How can deep learning be used to improve the heliostat field calibration, even with small data sets? - A transfer learning comparison study

Max Pargmann[1, a)] and Daniel Maldonado Quinto[2, b)]

[1]*Phd-Student, Institute of Solar Research, German Aerospace Center (DLR), Linder Hoehe, 51147 Cologne (Germany)*
[2]*Dr.-Ing.-Researcher, Institute of Solar Research, German Aerospace Center (DLR), Linder Hoehe, 51147 Cologne (Germany)*

[a)] Corresponding author: max.pargmann@dlr.de
[b)] daniel.maldonadoquinto@dlr.de

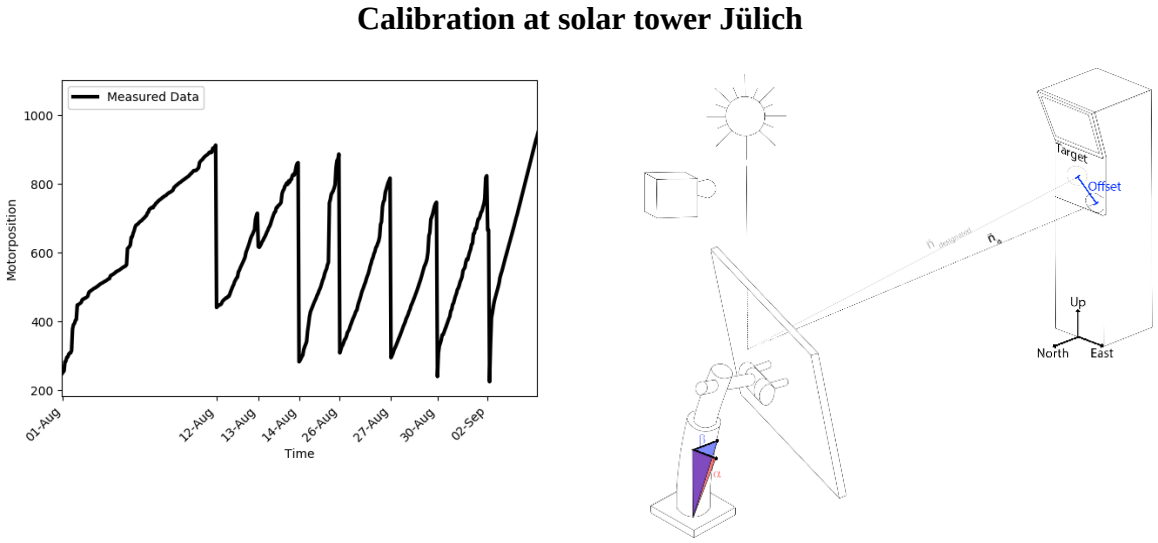
Abstract. A precise and reliable alignment of the two-axis heliostat tracking is of great importance for an efficient operation of solar power towers. In order to minimize the tracking error of heliostats, especially in large plants, it is essential to recalibrate the heliostat control unit regularly. Conventional calibration methods with regression can meet the requirements of frequent and regular use, but they cannot adequately account for the many factors that influence alignment. Deep learning algorithms have made remarkable progress in recent years and have the potential to reduce the number of calibrations over time while reducing tracking errors. However, neural networks are still rarely used for such purposes, because such algorithms usually require an extremely large amount of data to map the individual heliostat errors. We present a comparison of different pre-train studies for neural networks to reduce the amount of data per heliostat which are needed to improve the accuracy compared to a state-of-the-art method by applying supervised as well as unsupervised pretraining.

Introduction

For a proper operation of a solar tower power plant it is necessary to calibrate the heliostats regularly, because their ability to track the sun correctly directly influences the performance of the power plant. To keep the cost of solar power plants low, it is essential to keep the price of the heliostats also as low as possible. In order to achieve this goal, cheaper or fewer components are used to build the heliostats, which can reduce the accuracy of solar tracking. New heliostat designs can counteract this, but a key element of this development will be a cost effective, accurate and fast calibration.

For conventional calibration the heliostat control unit includes a numerical motion model for each individual heliostat, which determines the motor positions for a required nominal alignment depending on sun angles, heliostat position and mechanical construction parameters. The numerical model can represent the real behavior, limited by the quantity of influencing factors, only within a certain model error. For example, not all design uncertainties can be considered and measured after construction. It is also impossible to take bearing clearance, backlashes or ageing processes into account without making the model too complex. This is where deep learning comes into play. Neural networks work like an improved regression method which is independent of the number of influencing factors and without limitations implied by the default of a function template [1]. The downside is the amount of needed data, which is rather limited per heliostat. To get over this, appropriate pre-training approaches have shown in the past that neuronal networks can manage tasks with much less data [2][3]. In this paper three different supervised as well as unsupervised pre-training methods applied to a dataset measured at the solar tower in Jülich are compared to each other and estimations are given about their

potential for heliostat calibration. For validation the new calibration method is compared to the state-of-the-art calibration method used at the solar tower Jülich and former published results [4]. In the next chapters, first an outlook over the state-of-the-art algorithm and neural networks are given. Then the pre-training methods and the workflows are described, followed by the results. The paper is closed with a discussion and an outlook on the upcoming challenges.



(a) (b)

FIGURE 1. (a) shows the azimuthal motor movement of one heliostat in Jülich. The dataset is used for training and testing of the state-of-the-art as well as the NN calibration and includes 500 calibration points measured. [4] (b) Sketch of the heliostat calibration process. The shown heliostat has some deformation on its pedestal, which leads to an offset on the target, compared to a heliostat without errors. The shown error can be decomposed into two rotational misplacements α,β. Then a regression Algorithm tries to minimize the angle between the erroneous ($\vec{n}_{is}$) and the ideal heliostat ($\vec{n}_{designated}$) by varying these errors inside of a geometric model.

At the solar tower power plant in Jülich and in most other commercial power plants each heliostat is calibrated with a method developed by Stone [5]. While in operation each heliostat is moved individually from the receiver to a Lambertian white target, its focal spot is detected by a camera. An algorithm determines its centroid of area, which is then stored together with the motor and sun positions.
Figure 1 (a) shows the azimuthal motor position of the heliostat during calibration, which will be, without loss of generality, the main value of the following discussion. The dataset for the following simulation contains 500 datapoints, which were taken in the regular calibration process.

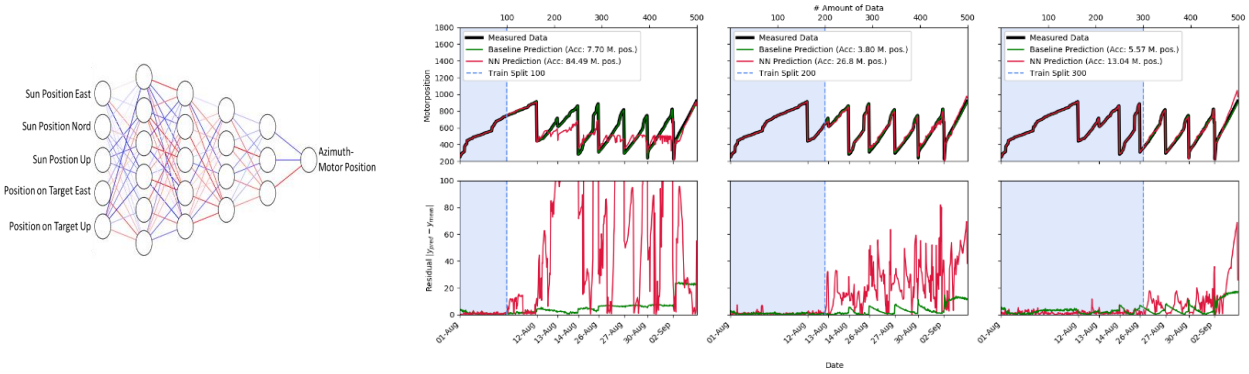
To determine the errors of the heliostat, a physical error based geometric model is used in Jülich. A first function calculates the designated alignment ($\vec{n}_{designated}$). For this, the sun position and some heliostat specific parameters are used, with which then the alignment of the heliostat can be calculated. A second geometric model then derives the position where the heliostat *is* actually pointing to ($\vec{n}_{is}$). The next step is to minimize the angle between the *should* and the *is* alignment, by varying the error parameters included in both functions. In Jülich, this is done by the Levenberg-Marquardt algorithm. The function to minimize is as follows:

$$F = \min_{\alpha,\beta,\gamma,\delta,\dots} \sum_{i=1}^{N} cos^{-1}\left(\vec{n}(\alpha,\beta,\gamma,\delta,\dots)_{is,i} \cdot \vec{n}(\alpha,\beta,\gamma,\delta,\dots)_{designated,i}\right) \quad \text{i)}$$

$\vec{n}_{is}$ and $\vec{n}_{designated}$ are the heliostat surface normals, pointing towards the focal spot. The function sums over all measured calibration points *i*. $\alpha,\beta,\gamma,\delta...$ are error parameters and are varied and determined by the regression algorithm (compare figure 1 (b)). In Jülich 8 of those are used. 6 describing rotational displacements and 2 different gear ratios of the used stepping motors. In general, the used function template does not have to be a physics based geometric model, a mathematical approach (e.g. a polynomial) where the error parameters have no physical meaning would also be possible. Then the calculated error parameters are stored and for the following operation the required alignment can be calculated by computing the bisector between the surface normal and the sun vector iteratively. In the following discussion the error based geometric model used in Jülich will be also called *baseline model*, because every accuracy of each tested algorithm is validated against this model. Nevertheless, even if it is called a baseline model, it is not needed to validate the results, because the predicted motorpositions of the test data set could be converted directly into *mrad*. In a first approximation, the accuracy of the examined heliostat can be approximated with *0.8mrad* per increment. However, a conversion was not made due to the angle dependencies of the conversion occurring in higher approximations.

Neural Networks for Heliostat Calibration

Neural networks (NNs) are a new type of regression algorithm. Except for a predefined underlying function template, a network consisting of interconnected nodes defined by an activation function with individual weights and biases. The nodes (or neurons) are structured layer wise. The input values are passed to the first (input-)layer of neurons. Depending on the varying input values the activation functions propagate the information, generally coded without any physical meaning, through the network. Due to this these layers are also called hidden-layer. At the last (output-)layer during the first iteration random outputs are compared to the desired values. Depending on the deviation the nodes are then adapted by the so-called backpropagation [6].
With this technique and sometimes billions of free parameters neural networks are capable to fit also the most complex functions. In recent years neural networks became the state-of-the-art algorithm for classification [7] and generation [8] tasks, but also regression tasks have made recognizable progress [9].



(a) (b)

FIGURE 2. (a) The network structure used for the heliostat calibration task. Each node represents an activation function, for this task the scaled exponential linear units (SELUs) are chosen. The red and blue lines represent the interconnection between the nodes. (b) The predictions of the baseline model and the not pretrained neural network shown in (a). The Accuracy of the neural network rises with the amount of data, while the baseline model stagnates at some point. Nethertheless the NN never reaches a competitive accuracy.

For the heliostat calibration a simple fully connected network is used, whose structure is shown in fig. 2. For training self-normalizing neural networks are used. Without pretraining, the network is initialized as suggested by Klammbauer [10]. Compared to more famous network structures like ResNet [11] or LeNet [12] the selected network is relatively small. This is due to the fact that the state-of-the-art heliostat calibration takes up very few calibration points in the regular operation mode. With a small database, larger networks would lead to an overfitting and thus to a deterioration of the prediction accuracy.

A well-trained network with the chosen in- and outputs calculates from the current sun position and the wanted position on the receiver the therefore needed heliostat motorposition. To reduce the complexity of the problem the network has only one output describing a horizontal or a vertical movement. So, two trained networks are needed for a full heliostat movement description.
The downside of neural networks is clearly the needed amount of data, which is in most cases much higher than for a normal regression. But this can be counteracted by an adequate pretraining. Here the neural networks aren't initialized randomly but with weights and biases which are already close to the global optimum and are derived in the pretraining step.

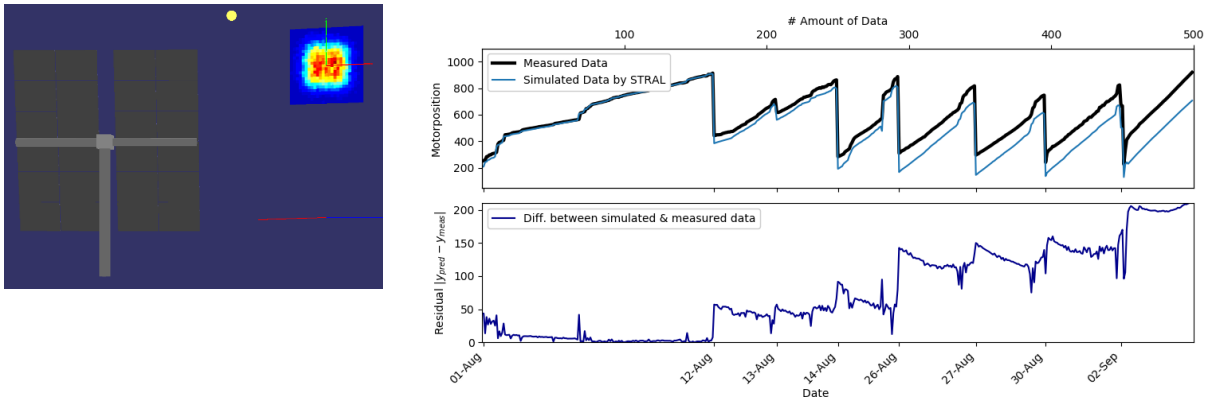
Pretraining Methods

The choice of initial parameters of a deep neural network can have a significant effect on the accuracy of a neural network. If the nodes are not randomly initialized, but some training has been done beforehand, we talk about pretraining. We differentiate between two variants, the supervised and the unsupervised case. In the supervised pretraining the neural network learns to solve a problem e.g. an image classification task with a database, which is similar to the actual training database, but available in large quantities. With this procedure, the network already learns to adapt relevant features inside the network structure. In the given example this can be the ability to extract sharp edges inside the image or more complex features like eyes, which also can help in the actual training step. In the unsupervised case, no extra dataset is needed. Instead the network tries to capture structure in the input distribution. This pretraining can be considered as a regularization method. By preactivating the nodes needed to solve the task in advance, the local parameter space becomes more complex, which renders it more difficult to travel significant distances via the backpropagation method [13] and therefore restricts (regularizes) the process. After the pretraining, the actual training process starts like the not pretrained neural network training, but with the neurons are initialized with the weights and biases from the pretraining. Also the learning rate, a factor which scales how much the weights and biases change per training step is much smaller (starting with 10^{-5} and is reduced over time) to preserve the pretain progress. In this paper 3 pretraining methods, 2 supervised, 1 unsupervised are analyzed for the heliostat calibration case. In the following we will go through the different pretraining techniques

Supervised Pretraining with the Baseline Model

Another possibility to create a large dataset is to use the baseline model used for the state-of-the-art calibration, mentioned earlier. Since this model is already used for heliostat control on solar towers, it is perfectly suited to calculate the expected heliostat position for given sun positions. It must be considered that the preliminary regression algorithm is also dependent on the amount of data (compare fig. 2).
Because of this it is possible that the test accuracy of the currently trained geometric model is worse than one with a smaller dataset. In figure 2 this is the case for the 200 and the 300 train set. To train a neural network with a train set including 300 calibration points the baseline model with the best test accuracy is used to create the artificial dataset instead of the equally large train split.
For this reason, the data required for the pre-training may only be generated from a geometry model, which received as much or less training data as the neural network will receive in the actual training process.

Supervised Pretraining with Raytracer Dataset



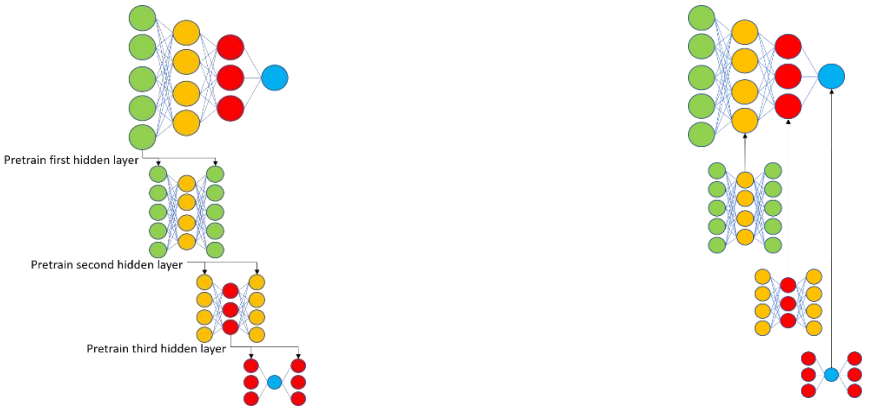
(a) (b)

FIGURE 3. (a) Picture of the simulated heliostat in STRAL including the target picture. (b) The upper graph shows the measured calibration data (black line) compared to the data simulated by the raytracer STRAL (light blue line) for the same time stamps. The lower graph shows the same data but relative to each other. While the shape of the curve looks pretty similar to the measured data, there is clearly a linear offset, which grows over time.

For a supervised pretraining a larger dataset similar to the original problem is needed. A raytracer like the software [14] can provide a sufficient large dataset, similar to the original task for this pretraining step. The software generates traced rays from the sun, which are reflected at a geometric model of the heliostat facets (compare fig. 3 (a)) Each heliostat has a defined aim point on the receiver. Knowing the aimpoint and the sun position, both can be used to determine the heliostats alignment, which is expressed by a horizontal and a vertical angle. Because the simulated heliostat and the real heliostat do not share the exact same movement geometry the angles calculated by STRAL and the real ones, derived by the measured motorpositions, are not the same and must be corrected, by offsets, stretching or compression which are adapted iteratively. In fig. 3 (b) results of the simulation and the adaption compared to real measured calibration data are shown. Because the simulated values did not fit the real measured data perfectly (only using one offset and one scaling factor), it was decided to fit only the first day as good as possible. While the shape of the simulated curve (upper graph) is very comparable to the measured one, the spacing between the two curves increases over time.

For the pretraining, 4000 random distributed, but real existing sun positions between 1. August and 1. October are simulated. The dataset was then split, separated in time, into a (pre-)training and a test set (90:10), with which the network was then trained.

Unsupervised Pretraining – Stacked Autoencoder



(a) (b)

FIGURE 2. (a) First unsupervised pretraining step. In this "greedy layer-wise" pretraining several autoencoders are created. The first takes the original training data as input and output layer and is trained until no enhancement is recognizable anymore. After this, for every layer of the original neural network, a new autoencoder is created and trained using the one step earlier trained hidden layer as the new input and output layers. (b) After all autoencoders are fully trained the original neural network is initialized by the trained hidden layers of each. After this the real training step takes place

As described before in the unsupervised pretraining there is no need for an additional dataset, instead an unsupervised training with the same dataset used for the actual training process takes place. Actually, unsupervised pretraining is also the way to go, if there is an additional dataset but with unlabeled data. The pretraining takes place in a so-called *greedy layer wise* [9] process. For this every layer of the neural network is trained individually inside a shallow stacked autoencoder (SAE), a symmetrical neural network where input and output layer are the same (compare fig 2.). The deep neural network is then initialized with the trained hidden layers of the autoencoders for the actual training process. Under the assumption that it is easier to train several flat networks than one deep network, this approach helps to bring the network close to a local optimum even before the actual training.

Workflow

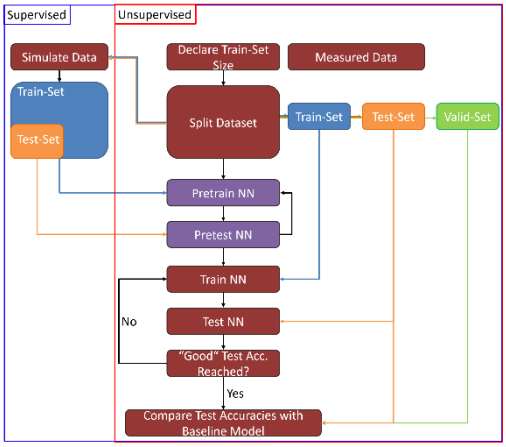


FIGURE 5. The workflow for the supervised as well as the unsupervised pretraining. For the generation of the pre-training datasets the model always used is the one which has achieved the best result with a dataset of the same or smaller size.

The main goal, is to increase the prediction accuracy compared to the baseline model by also holding the needed amount of data as small as possible. Therefore, the dataset, containing 500 calibration points is split into different training and an equally large test set. All calibration points, which are not included in one of the two sets are declared as a third, the validation set, which helps to test generalization afterwards. The datasets are strictly separated in time. For example, a trainingset is declared containing the earliest 50 datapoints, then the following 50 datapoints are declared as the test set. All other points are inside the validation set and are not shown to the network over the whole training process. In case of supervised pretraining the same (or less, if it generates a better pretraining test accuracy) training and test data are used to generate artificial training data for pretraining the network. At least the accuracy is validated with the complete dataset against the baseline model.

Results

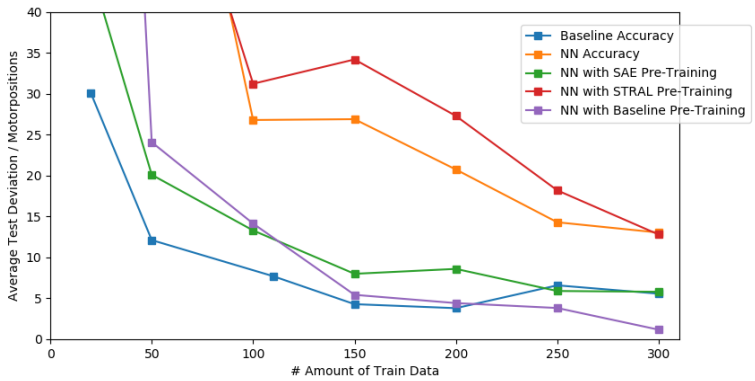


FIGURE 6. Results of all pretraining methods, the baseline model and the not pretrained NN. The unsupervised pretraining as well as the supervised pretraining with the baseline model reach a competitive accuracy compared to the baseline model.

All tested algorithms as well as the baseline model show a clear dependency on the amount of train data. Although the accuracy of the baseline model falls by far the fastest, it is also the only curve that stops improving on a medium accurate level within the available dataset range. The orange line shows the NN test accuracy without pretraining. Surprisingly the pretraining done with STRAL makes the prediction accuracy slightly worse. Which might be due to the strong deviation between simulated and real data. Because the learning rate is considerable smaller while training, compared to the pretraining step, in order to preserve the pretraining progress, the network might be stuck in a local minimum which is far from the global optimum. Both the network that was trained unsupervised and the network that was trained by the baseline model achieve an accuracy that is approximately equal to the baseline model. The over all best accuracy is achieved using the supervised baseline model pretraining [4].
Figure 7 shows the respective best prediction results of every method, which is for all neural networks at 300 training data points. Also, the best result of the baseline model, at 200 datapoints, is drawn in all plots for better comparison. All Neural Network algorithms have in common that the prediction accuracy of the training set (blue background) is very good, but this does not necessarily lead to a high test prediction accuracy. Especially the not pretrained as well as the model pretrained with STRAL do have a lack of generalization, leading to very poor test accuracies. For the unsupervised pretraining the situation is different. Although an accuracy was achieved within the available dataset that is similar to the baseline model, more data was needed to achieve this. The picture may change for an even bigger dataset, because the prediction accuracy of the neural network is monotonically falling inside the test range, while the baseline model already stagnates at 200 train data. But to confirm this, more data would be needed. The model of the pretrained with the baseline model is with an average deviation of 1.21 increments by far the best.

Discussion and Outlook

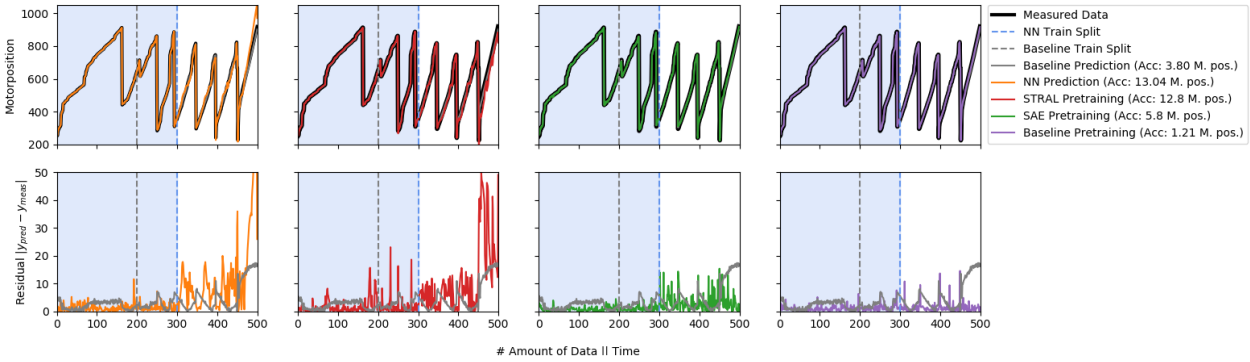


FIGURE 7. The best predictions of all tested methods. For a better comparability the best result of the baseline model, using 200 calibration points, is also drawn inside every graph.

The results have shown, that pretraining is the key for the use of neural networks for heliostat calibration. Even if only one network reached a better accuracy then the baseline model. The very steep rise in accuracy indicates that the other algorithms become completable or even better, with a larger dataset.

Although pretraining with STRAL helped least, the fact that the results were even worse than without pretraining suggests that the network has converged to a false local minimum. Nevertheless, the methodology should have more potential than it shows. The simulated data with which the network was pretrained, were created by a simulation of a perfect heliostat with no errors, and an oversimplified movement geometry, and therefore had a huge discrepancy to the real measured values. By implementing a more realistic heliostat model, it should be possible to create calibration data closer to the real measurements. With this in mind, the pretraining with raytracer data remains a relevant concept, since a very accurate pretraining can be performed even with very few real calibration points.

The unsupervised pretraining works well, although not the best. Because the methodology is independent of any pretraining dataset, it is hard to say how this can be improved. As with every neural network architecture there are a few dozens of hyperparameters which can be adjusted to improve the results. This has already been tried, but there are certainly still possibilities for optimization. However, it should not be forgotten that the goal is not to control one heliostat, but several thousands independently. This means that although some parameters can be optimized in real operation, for example by Bayesian search, such an optimization will tend to find on average a good rather than an optimal result. So, there is still a potential to improve the results of the unsupervised training, but they can still be interpreted as average good optimization like it would take place at the solar tower. Therefore, the unsupervised pretraining is a stable method to reach completable accuracy compared to the state of the art calibration, with a moderate amount of data. Whether it is possible to reach even higher accuracies with a bigger dataset still needs to be tested, but seems likely.

The best prediction accuracy, achieved by the baseline pretrained neural network, is close to 3 times better than the baseline model itself.

It still has some narrow outlines, but a lot of them are at the same position, where also peaks of the baseline model prediction are located. This indicates that these peaks are real and not an artificial product of the regression, but an erroneous behavior of the heliostat. It does not have to be this error, but from the daily operation of the power plant it is known that the heliostats sometimes miscount by a few increments. However, for a given geometric model such an (not random) error would lead to high error, which may decrease over time, because the axis where the error occurred becomes less important for alignment depending on the angle of the sun, which is the peak behavior of the baseline model between 200 and 450 datapoints. Whereas such an error, which is not contained in the geometry model of the baseline model, could make the regression results much worse, a neural network could expect this, if this is reproducible and if it is also included in the training dataset. Since such an error shifts slightly every day in time, it is possible that the network tries to compensate for this error, but cannot predict the exact time. This slight time difference could lead to the narrow peaks that can be seen at 250, 290, 350 and 450, which both the NN and the baseline model have in common.

In summary, it can be said that neural networks can be used as an alternative regression method to heliostat calibration, also thanks to the possibility of pretraining starting around 300 calibration points.

Outlook

Although the results seem promising, questions remain concerning the stability of the process and the network's ability to generalize in long term. Because the results depend on many factors e.g. how the pretraining set is created, a much bigger dataset must be measured, also from different heliostats, to test the stability of the procedure at longterm. Simulation data can also be used for this purpose, which is currently being done, but these data do not reach the complexity of real measurements, where errors such as noise, miscounting or more general non-linear errors play a greater role.

REFERENCES

1. Z. Lu, H. Pu, F. Wang, Z. Hu, L. Wang, *The expressive power of neural networks: A view from the width (*Advances in neural information processing systems, 2017), pp. 6231-6239
2. S. Feng, H. Zhou, H. Dong, *Using deep neural network with small dataset to predict material defects* (Materials & Design 162, 2019) pp. 300-310.
3. A. Noguchi, T. Harada, Image generation from small datasets via batch statistics adaptation (2019) arXiv:1904.01774
4. M. Pargmann, D. Maldonado Quinto, P. Schwarzbözl, *High Accuracy Data-Driven Heliostat Calibration and State Prediction with Deep Neural Networks* (Solar Energy, Submitted)
5. K. W. Stone, *Automatic heliostat track alignment method, (*US Patent 4,564,275 , 1986)
6. Goodfellow, Ian; Bengio, Yoshua; Courville, Aaron (2016). *6.5 Back-Propagation and Other Differentiation Algorithms*. (Deep Learning. MIT Press) pp. 200–220. ISBN 9780262035613.
7. Q. Xie, M.-T. Luong, E. Hovy, Q. V. Le, *Self-training with noisy student improves imagenet classification*, (Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020), pp. 10687-10698.
8. S. Pidhorskyi, D. Adjeroh, G. Doretto, *Adversarial latent autoencoders*, arXiv:2004.04467v1 (2020).
9. S. Feng, H. Zhou, H. Dong, *Using deep neural network with small dataset to predict material defects* (Materials & Design 162, 2019) pp. 300-310.
10. G. Klambauer, T. Unterthiner, A. Mayr, S. Hochreiter, *Self-normalizing neural networks*, (CoRR abs/1706.02515, 2017). URL: http://arxiv:org/abs/1706:02515
11. Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, *Deep Residual Learning for Image Recognition (*Computer Vision and Pattern Recognition, 2015), arXiv:1512.03385
12. Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, *Going Deeper with Convolutions (*Computer Vision and Pattern Recognition, 2014) arXiv:1409.4842
13. Dumitru Erhan, Yoshua Bengio, Aaron Courvill *Why Does Unsupervised Pre-training Help Deep Learning? (*Journal of Machine Learning Research 11, 2010), pp- 625-660
14. Belhomme, B. and Pitz-Paal, R. and Schwarzbözl, P. and Ulmer, S. *A New Fast Ray Tracing Tool for High-Precision Simulation of Heliostat Fields (*Journal of Solar Energy Engineering 131, 3, 2009)