injecting knowledge by utilizing $\mathcal{T}$-Nets on a given ANN task, the training duration and prediction performance were analyzed. A total of 2,500 synthetic regression problems were randomly created and then two ANNs were trained on each problem; one lets the $\mathcal{T}$-Net predict the initial weights and biases based on a polynomial approximation, and the second one applies the *Xavier* uniform initializer [15] as a benchmark. Early stopping was used to indicate convergence during the optimization. Besides the different initialization, all other factors and parameters remained the same.

By applying the $\mathcal{T}$-Net for initialization, in 91% of the regression test cases the prediction performance increased and 96% required less epochs to converge, i.e., hitting the early stopping criterion, compared to the naive benchmark. More specifically, the training duration could be reduced on average by 64%. In addition, the resulting ANN performance in terms of the mean absolute error (MAE) showed an average increase of 2.7%. Figure 2 illustrates these two benefits in more detail.

This condensed evaluation demonstrates the benefits of the proposed framework and emphasizes the potential for knowledge injection into ANNs. A more sophisticated evaluation is currently a work in progress.

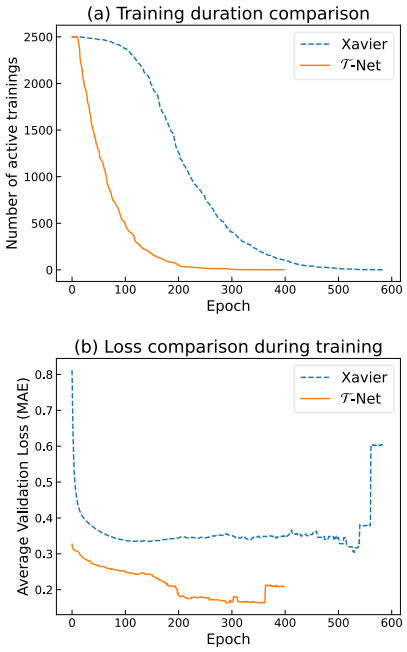

Figure 2: Comparison between $\mathcal{T}$-Net injection and *Xavier* uniform initialization in terms of (a) training duration and (b) performance.

4. Conclusion

In this paper, we have introduced a novel approach of knowledge injection into ANNs by utilizing existing domain models for initialization. Therefore, a semantic mapping from the domain model's internals to weights and biases is applied. Instead of engineering such an explicit mapping by hand, we designed a machine learning framework capable of training an ANN to perform this transformation. We call such a transformation network $\mathcal{T}$-Net. Besides the reduction of manual effort, it has the big advantage of decoupling the complexity of the domain model and ANN space.

Based on promising initial experiments, we hypothesize that this framework can generate $\mathcal{T}$-Nets with sufficient fidelity by appropriately addressing the following aspects: (1) its network specification (e.g., architecture and activation functions), (2) the learning behavior (e.g., loss function and optimizer), (3) the training data generation (e.g., diversity of domain models), and (4) the numeric encoding of the domain model algebra.

References

[1] R. Rai, C. K. Sahu, Driven by Data or Derived Through Physics? A Review of Hybrid Physics Guided Machine Learning Techniques With Cyber-Physical System (CPS) Focus, IEEE Access 8 (2020) 71050–71073. doi:10.1109/ACCESS.2020.2987324, conference Name: IEEE Access.

[2] L. von Rueden, S. Mayer, K. Beckh, B. Georgiev, S. Giesselbach, R. Heese, B. Kirsch, M. Walczak, J. Pfrommer, A. Pick, R. Ramamurthy, J. Garcke, C. Bauckhage, J. Schuecker, Informed Machine Learning - A Taxonomy and Survey of Integrating Prior Knowledge into Learning Systems, IEEE Transactions on Knowledge and Data Engineering (2021) 1–1. doi:10.1109/TKDE.2021.3079836, conference Name: IEEE Transactions on Knowledge and Data Engineering.

[3] C. Deng, X. Ji, C. Rainey, J. Zhang, W. Lu, Integrating Machine Learning with Human Knowledge, iScience 23 (2020) 101656. URL: https://www.sciencedirect.com/science/article/pii/S2589004220308488. doi:10.1016/j.isci.2020.101656.

[4] A. Karpatne, G. Atluri, J. H. Faghmous, M. Steinbach, A. Banerjee, A. Ganguly, S. Shekhar, N. Samatova, V. Kumar, Theory-Guided Data Science: A New Paradigm for Scientific Discovery from Data, IEEE Transactions on Knowledge and Data Engineering 29 (2017) 2318–2331. doi:10.1109/TKDE.2017.2720168, conference Name: IEEE Transactions on Knowledge and Data Engineering.

[5] H. D. Gupta, V. S. Sheng, A Roadmap to Domain Knowledge Integration in Machine Learning, in: 2020 IEEE International Conference on Knowledge Graph (ICKG), IEEE, Nanjing, China, China, 2020, pp. 145–151. doi:10.1109/ICBK50248.2020.00030.

[6] A. Borghesi, F. Baldo, M. Milano, Improving Deep Learning Models via Constraint-Based Domain Knowledge: a Brief Survey, arXiv:2005.10691 [cs, stat] (2020). URL: http://arxiv.org/abs/2005.10691, arXiv: 2005.10691.

[7] T. Dash, S. Chitlangia, A. Ahuja, A. Srinivasan, Incorporating Domain Knowledge into Deep Neural Networks, arXiv:2103.00180 [cs] (2021). URL: http://arxiv.org/abs/2103.00180, arXiv: 2103.00180.

[8] Ç. Gülçehre, Y. Bengio, Knowledge matters: importance of prior information for optimization, The Journal of Machine Learning Research 17 (2016) 226–257.

[9] J. Xu, Z. Zhang, T. Friedman, Y. Liang, G. Broeck, A Semantic Loss Function for Deep Learning with Symbolic Knowledge, in: Proceedings of the 35th International Conference on Machine Learning, PMLR, Stockholm, Sweden, 2018, pp. 5502–5511. URL: http://proceedings.mlr.press/v80/xu18h.html, iSSN: 2640-3498.

[10] Z. Hu, Z. Yang, R. Salakhutdinov, X. Liang, L. Qin,

H. Dong, E. P. Xing, Deep generative models with learnable knowledge constraints, in: Proceedings of the 32nd International Conference on Neural Information Processing Systems, NIPS'18, Curran Associates Inc., Red Hook, NY, USA, 2018, pp. 10522–10533.

[11] M. Diligenti, S. Roychowdhury, M. Gori, Integrating Prior Knowledge into Deep Learning, in: 16th IEEE International Conference on Machine Learning and Applications (ICMLA), IEEE, Cancun, Mexico, 2017, pp. 920–923. doi:10.1109/ICMLA.2017.00-37.

[12] N. Muralidhar, M. R. Islam, M. Marwah, A. Karpatne, N. Ramakrishnan, Incorporating Prior Domain Knowledge into Deep Neural Networks, in: 2018 IEEE International Conference on Big Data (Big Data), IEEE, Seattle, WA, USA, USA, 2018, pp. 36–45. doi:10.1109/BigData.2018.8621955.

[13] D. Aguirre, O. Fuentes, Improving Weight Initialization of ReLU and Output Layers, in: I. V. Tetko, V. Kůrková, P. Karpov, F. Theis (Eds.), Artificial Neural Networks and Machine Learning – ICANN 2019: Deep Learning, Lecture Notes in Computer Science, Springer International Publishing, Cham, 2019, pp. 170–184. doi:10.1007/978-3-030-30484-3_15.

[14] Y. LeCun, L. Bottou, G. B. Orr, K. R. Müller, Efficient BackProp, in: G. B. Orr, K.-R. Müller (Eds.), Neural Networks: Tricks of the Trade, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 1998, pp. 9–50. URL: https://doi.org/10.1007/3-540-49430-8_2. doi:10.1007/3-540-49430-8_2.

[15] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, JMLR Workshop and Conference Proceedings, Chia Laguna Resort, Sardinia, Italy, 2010, pp. 249–256. URL: http://proceedings.mlr.press/v9/glorot10a.html, iSSN: 1938-7228.

[16] K. He, X. Zhang, S. Ren, J. Sun, Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification, in: 2015 IEEE International Conference on Computer Vision (ICCV), IEEE, Santiago, Chile, 2015, pp. 1026–1034. doi:10.1109/ICCV.2015.123, iSSN: 2380-7504.

[17] P. Costa, P. Larzabal, Initialization of Supervised Training for Parametric Estimation, Neural Processing Letters 9 (1999) 53–61. URL: https://doi.org/10.1023/A:1018671912219. doi:10.1023/A:1018671912219.

[18] D. Mishkin, J. Matas, All you need is a good init, in: Y. Bengio, Y. LeCun (Eds.), 4th International Conference on Learning Representations: Conference Track Proceedings, ICLR, San Juan, Puerto Rico, 2016. URL: http://arxiv.org/abs/1511.06422.

[19] J. Qiao, S. Li, W. Li, Mutual information based weight initialization method for sigmoidal feedforward neural networks, Neurocomputing 207 (2016) 676–683. URL: https://doi.org/10.1016/j.neucom.2016.05.054. doi:10.1016/j.neucom.2016.05.054.

[20] G. Li, H. Alnuweiri, Y. Wu, H. Li, Acceleration of back propagation through initial weight pretraining with delta rule, in: IEEE International Conference on Neural Networks, IEEE, San Francisco, CA, USA, 1993, pp. 580–585 vol.1. doi:10.1109/ICNN.1993.298622.

[21] H. Shimodaira, A weight value initialization method for improving learning performance of the backpropagation algorithm in neural networks, in: Proceedings Sixth International Conference on Tools with Artificial Intelligence. TAI 94, IEEE, New Orleans, LA, USA, 1994, pp. 672–675. doi:10.1109/TAI.1994.346429.

[22] G. E. Hinton, S. Osindero, Y.-W. Teh, A fast learning algorithm for deep belief nets, Neural Computation 18 (2006) 1527–1554. URL: https://doi.org/10.1162/neco.2006.18.7.1527. doi:10.1162/neco.2006.18.7.1527.

[23] H. Larochelle, Y. Bengio, J. Louradour, P. Lamblin, Exploring Strategies for Training Deep Neural Networks, The Journal of Machine Learning Research 10 (2009) 1–40.

[24] S. Z. Seyyedsalehi, S. A. Seyyedsalehi, A fast and efficient pre-training method based on layer-by-layer maximum discrimination for deep neural networks, Neurocomputing 168 (2015) 669–680. URL: https://www.sciencedirect.com/science/article/pii/S0925231215007389. doi:10.1016/j.neucom.2015.05.057.

[25] G. G. Towell, J. W. Shavlik, Knowledge-based artificial neural networks, Artificial Intelligence 70 (1994) 119–165. URL: https://www.sciencedirect.com/science/article/pii/0004370294901058. doi:10.1016/0004-3702(94)90105-8.

[26] I. Ivanova, M. Kubat, Initialization of neural networks by means of decision trees, Knowledge-Based Systems 8 (1995) 333–344. URL: https://www.sciencedirect.com/science/article/pii/0950705196819174. doi:10.1016/0950-7051(96)81917-4.

[27] G. Thimm, E. Fiesler, Neural network initialization, in: J. Mira, F. Sandoval (Eds.), From Natural to Artificial Neural Computation, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 1995, pp. 535–542. doi:10.1007/3-540-59497-3_220.

[28] A. Banerjee, Initializing Neural Networks Using Decision Trees, in: Computational Learning Theory and Natural Learning Systems: Volume IV: Making

Learning Systems Practical, volume Making Learning Systems Practical of *Computational Learning Theory and Natural Learning Systems*, MIT Press, Cambridge, MA, USA, 1997, pp. 3–15.

[29] R. Setiono, W. K. Leow, On mapping decision trees and neural networks, Knowledge-Based Systems 12 (1999) 95–99. URL: https://doi.org/10.1016/S0950-7051(99)00009-X. doi:10.1016/S0950-7051(99)00009-X.

[30] R. Balestriero, Neural Decision Trees, arXiv:1702.07360 [cs, stat] (2017). URL: http://arxiv.org/abs/1702.07360.

[31] S. Wang, C. Aggarwal, H. Liu, Using a Random Forest to Inspire a Neural Network and Improving on It, in: Proceedings of the 2017 SIAM International Conference on Data Mining (SDM), SIAM, Houston, Texas, USA, 2017, pp. 1–9. URL: https://epubs.siam.org/doi/abs/10.1137/1.9781611974973.1. doi:10.1137/1.9781611974973.1.

[32] G. Biau, E. Scornet, J. Welbl, Neural Random Forests, Sankhya A 81 (2019) 347–386. URL: https://doi.org/10.1007/s13171-018-0133-y. doi:10.1007/s13171-018-0133-y.

[33] K. D. Humbird, J. L. Peterson, R. G. Mcclarren, Deep Neural Network Initialization With Decision Trees, IEEE Transactions on Neural Networks and Learning Systems 30 (2019) 1286–1295. doi:10.1109/TNNLS.2018.2869694, conference Name: IEEE Transactions on Neural Networks and Learning Systems.