AI

MDPI

Article

A NEAT Based Two Stage Neural Network Approach to Generate a Control Algorithm for a Pultrusion System

Christian Pommer *, Michael Sinapius, Marco Brysch and Naser Al Natsheh

Institut for Mechanics and Adaptronics, Technische Universität Braunschweig, Langer Kamp 6, 38106 Braunschweig, Germany; m.sinapius@tu-bs.de (M.S.); marco.brysch@bgr.de (M.B.); naser.natsheh@tu-braunschweig.de (N.A.N.)

* Correspondence: c.pommer@tu-bs.de

Abstract: Controlling complex systems by traditional control systems can sometimes lead to sub-optimal results since mathematical models do often not completely describe physical processes. An alternative approach is the use of a neural network based control algorithm. Neural Networks can approximate any function and as such are able to control even the most complex system. One challenge of this approach is the necessity of a high speed training loop to facilitate enough training rounds in a reasonable time frame to generate a viable control network. This paper overcomes this problem by employing a second neural network to approximate the output of a relatively slow 3D-FE-Pultrusion-Model. This approximation is by orders of magnitude faster than the original model with only minor deviations from the original models behaviour. This new model is then employed in a training loop to successfully train a NEAT based genetic control algorithm.

Keywords: neural network; Genetic Programming; Finite Element Modelling; pultrusion; reinforcement learning

Citation: Pommer, C.; Sinapius, M.; Brysch, M.; Al Natsheh, N. A NEAT Based Two Stage Neural Network Approach to Generate a Control Algorithm for a Pultrusion System. *AI* **2021**, 2, 355–365. https://doi.org/10.3390/ai2030022

Academic Editor: Andrea Prati

Received: 12 July 2021
Accepted: 3 August 2021
Published: 5 August 2021

1. Introduction—Neural Networks in Engineering. Just a Modern Buzzword?

Artificial Neural Networks (ANN) are increasingly used in a wide range of engineering applications [1–7]. ANN have proven their excellent pattern recognition capability [8], whether optical [9], acoustical [10] or in other multi-feature datasets [11]. Even though they require high computational power and large high-speed memory for effective training, as well as a lot of, often counter-intuitive, hyperparameters adjustments, they can fit any function if they have a reasonable network complexity. The classic use case as a high-level classification algorithm is but one of their many uses.

The still rapidly increasing computing power enables the use of ANN tremendously. ANNs are already present in smart phones and are considered capable of managing highly complex environments such as modern computer games [12] with their complex control schemes in real time. This capability opens up an extremely wide range of applications.

One application is the control of real engineering systems. The authors of [13] show that neural networks can be used to implement near-optimal control algorithms for difficult-to-control systems with constraints.

One approach for such a control model is the use of reinforcement learning with a genetic algorithm such as the NeuroEvolution of Augmenting Topologies (NEAT) algorithm [14]. The NEAT algorithm enables genetic evolution of a neural network to optimize the behavior for a given problem. The authors of [15] describe such an implementation for a nonholonomic wheeled robot.

This paper takes the same approach and describes the successful implementation of a two-step approach to generate a control neural network using the NEAT algorithm for a pultrusion system like the one displayed in Figures 1 and 2.

Figure 1. Small experimental pultrusion system.

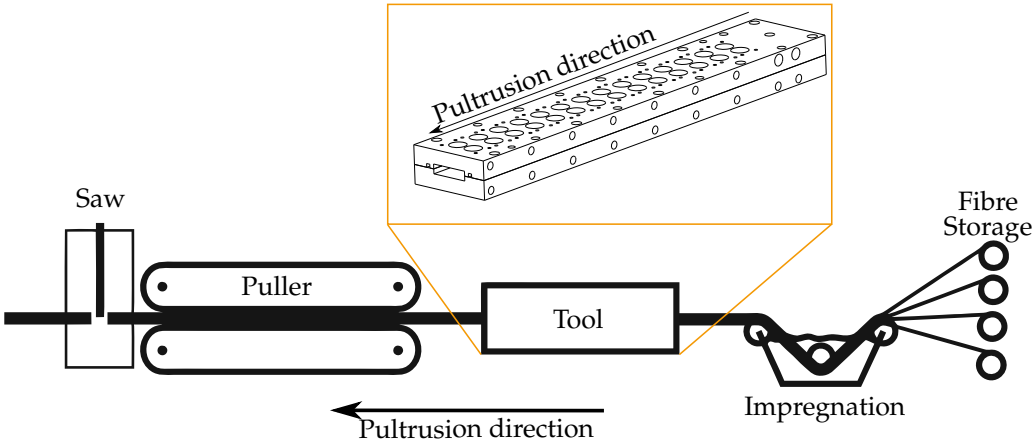

Figure 2. Schematic of a pultrusion line with an example tool.

The pultrusion system is represented by a Finite Element, short FE, model, but it is too slow to complete enough training rounds in a reasonable time. Therefor, an ANN is first trained to approximate the output of the FE model of a pultrusion process. This approximation deviates in accuracy and stability, but is significantly faster than the original model. The time gained by the increased speed is used to train a control algorithm based on NEAT. This evolutionary based algorithm benefits strongly from high-speed simulation due to its high parallelizability and the many training rounds required.

This is an alternative approach to the previously presented method [16], a more traditional control algorithm for a pultrusion based on data received on the resonant ultrasonic spectroscopy [17].

2. Pultrusion Model and Neural Network Training

Training a reinforcement learning based neural network for controlling real engineering systems requires a large amount of training cycles. Using an FE model for this purpose can lead to unreasonably long training times, especially if the task is not yet reasonably defined as is briefly explained later. An alternative to using the FE model directly is to train an ANN so that it behaves like the FE Model. The ANN is much faster than the FE model with comparable accuracy. Both the model and the process used to train the ANN FE model are described in Figure 3. A total of 196,620 samples with a 32 samples per batch

are fed to the network during training. The input and output temperature is normalized in the following way

$$T_{ANN} = \frac{T_{FE}}{255}. \tag{1}$$

The degree of cure does not have to normalizes as it is already in range of 0 and 1. The corresponding FE model is described in more detail in [18].

It consists of two intertwined partial differential equation models shown in the upper half of Figure 3.

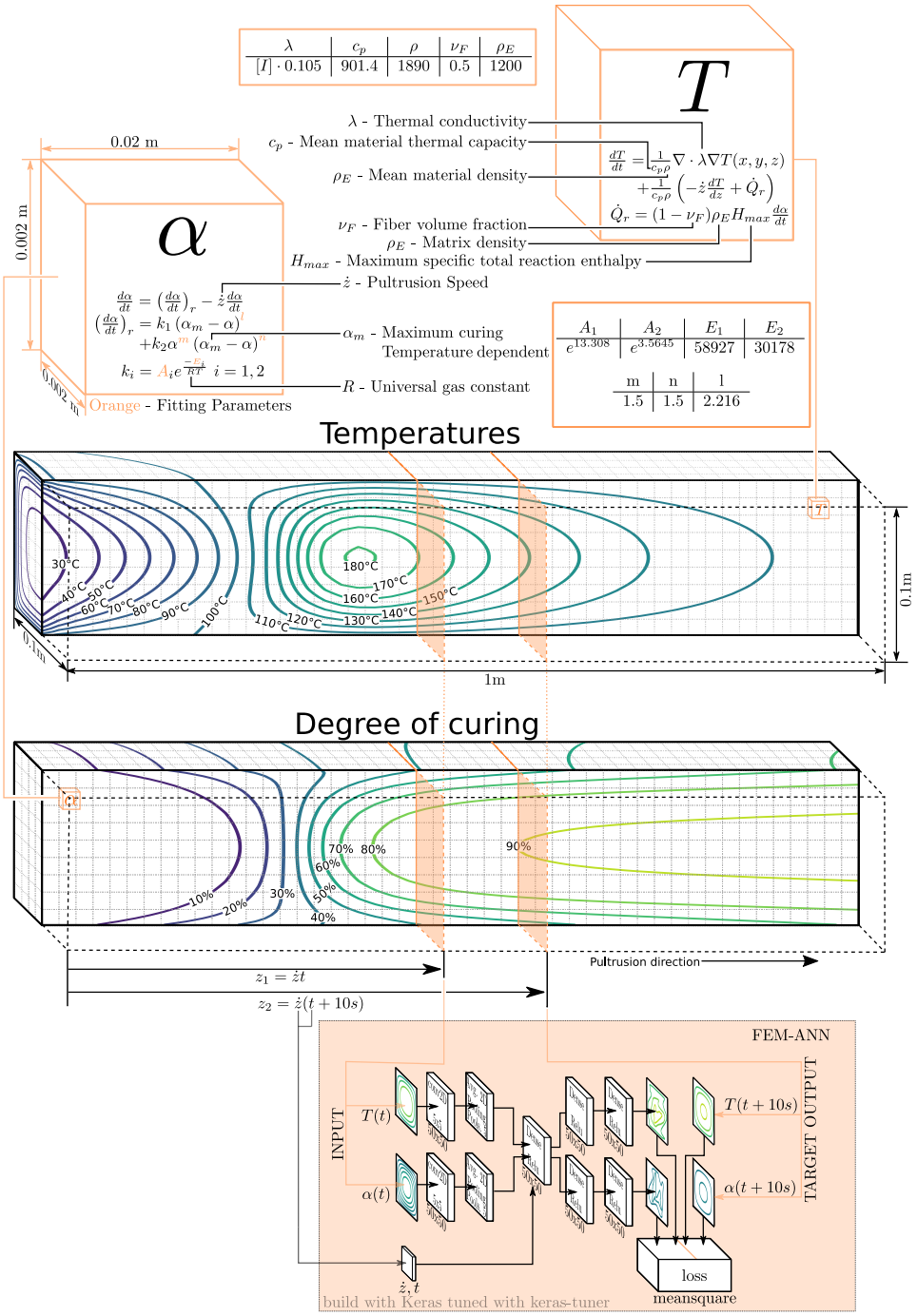

Figure 3. Setup for ANN-FEM-Training.

The ANN based on an FE model is faster than the original FE-Model by a factor of 1000. This allows incredibly fast training cycles in a very complex environment even

without the use of high performance computing (HPC). However, the ANN itself has to be trained on an HPC, especially because hyper-parameter tuning is necessary to maximize accuracy. The ANN later runs in a loop and feeds its own output into the input for the next iteration. This exponentially increases any instability and degradation between FE-model and neural network. Hyperparameter tuning minimizes this divergence. Figure 4 compares the accuracy of the ANN with the original model by calculating the mean square error (MSE) during a series of random speed changes of the pultrusion.

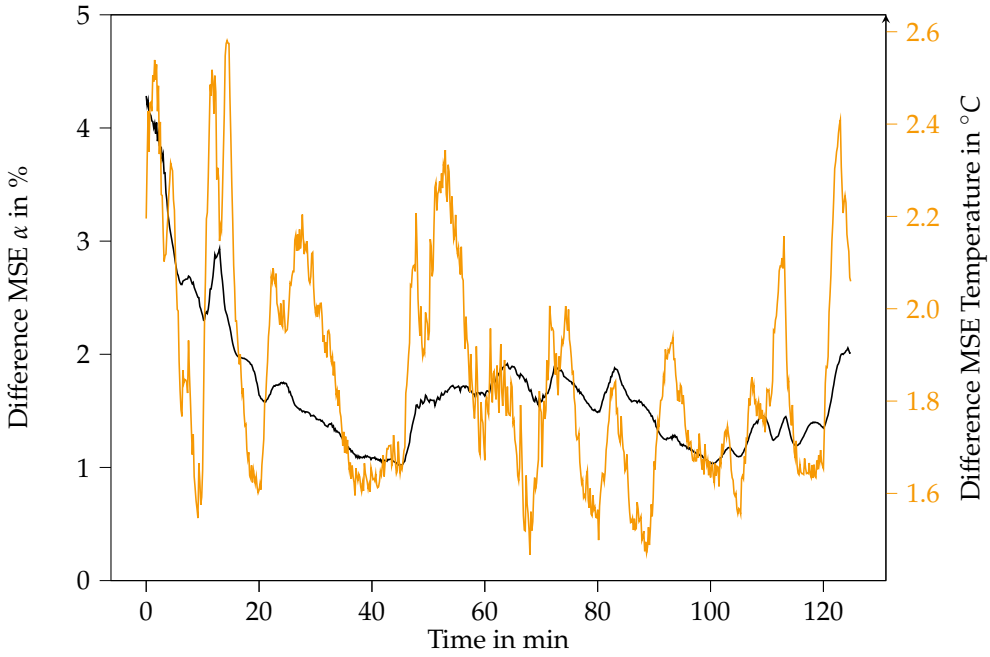

Figure 4. Mean square error between FE model and the ANN during a 16 min initial period and a series of random speed changes between 0.005 $^{m}/_{min}$ and 0.2 $^{m}/_{min}$ every 5 min.

The MSE of the temperature is about 2 °C and the curing error is around 1.5%, showing very good process mapping by the ANN. This is further emphasised by watching the presentation of results in Supplementary Video S1. The training loop is quite simple. A number of slices are continuously moved through the virtual mold. The change in temperature and degree of curing of each slice is calculated by the ANN. The ANN-FE model can handle multiple slices at once without increasing the computation time. The training loop is displayed in Figure 5.

The control ANN is built by applying reinforcement learning techniques. While these techniques can produce very satisfactory results through back-propagation, they require a fixed network structure. While it is feasible to use a fixed network structure together with back-propagation a different approach is chosen. An alternative to back-propagation is the use of genetic algorithms. Genetic algorithms are a good alternative especially when the network structure is not fixed. As shown in [19–21] genetical algorithms are not inferior to other methods especially for simple networks. For ease of implementation, the NEAT-algorithm [14] with the PYTHON implementation [22] is chosen. This algorithm is simple and quick to implement and highly scalable.

The chosen task of creating a control algorithm for the given system is similar to the game Flappybird. In this game, a player has to pass through a number of small-width gates while moving forward. If he fails to pass a gate, the game is over. A number of healing value "gates" is defined. If the healing value α is not within a certain range of the healing target α_{target} after a predefined period of time, the task ends and the accumulated points are stored as that person's fitness. The task itself is executed in several sub-processes, which are connected to the main process via input and output queues. A major advantage of the

queue system is its excellent scalability. The number of sub-processes is only limited by the system resources and the speed increase is linear as long as the system can keep up.

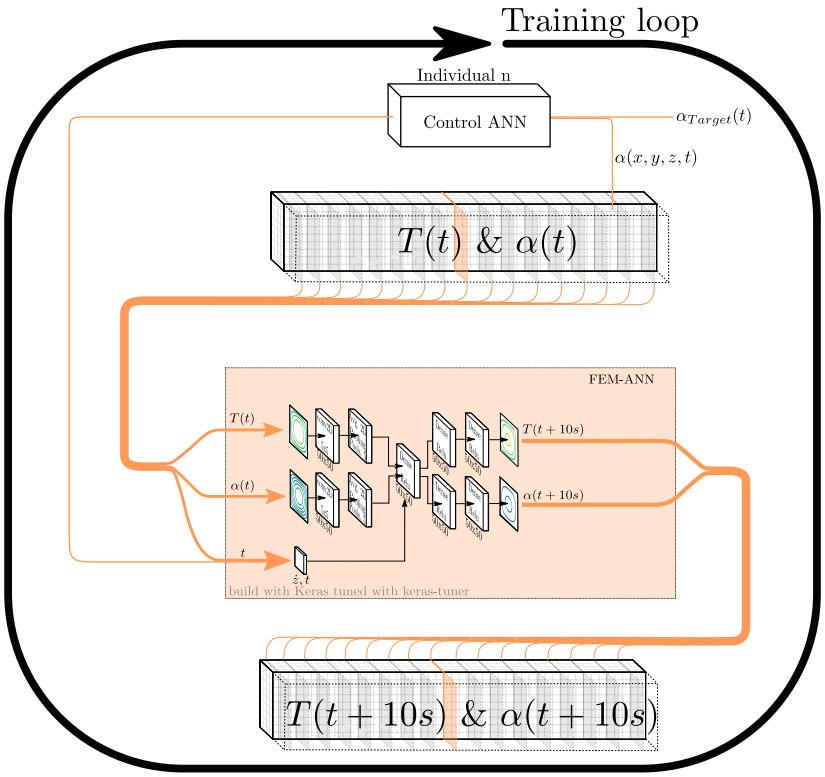

Figure 5. ANN-Training Loop.

One of the most important steps is the implementation of a reward function. This function defines the optimization plane and should have a non-zero gradient at any given point except the global optimum. For this particular task there are three things that should be accomplished:

- The further an individual gets, the higher the fitness function result should be
- If a gate is missed, the fitness function should reflect the distance from the respective gate
- The fitness function should output the highest value when the curing target is reached

Those items can be accomplished by the following fitness function P

$$P = \sum_{t=0}^{\infty} P_r(t) \tag{2}$$

with

$$P_r(t) = \begin{cases} |1 + \alpha - \alpha_{Target}| & t \neq t_{Change} \\ \left(1 - |\alpha - \alpha_{Target}|\right) \cdot 1000 & t = t_{Change} \cap |\alpha - \alpha_{Target}| \geq 1\% \\ 1000 & t = t_{Change} \cap |\alpha - \alpha_{Target}| < 1\% \end{cases} \tag{3}$$

This fitness function in combination with the selected scoring function allows the algorithm to determine the fitness of each individual. The next generation of individuals is then generated by mutation and heredity of the fittest individuals.

The NEAT-algorithm itself is very powerful, but requires the adjustment of a number of hyperparameters to converge in a reasonable manner and time. In particular, the mutation power and the mutation rate are of high importance. These parameter determine the convergence ability and the convergence speed. In the present case, a slower mutation

rate is necessary for convergence. A valid pair of hyperparameters is contained in the Appendix A.

Figure 6 shows the results of this optimization strategy with the aforementioned parameters over several generations. All nodes use the sigmoid activation function. It is very interesting to note that despite the specification of the last 19 time steps as well as the target value for curing as input, not all inputs are used in the control ANN. Looking closely at the generated ANN, it is apparent that the ANN is emulating some kind of PD controller. The integrator is omitted because there is no memory cell. This in turn is a strong indication of the use of a memory cell or memory variable to optimize the behavior. While this is not directly provided by the NEAT Python algorithm, an implementation is possible by adding an additional output that is fed back at the next iteration.

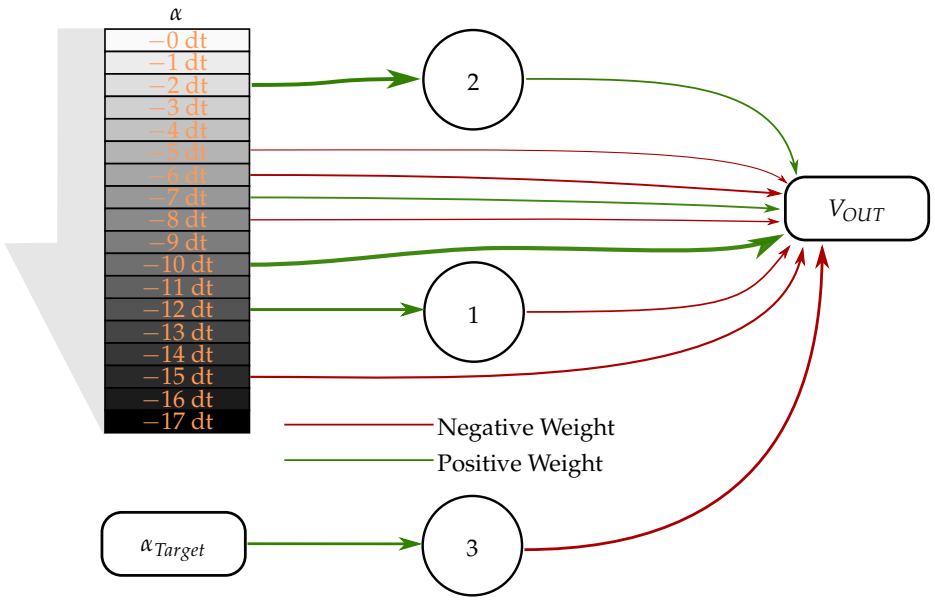

Figure 6. Structure of the best genome found; Line thickness represents weight strength; V_{OUT}, 1, 2, 3 are different Neurons; Inactive Links and some inactive Inputs omitted.

Figure 7 represents the solution of the evaluation problem for the best individual after 270 Generations. The speed is limited between 0.05 $^{m}/_{min}$ and 0.2 $^{m}/_{min}$, which is typical for pultrusion. The black horizontal lines in the upper figure represent the target curing value while the orange curve shows the simulated curing by the ANN-Model. The lower figure shows the speed in the pultrusion system. The figure clearly shows that the algorithm is capable of controlling the FE-ANN to stay within 1% of the target cure value by adjusting the pultrusion speed. The 1% distance is the result of the chosen fitness function. There seems to be a constant deviation of around 1%, which could be reduced by further training or a change in the fitness function. However, a 1% deviation is within an acceptable range and the deviation has to be confirmed first on the FE model.

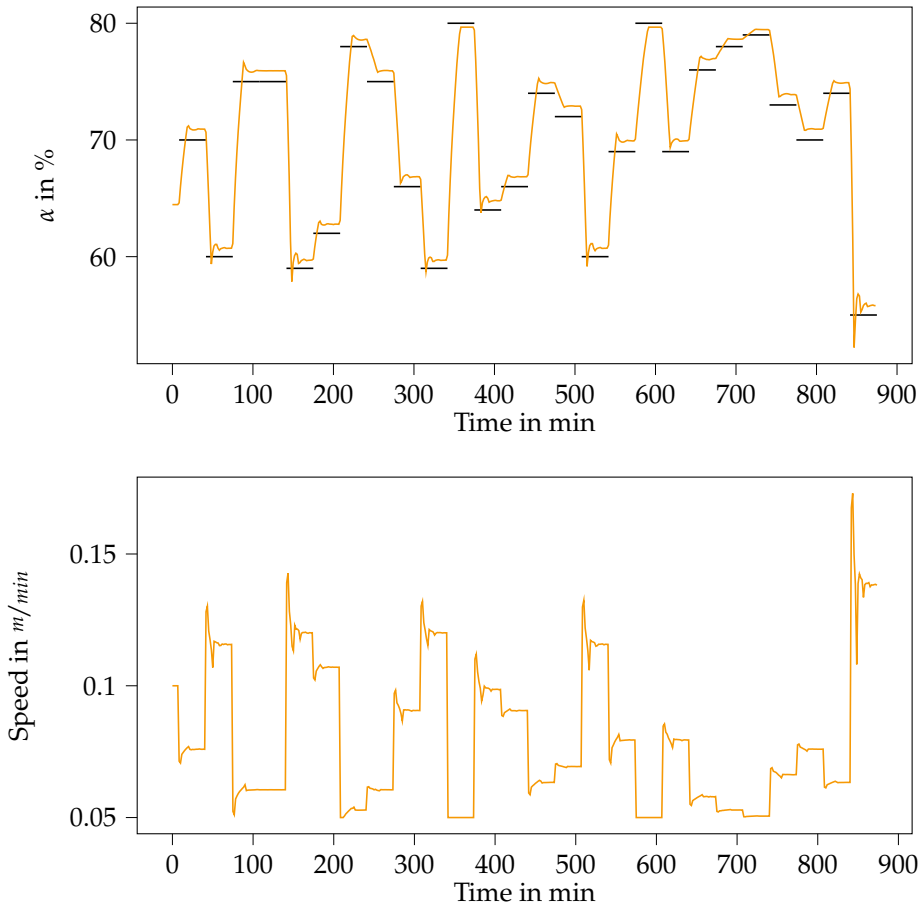

Figure 7. Best run after 270 Generations with curing α in orange, the target curing black in the top figure and Speed set by the ANN in the figure at the bottom.

3. Discussion

As Section 2 shows, a two-step approach can lead to a significant gain in computation time, which directly affects the training time of the control algorithm. While there is a difference in the real FE model and the fitted ANN, the difference is relatively insignificant with respect to the difference in the optimized control algorithm. To demonstrate the ability of the control ANN, the system is compared to the FE model. Figure 8 clearly shows that there is only a minor difference in behaviour for the control-ANN applied to the original FE-Model of around +1%. This is a systematic error in the control algorithm.

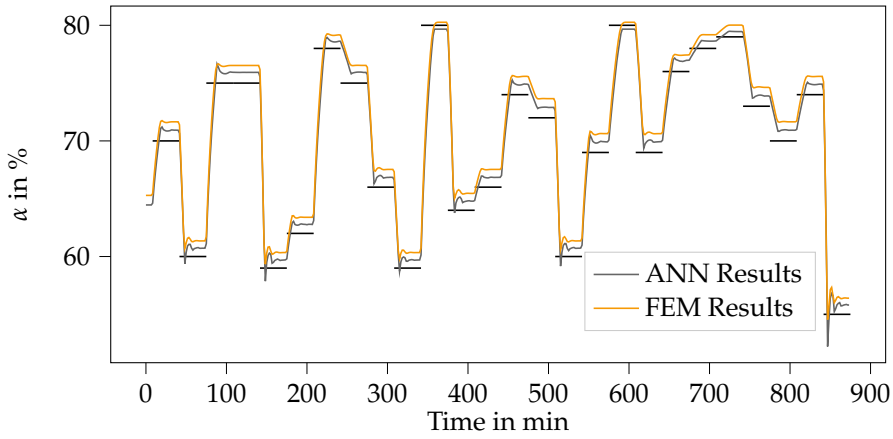

Figure 8. Comparison between the results received by the ANN model and the FE-Model for the NEAT-control algorithm.

A fast and easy fix for this systematic error is a change in the target curing value α_{Target_M} which is fed to the model like,

$$\alpha_{Target_M} = \alpha_{Target} - 1.6\% \tag{4}$$

This small change nearly fully negates the systematic error and leads to a very good deviation of below 0.5% for most curing targets as displayed in Figure 9.

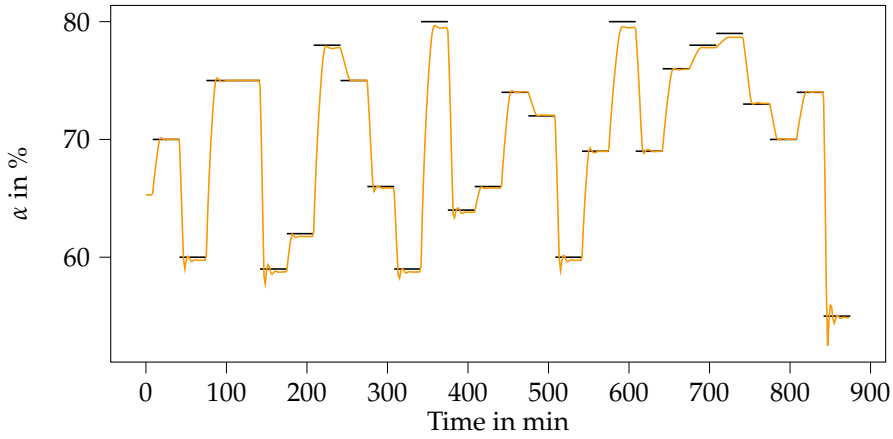

Figure 9. Corrected ANN-Control model tested on the original FE-Model.

The two-step approach of first training an ANN to approximate a FE model and then training a control algorithm on this much faster ANN model yields a significant speed advantage of several orders of magnitude. This approach can be applied to a variety of scientific problems. It works especially well for complex numerical or real world systems without a deterministic scientific model as long as a reasonable amount of training data is available.

The first part, in particular, is responsible for the high speed-up. The second part, i.e., the NEAT algorithm, is easily applicable to any problem where a clear evaluation and fitness function can be created. Especially the fitness function is crucial for a successful implementation. Local minima are particularly problematic, so a well-thought-out implementation is important. The algorithm will find and exploit any loopholes in the formulation. For example, an implementation such as

$$P_r(t) = \frac{1}{n + (\alpha_{target} - \alpha)} \tag{5}$$

for small values of n might lead to oscillation since it might be more valuable to obtain a few values α extremely close to the α_{target}, rather than a large number of values slightly further from α_{target}.

4. Conclusions

This paper shows that a two-step approach, in which the system under study is first approximated by an artificial neural network. In the sense of preprocessing, this step provides a training system that can be quickly examined and used to train an ANN-based control algorithm at a reasonable speed. The speed increase in this case is more than 52×.

The control-model generated is based on the NEAT-algorithm and is implemented with an appropriate evaluation and fitness function. The evaluation function observes and evaluates whether a certain range, e.g., 1% of the target cure value, is reached after a certain period of time. Failing to do so results in an instant loss and stop all further gain in the fitness function.

This algorithm results in a usable control that emulates some kind of PD control based on the observed final structure. Applying the control algorithm on the original FE model results in a viable behavior.

In future research, both the control ANN as well as the FE-ANN, can be further enhanced to not only react to a single set of epoxy curing parameters, but rather to obtain a universal epoxy behaviour with variable parameters. This can be archived by training the FE-ANN on different epoxy parameters and even different geometries. Memory cells need to be added to the control algorithm to explore and store the system behavior of a random epoxy system.

A next step will be the implementation of an auto-encoder algorithm to obtain the degree of curing directly from the sensor signal of the real pultrusion system.

Supplementary Materials: The following are available online at www.mdpi.com/xxx/s1, Video S1: FEM_ANN_DIFF.

Author Contributions: C.P. conceived, M.B. supported with the HPC implementation, designed and performed the experiments. C.P., N.A.N. and M.S. commonly analyzed the data and wrote the paper. All authors have read and agreed to the published version of the manuscript.

Funding: Funding was provided by "Deutsche Forschungsgemeinschaft"; Project number: 289466965.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The NEAT-Algorithm as well as the best model are committed under https://github.com/IMA-ChristianPommer/Pultrususion-FEM-NEAT (accessed on 12 May 2021). (Commit 538ee11).

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ANN	Artifcial Neural Network
NEAT	NeuroEvolution of Augmenting Topologies
FE	Finite Elements
HPC	High Performance Computer

Appendix A. Hyperparamter NEAT-Model

```
[NEAT]
fitness_criterion     = max
fitness_threshold     = 20,000.0
pop_size              = 250
reset_on_extinction   = 1

[DefaultGenome]
num_hidden                = 1
num_inputs                = 20
num_outputs               = 1
initial_connection        = partial_direct 0.5
feed_forward              = True
compatibility_disjoint_coefficient    = 1.0
compatibility_weight_coefficient      = 0.6
conn_add_prob             = 0.05
conn_delete_prob          = 0.05
node_add_prob             = 0.05
node_delete_prob          = 0.05
```

```
activation_default      = sigmoid
activation_options      = sigmoid
activation_mutate_rate  = 0.0
aggregation_default     = sum
aggregation_options     = sum
aggregation_mutate_rate = 0.0
bias_init_mean          = 0.0
bias_init_stdev         = 1.0
bias_replace_rate       = 0.01
bias_mutate_rate        = 0.7
bias_mutate_power       = 0.005
bias_max_value          = 30.0
bias_min_value          = -30.0
response_init_mean      = 1.0
response_init_stdev     = 0.0
response_replace_rate   = 0.0
response_mutate_rate    = 0.0
response_mutate_power   = 0.0
response_max_value      = 30.0
response_min_value      = -30.0

weight_max_value        = 30
weight_min_value        = -30
weight_init_mean        = 0.0
weight_init_stdev       = 1.0
weight_mutate_rate      = 0.8
weight_replace_rate     = 0.01
weight_mutate_power     = 0.005
enabled_default         = True
enabled_mutate_rate     = 0.01

[DefaultSpeciesSet]
compatibility_threshold = 3.0

[DefaultStagnation]
species_fitness_func = max
max_stagnation  = 40
species_elitism = 2

[DefaultReproduction]
elitism            = 3
survival_threshold = 0.2
```

References

1. Alkinani, H.H.; Al-Hameedi, A.T.; Dunn-Norman, S.; Flori, R.E.; Alsaba, M.T.; Amer, A.S. Applications of Artificial Neural Networks in the Petroleum Industry: A Review. In Proceedings of the SPE Middle East Oil and Gas Show and Conference, Manama, Bahrain, 18–2 March 2019; OnePetro: Mnama, Bahrain, 2019. [CrossRef]
2. Gonzalez-Fernandez, I.; Iglesias-Otero, M.A.; Esteki, M.; Moldes, O.A.; Mejuto, J.C.; Simal-Gandara, J. A critical review on the use of artificial neural networks in olive oil production, characterization and authentication. *Crit. Rev. Food Sci. Nutr.* **2019**, *59*, 1913–1926. [CrossRef] [PubMed]
3. Khoo, Y.; Lu, J.; Ying, L. Solving parametric PDE problems with artificial neural networks. *Eur. J. Appl. Math.* **2021**, *32*, 421–435. [CrossRef]
4. Mohandes, S.R.; Zhang, X.; Mahdiyar, A. A comprehensive review on the application of artificial neural networks in building energy analysis. *Neurocomputing* **2019**, *340*, 55–75. [CrossRef]

5. Travassos, X.L.; Avila, S.L.; Ida, N. Artificial Neural Networks and Machine Learning techniques applied to Ground Penetrating Radar: A review. *Appl. Comput. Inform.* **2020**, *17*, 296–308. [CrossRef]
6. Yeganeh, A.; Shadman, A. Monitoring linear profiles using Artificial Neural Networks with run rules. *Expert Syst. Appl.* **2021**, *168*, 114237. [CrossRef]
7. Alanis, A.Y.; Arana-Daniel, N.; López-Franco, C. (Eds.) *Artificial Neural Networks for Engineering Applications*; Elsevier: St. Louis, MO, USA, 2019.
8. Pancioni, L.; Schwenker, F.; Trentin, E. *Artificial Neural Networks in Pattern Recognition*; Springer International Publishing: Cham, Switzerland, 2018; Volume 11081. [CrossRef]
9. Adhan, S.; Pintavirooj, C. Thai sign language recognition by using geometric invariant feature and ANN classification. In Proceedings of the 2016 9th Biomedical Engineering International Conference (BMEiCON), Laung Prabang, Laos, 7–9 December 2016; pp. 1–4. [CrossRef]
10. Trentin, E.; Gori, M. A survey of hybrid ANN/HMM models for automatic speech recognition. *Neurocomputing* **2001**, *37*, 91–126. [CrossRef]
11. Paul, P.S.; Varadarajan, A.S. ANN assisted sensor fusion model to predict tool wear during hard turning with minimal fluid application. *Int. J. Mach. Mach. Mater.* **2013**, *13*, 398. [CrossRef]
12. Gourdeau, D.; Archambault, L. Discriminative neural network for hero selection in professional Heroes of the Storm and DOTA 2. *IEEE Trans. Games* **2020**, 1. [CrossRef]
13. Zhang, H.; Luo, Y.; Liu, D. Neural-network-based near-optimal control for a class of discrete-time affine nonlinear systems with control constraints. *IEEE Trans. Neural Netw.* **2009**, *20*, 1490–1503. [CrossRef] [PubMed]
14. Stanley, K.O. Efficient Evolution of Neural Networks through Complexification. Ph.D. Dissertation, The University of Texas at Austin, Austin, TX, USA, 2004.
15. Caceres, C.; Rosario, J.M.; Amaya, D. Approach of Kinematic Control for a Nonholonomic Wheeled Robot using Artificial Neural Networks and Genetic Algorithms. In Proceedings of the 2017 International Conference and Workshop on Bioinspired Intelligence (IWOBI), Funchal, Portugal, 10–12 July 2017; pp. 1–6. [CrossRef]
16. Pommer, C.; Sinapius, M. Proof of Concept for Pultrusion Control by Cure Monitoring Using Resonant Ultrasound Spectroscopy. *J. Compos. Sci.* **2020**, *4*, 115. [CrossRef]
17. Pommer, C.; Sinapius, M. A Novel Approach to Monitoring the Curing of Epoxy in Closed Tools by Use of Ultrasonic Spectroscopy. *Sensors* **2017**, *18*, 96. [CrossRef] [PubMed]
18. Pommer, C. Geregelter Pultrusionsprozess mit In-Situ-Aushärtungsüberwachung. Ph.D. Dissertation, DLR, Deutsches Zentrum für Luft- und Raumfahrt, Köln, Germany, 2019.
19. Gupta, J.N.; Sexton, R.S. Comparing backpropagation with a genetic algorithm for neural network training. *Omega* **1999**, *27*, 679–684. [CrossRef]
20. Such, F.P.; Madhavan, V.; Conti, E.; Lehman, J.; Stanley, K.O.; Clune, J. Deep Neuroevolution: Genetic Algorithms Are a Competitive Alternative for Training Deep Neural Networks for Reinforcement Learning. *arXiv* **2017**, arXiv:cs.NE/1712.06567.
21. Awolusi, T.; Oke, O.; Akinkurolere, O.; Sojobi, A.; Aluko, O. Performance comparison of neural network training algorithms in the modeling properties of steel fiber reinforced concrete. *Heliyon* **2019**, *5*, e01115. [CrossRef] [PubMed]
22. McIntyre, A.; Kallada, M.; Miguel, C.G.; da Silva, C.F. Neat-Python. Available online: https://github.com/CodeReclaimers/neat-python (accessed on 12 May 2021).