High Accuracy Data-Driven Heliostat Calibration and State Prediction with Deep Neural Networks

Max Pargmann[a,*], Daniel Maldonado Quinto[a], Peter Schwarzbözl[a], Robert Pitz-Paal[a]

[a]*German Aerospace Center (DLR), Institute of Solar Research, Linder Höhe, D-51147 Köln, Germany*

Abstract

The efficiency of solar tower power plants depends strongly on the ability to reflect the sun light onto a defined point on the receiver. Due to the high demands on the heliostats to achieve high accuracy at low costs, a regular calibration is necessary to reduce the tracking error. In this paper a new method for improving existing calibration methods using deep learning is presented. The results are validated by using calibration data recorded at the Solar Tower Jülich with the example of one heliostat. Through a combination of Self-Normalizing Neural Networks and transfer learning it is possible to benefit from the advantages of neural networks already with a training dataset of only 300 measuring points and with that to achieve a test accuracy which was approximately three times more accurate than the best result of the compared regression algorithm used in Jülich. Furthermore we give recommendations on the structure of the dataset and the neural network (NN) pretraining necessary for these results.

Keywords: Concentrating solar power, Solar tower power plant, Heliostat aiming, Artificial intelligence, Neural Networks

1. Introduction

The heliostat field of a solar tower power plant can cause up to 50% of the capital expenses (CapEx) and thus contributes significantly to the levelized cost of energy. The ability of these heliostats to redirect the sun to the correct designated position on the receiver directly affects the amount of electricity generated and is responsible for protecting the receiver from local overheating to ensure a long term operation. To reduce the levelised cost of electricity to 0.6 USD/kWh or below, the costs per Heliostat must not be more than 75 USD/m^2 (Pfahl et al., 2017) while maintaining the sun tracking accuracy.
The accuracy is influenced by various factors, these include misalignment by torsion, mechanical deformation, gear backlash or local wind speeds. For more cost-effective heliostats these sources play a greater role because less material or cheaper motors are used. New heliostat designs or cheaper production methods may counteract this, but the key element of this development will be an intelligent calibration system with low hardware costs which is part of current research. This paper will frist give a brief introduction to neural networks and the current calibration system in Jülich. Afterwards the workflow, how the data was gathered and how the (pre)training was done is described. Then the results from the state-of-the-art algorithm and the neural network are compared followed up by a discussion.

1.1. Deep neural networks

A neural network is an algorithm that tries to map the functioning of our brain by mathematical simplifications. The brain's neurons are represented by activation functions with weights and biases. These artificial neurons are arranged in layers and are linked together layer-wise (compare fig. 1). There are 3 different types of layers, the input, the hidden and the output layers. The first (input) and the last (output) layer symbolize accessible values, while all layers in between are called hidden layers and usually have no real physical meaning. If there is more than one hidden layer, we speak about deep learning. With such a network structure it is possible to map any input vector to every output vector. It is mathematically proven that a neural network with more than one hidden layer can act as a universal function approximator (Lu et al., 2017). The mapping function must be learned and is generated during the training process.
To train a NN, first the weights have to be initialized - usually this happens randomly with some conditions to the random distribution. During the training process, data is fed into the network via the input layer, which then propagates through the network layer per layer until they reach the output layer. There, the initially random outputs, are compared with the target values. The weights and biases of the neurons are then adjusted depending on the deviation between the output and the target value using the so-called backpropagation, a stochastic gradient method. This way the information about the relationship between input and output is stored within the network. Modern

*Corresponding author
Email address: max.pargmann@dlr.de (Max Pargmann)

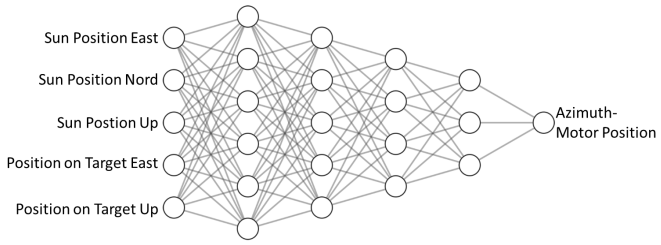


Figure 1: Schematic drawing of the full dense neural network (FNN) used in this paper. For sake of simplicity only one output is used for regression. The target position east and up are measured while calibration, but can later, after training, be used to control the focal point position

NNs received remarkable results in many different areas, from classification (Xie et al., 2020), over regression (Feng et al., 2019), to imaging tasks (Pidhorskyi et al., 2020). By using modern deep network structures and the availability of a sufficiently large dataset, even the most complicated tasks can be performed. Thanks to the ability of neural networks to adapt from similar tasks to the current problem (transfer learning), nowadays it is not even necessary to have a large dataset of the actual problem, if an appropriate pretraining has taken place (Feng et al., 2019),(Noguchi and Harada, 2019).

On the other hand deep full dense networks (FNNs) are still very rarely used, because they are hard to train due to vanishing gradients and converging to poor local minimums in the training process (LeCun, 2015). Nevertheless, these networks have also made astonishing progress in recent years. With the use of SNNs (Klambauer et al., 2017) they have become deeper than ever before and with a stacked autoencoder (SAE) unsupervised pretraining (Bengio et al., 2007) they have achieved remarkable results with only a few data points (Feng et al., 2019).

1.2. State-of-the-Art

The Camera-Target Method (Stone (1986)-method) is one of the oldest and up for today the most widely used calibration method in commercial solar tower power plants. For the calibration process, the focal spot of each heliostat is moved individually from the receiver to a Lambertian target, which is located below the receiver. A camera then takes a picture of the focal spot and compares its centroid of area with a reference position, after which all information is stored (compare fig. 2). Using an underlying function template, in most cases an error-based geometry model can be determined by regression using e.g. a Newton's Method.

The function template underlying the regression algorithm can include multiple error sources, from torsion, displacement or gear ratio to more complex errors like the influence of local wind speed or mechanical deformation on different angles. The more degrees of freedom the function template has, the more accurately it can represent reality, but also carries the risk, if the database is small, of converging towards a local minimum and missing the global optimum. There are plenty of other calibration methods (Sattler et al., 2020) but for every new one the Stone-Method is the baseline model, when it comes to accuracy.

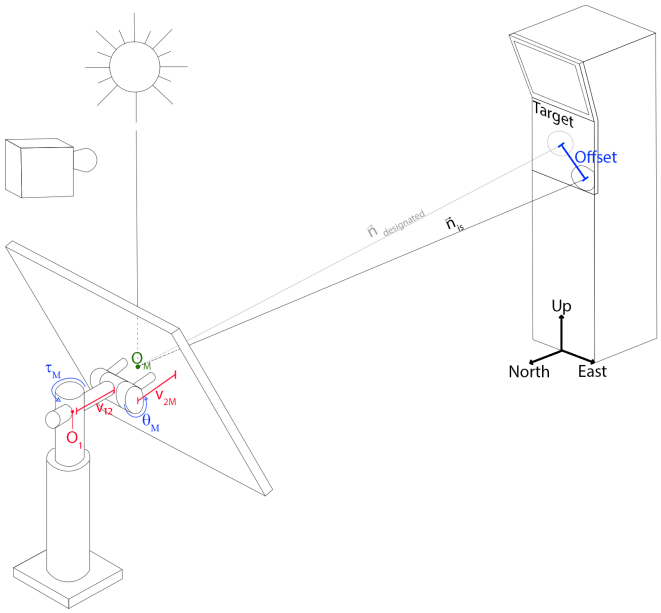


Figure 2: Schematic drawing of the Stone Method. The variables in blue are measured in every calibration (including the sun position). The red ones are measured once for every heliostat and the green is calculated. The shown method is described in 2.1 and variables are explained in more detail in chapter Appendix A of the appendix

In this paper it is shown, with an example of the Stone method that it is possible to improve regression based calibration algorithms with the support of neural networks without the need to replace existing hardware.

2. Research Setting and Methods

2.1. Calibration at the Solar Tower Jülich

The data used in this paper was gathered at the solar tower in Jülich where the Stone method is used to calibrate the heliostats. Using the focal spot position, the sun vector, the heliostat position and a few specific heliostat parameters which are measured once (axis offset, basic orientation etc.), the current alignment is calculated using a geometry model.

This model includes, besides the controllable alignment parameters (θ_M, τ_M), error parameters which have to be determined by regression. In Jülich eight of these errors $(\alpha, \beta, \gamma, \delta, \theta_k, \tau_k, \mathrm{GR1}, \mathrm{GR2})$ are allocated to every heliostat. For a detailed overview of all regression parameters see Appendix A. The Levenberg-Marquardt (LM) algorithm is used to minimize the function:

$$F = \min_{\alpha,\beta,\gamma,\delta,\theta_k,\tau_k,\mathrm{GR1},\mathrm{GR2}} \sum_{i=1}^{N} \arccos(\vec{n}_{\mathrm{is},i} \cdot \vec{n}_{\mathrm{model},i}). \quad (1)$$

The function template sums over all angular deviations for every measurement point i. $\vec{n}_{\text{model}}$ is the heliostat surface normal, which points towards the receiver. Thereby both rotation axes are positioned in the suspension point $\vec{O}_1 = \vec{O}_M$ of the heliostat. It is described by:

$$\begin{aligned}\vec{n}_{\text{model}} = R_{01}(\alpha,\beta)\cdot R_\theta(\frac{\theta_M}{GR1}+\theta_k)\\ \cdot R_{12}(\gamma)\cdot R_\tau(\frac{\tau_M}{GR2}+\tau_k)\cdot R_{2M}(\delta)\\ \cdot \vec{n}_{\text{ba}}.\end{aligned} \tag{2}$$

With $\vec{n}_{\text{ba}}$ the basis alignment.
$\vec{n}_{\text{is}}$ is the actual alignment including a suspension point shift and is calculated with:

$$\vec{n}_{\text{is}} = \left(\vec{CA} - \vec{O}_M\right)/\text{Norm}. \tag{3}$$

$\vec{CA}$ is the centroid of area of the focal point and $\vec{O}_M$, which is defined as:

$$\begin{aligned}\vec{O}_M = \vec{O}_1\\ +R_{01}(\alpha,\beta)\cdot R_\theta(\theta_M/GR1+\theta_k)\\ \cdot\left(\vec{V}_{12}+R_{12}(\gamma)\cdot R_\tau(\tau_M/GR2+\tau_k)\cdot\vec{V}_{2M}\right).\end{aligned} \tag{4}$$

$\vec{V}_{12}$ and $\vec{V}_{2M}$ are suspension point displacements relative to the first and second movement axis (compare Fig. 2). Both are measured once and are constants individual for every heliostat.
All R functions are rotation matrices, where α, β, γ, δ are the angles of the rotation and tilting errors, GR1 and GR2 are gear ratios and τ_k and θ_k define axis offsets. All these parameters are calculated by the LM Algorithm. θ_M and τ_M are the current motor positions of the heliostat. They are measured in real time and because the heliostats in Jülich use stepping motors they are measured as discrete steps which can be converted from motor positions to angles and vice versa.
After calculation, the error parameters are stored, if the prediction (test-)accuracy is better than any result before. With the defined geometry model, the required orientation of the heliostat to hit the receiver can be calculated iteratively by computing the bisector between n_{is} and the sun vector.

2.2. Potential of applying neural networks for heliostat field calibration

Even if the heliostat calibration is an analytically comprehensible task, taking all internal and external influencing factors into account it's a complex system for which the use of neural networks is suitable. NNs for alignment detection of heliostats in solar tower systems have already been introduced by Carballo et al. (2018) and have shown the upcoming possibilities. But up for today the applications for calibration tasks are still limited, usually require new hardware and (labeled) datasets which has to be created specifically for the NNs. The potential of applying neural networks is there, but the main challange of the methodology is clearly the size and the imposed conditions on the required dataset.
Besides, since every heliostat contains its own individual errors and e.g. using the Stone method there is more a lack of data rather than a surplus.

With transfer learning it is possible to overcome this disadvantage. Before the training on the real problem a pretraining, for example with simulated data is done. In the real training process, instead of assigning random values to each node at the beginning of the training, the network is initialized with the node values calculated in the pretraining, which are already close to the global optimum.
For the heliostat calibration task, the function template, calculated by the LM algorithm can generate data, which is close to real measured data, but with a lack of complexity, which are optimal conditions for an appropriate pretraining.

2.3. Method

For the calibration task a network structure was chosen to simplify the task as much as possible (only one output) but at the same time to collect as much data as possible without any constraints and to allow free heliostat control (two additional inputs). Therefore a network with 6 layers

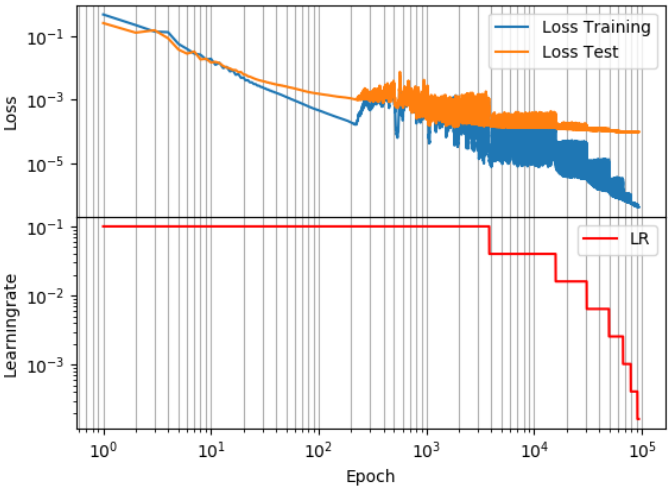


Figure 3: An example learning curve in the pretrain process using 4000 data points simulated by the LM-template. The loss shown is the mean squared error (MAE). The MAE at the end of the pretraining varies between 2×10^{-5} and 9×10^{-5} depending on the start conditions. Every loss in this range is acceptable for the following train step.

5-(6-5-4-3)-1 is used (compare fig. 1). For the activation function SELUs (Klambauer et al., 2017) are chosen, except for the output layer, which is linear.
To regularize the prediction error the L1-Huber loss (Huber, 1992) is used. To enhance the learning process an

adaptive learning rate, a factor which controls the amount of change inside the nodes, is selected. The used learning rate becomes smaller every time the test loss did not reduce over a certain amount of epochs. An epoch refers to one cycle during which the network has seen all training data once. This is enhanced by using AdaMax (Kingma and Ba, 2014) as an optimizer.

The training process is divided into pretraining and actual training. The pretraining is done using data generated by the LM algorithm and initialization recommended by Klambauer et al. (2017). A learning curve of the pretraining is shown in fig. 3. On the X-axis the training epochs are shown. In the upper graph the y-axis refers to the mean absolute error (MAE) loss function. Even if the Huber loss is used for training, the MAE gives a better indication of the deviation between real and predicted values. The dataset is splited into a train and a test set (this is described in more detail in chapter 2.4). Only the loss of the training set influences how the nodes of the network are updated. It is easy to see, why the train-test split is necessary. Although the training loss never stops falling and easily reaches accuracies up to 99.9%, the test accuracy stagnate at some point.

In the lower graph the adaptive learning rate is plotted. It starts with a value of lr = 0.1 and is reduced over time. The learning rate while training is considerably smaller (lr $= 1 \times 10^{-5}$) in order to preserve the pretraining progress. Alternatively, the first layers could be completely excluded from the training process, this is called freezing, but better results could be achieved with the above mentioned procedure.

As the inputs, the sun position, which is relevant for the task, and the measured focal point position on the target was chosen. The latter includes two big benefits. On the one hand we can use all collected data, because we don't have to care about the focal points position, which otherwise would first have to be moved to a specific reference point. On the other hand, after training these nodes can be used as a control option to freely move the focal point to different positions on the receiver/target.

The output is either the measured azimuth or the elevation alignment of the heliostat given in motorpositions. Without loss of generality only the horizontally mounted motor for the azimuth movement will be considered in the following.

2.4. Workflow

The whole workflow is shown in Fig. 4. In the first step a training set size is chosen for all data gathered at the solar tower. Then an equally large test-set is declared (choosing a train-test split of 50:50). Training and test-set are separated in time, because this gives a better overview how good the network will predict future data. For the collected dataset including 500 measuring points (explained more deeply in chapter 3.1), this means that depending on the selected training set size, a third set, the validation set, may remain in addition to the test-set.This third part

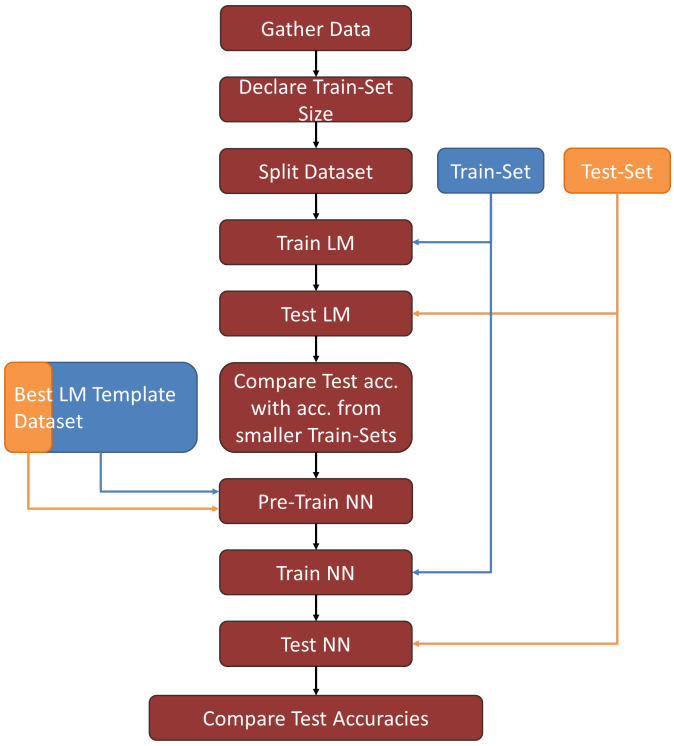


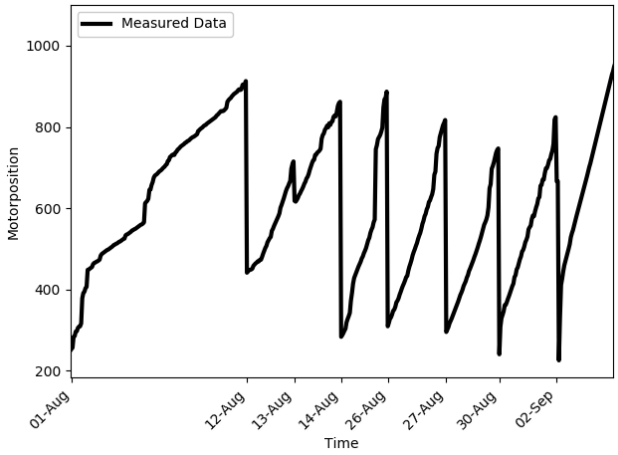
Figure 4: Sketch of the workflow. For an application at the solar tower no training size would be declared in the second step, but all usable data would be split into training-, test-set. The fine tuning, e.g. choosing different network structures are not shown inside the sketch

is added to the test-set independently after the training. If the training set becomes so large, that a 50:50 ratio is not possible anymore, the test-set percentage is reduced, but the validation set remains at a constant value of 5% of the complete dataset.

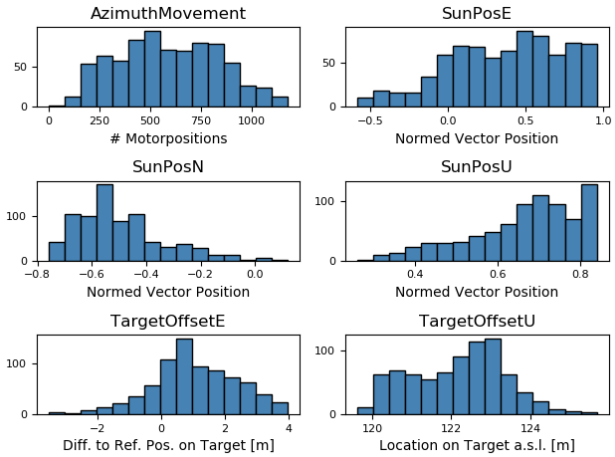
Then the LM regression algorithm is trained with the training set. After finishing, the newly calculated geometry model from the function template is tested on the test and validation set and its accuracy is evaluated.

If the prediction accuracy is better then any accuracy before, the new parameters get stored. The geometry model with the highest test accuracy is now used to generate pretraining data for the neural network. For this 4000 randomly distributed but real sun positions between August 1st and October 1st are generated. This dataset is then split into a 90:10 training-test-set which the network is then pretrained with. As soon as the test loss stops falling, the actual training process begins. The network is now trained/tested with the same training test data split as the Levenberg-Marquardt algorithm before. Instead of a random initialization, the previously calculated weights and biases of the pretraining are used.

The whole training process takes less than 30 minutes on an Intel i7 processor without a graphic card. The process can be accelerated either by using a faster pc (including a GPU), or e.g. using a less conservative loss function, like



a) All measured azimuth motor positions of the example heliostat in the middle of the solar field, 57.2m east, 243.5m north of the base of the solar tower.



b) All input and output parameters of the network. the vertical target reference position (TargetOffsetU) is at an altitude of 123.075m above sea level

Figure 5: Different visualizations of the collected data at the solar tower Jülich.

the cyclical learning rate(Smith, 2015). At the end the test accuracies are compared to each other.

3. Case Study

3.1. Dataset assembling

To test the different calibration algorithms, a dataset of approximately 500 measurement points was used, which were recorded within 8 days, whereby a large part of the data falls on the first measurement day. The data was recorded at different times between 9:00 and 15:30. The amount of data varies between 1 and nearly 200 measurement points per day. The heliostat alignment is given in motor positions. There were no special requirements when creating the dataset. The histogram (Fig. 5b) shows the values required for calibration. The values *TargetOffsetU*p and *TargetOffsetE* are showing the errors of the respective heliostat. The focal point of this particular heliostat is often located in the center of the target but tends to drift eastwards and downwards. It is rarely above or west of the target center. We will come back to this later in the discussion.

Because the amount of data is more relevant for the upcoming discussion, the x-axis of the upcoming Graphs will show the dataset size used for training instead of the dates they were recorded. Nevertheless the shown data will be the same and in the same order as in Fig. 5a.

3.2. Levenberg-Marquardt Results

At the Jülich solar tower, the Levenberg-Marquardt algorithm is used to determine the geometry model from the function template described in 2.1. However, in general any Newton- or Trust Region-method can be used here. The limiting factor of all these methods is the function template that has to be minimized. For high accuracy the template requires a high number of free parameters (in Jülich, the maximum is 8). More degrees of freedom can lead to a better overall result, but with few measurement points there is a risk of finding only a local optimum, which can significantly reduce test accuracy. Figure 6 shows the

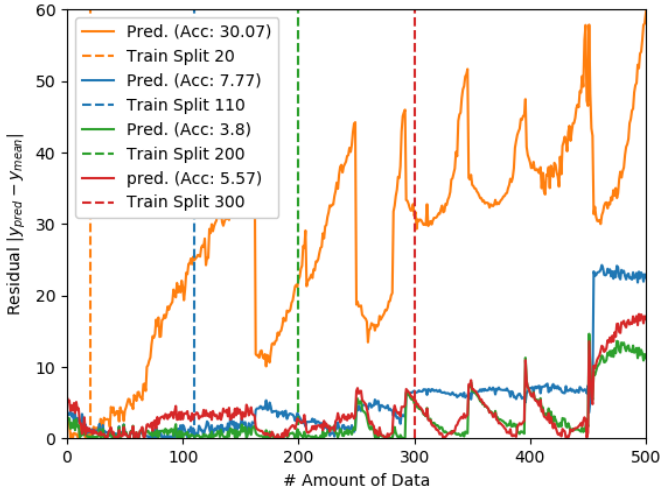


Figure 6: Different predictions of the LM algorithm depending on the amount of data used for training. The different training-sets are located on the left of the dotted lines.

predictions of the LM-algorithm for different training data sizes. For the smallest training set the training accuracy is high, but the algorithm has clearly converged to a local optimum, which leads to a bad prediction accuracy. However, the accuracy improves continuously with increasing the amount of data up to about 200 data points.

Furthermore, the error remains at a roughly constant level. The peaks appearing in the red and green line and the plateau in the rear part of the graph either indicate that

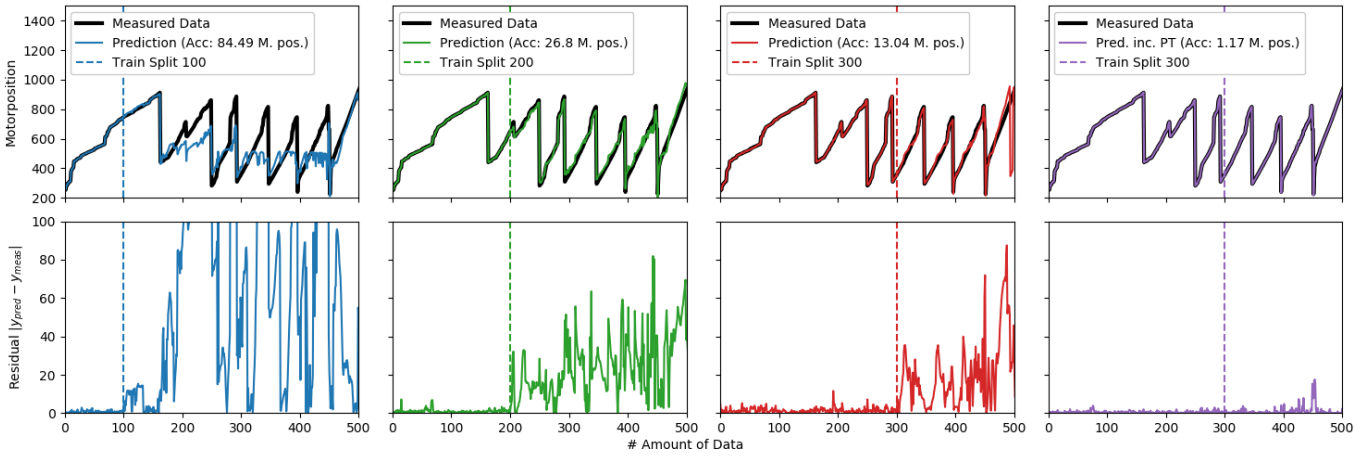


Figure 7: Different predictions of the NN algorithm depending on the amount of data used. Except for the far right figure, no pretraining has taken place. The pretraining graph was trained with the same data as the graph left to it (red line) but has a significantly higher accuracy.

the global optimum is not found yet or the function template cannot completely map the real heliostat due to missing modeling parameters, like orientation dependent deformation.

3.3. Neural Network Results

In Fig. 7 the absolute and relative prediction results of the trained neural networks, with and without pretraining, are shown. The difference between Training and Test dataset is clearly visible. As already seen in Fig. 3, the training accuracy reaches a far smaller loss, but this does not necessarily indicate a high test accuracy.
In the graph on the very left (blue line), the network was trained with only 100 training data without pretraining. These correspond exclusively to measurements of the first half of a day. This leads to the fact that on later days, forecasts for later day times are correspondingly worse. This can be recognized by the missing peaks of the prediction. The lack of extreme values in the dataset is therefore a significant factor in training.
For training sets containing 200 and 300 measuring points, the network has already seen two full measurement days during training. Here, it manages to predict a similar curve as measured in reality, at least in the absolute view. In the relative view, however, it is clear that a competitive accuracy has not yet been reached.
The best prediction accuracy using the same amount of data as the graph to its left (red line)but including pretraining is shown on the very right (violet line). The dataset for the pretraining was created by the geometry model calculated from the LM by using 200 data points (Fig. 6, green line), which was the geometry model with the highest prediction accuracy. Including pretraining the mean prediction accuracy is close to 5 times better than the LM results with 300 data points and 3 times better than the best LM result archived with the function template used in Jülich.

3.4. Comparison

After all algorithms have been trained, the errors of the different algorithms can be compared. Since our dataset only contains 500 data points, no more than 300 for training were ever used, as the test dataset would otherwise have become too small to make reliable statements. Fig. 8 shows the prediction accuracy of the conventional method and the neural networks, with and without pretraining, depending on the amount of data.
The behavior of Levenberg-Marquardt described in 3.2 can be seen more clearly here. The accuracy is increasing up to about 200 data points, but then it stagnates at about the same or slightly worse level.
The prediction deviation of the neural network without pretraining decreases rapidly over the whole test area, but never reach a competitive accuracy within the available data set.
The highest accuracy is achieved starting from 300 data points by the neural network with pretraining. As mentioned before, the function template calculated by the LM Algorithm at 200 data points was used to pretrain the networks also for higher amount of data.

4. Discussion

By extrapolation of the orange curve in Fig. 8 it is possible to get an idea of the possibilities of using neural networks. Although this is only an experiment of thoughts, because the output of a NN is not predictable, the orange line can be extrapolated and will become competitive with more data. NNs already have shown impressive results in other, more complex tasks when there was a sufficient database available. So, this should be the case here too. It becomes particularly likely if the results of the network with pretraining are considered.
The pretraining enhances the method in many ways. On the one hand it reduces the required dataset considerably,

resulting in an improvement of the prediction accuracy already at 300 data points. On the other hand it can compensate missing extreme values inside the real dataset.
Compared to the state-of-the-art algorithm it has to be mentioned that on the one hand the information about the physical error parameters represented in the geometry model is lost through the use of NNs, because the parameters are not calculated anymore. On the other hand the calibration can benefit from the missing limitation of a prescribed function template through the increased complexity. Especially considering that the error parameters are not known to be used in power plant operation anyway.
In addition, if the decision is made to include complex properties such as local wind speed into the calibration, no complicated differential equations or even simulations are necessary, but the simple addition of further input nodes and a new training are sufficient.
However, unlike the classical approach, there is no clearly traceable control function after completion of the training with a neural network. Although the networks are capable of abstraction to a certain extent, they work best for values between known extrema included in the (pre-)dataset. Aiming to a designated position that goes beyond the min-max values contained in the training set, or a position that was rarely used could lead to higher errors. The heliostat examined here, for example, tends to drift to the lower east of the target center as described in chapter 3.1. Targeting a top west position could cause a larger error. Therefore, it is recommended to include extreme values (e.g. the edges of the target) already in the dataset.
A similar situation applies to the other input parameters. The sun positions can become more extreme depending on the season. For measurements taken in this work this is not noticeable, because the days became shorter during this time. A different picture may appear from spring to summer.
Although the dataset can be improved by specific early morning and late evening measurements, not every heliostat can be measured on the (ideal) longest day of the year, here only a suitable pretraining can help.
One also has to keep in mind that the network was trained on data pointing at the target. When the receiver is to be targeted, in best case the network should already have adapted the real tracking function and there is no additional error or only a vertical offset should be needed to correct this issue. In the worst case another mapping function is needed. This still has to be tested.
In general, all points in Fig. 8 are located within a convergence band which allows better and worse results for both the Levenberg Marquardt algorithm and the neural network. This band depends on the starting conditions, the structure of the network, the function template, the dataset, the loss, the activation function and many other factors and is therefore too complex to give an estimate of the statistic bandwidth.
All these internal parameters are most likely not fully optimized yet. Nevertheless this is similar to the situation at the solar tower, where these parameters also can't be optimized for each individual heliostat in the field. More likely, only a few of these parameters will be optimized (e.g. with bayesian parameter search) for a given heliostat or a group of heliostats and then are adopted to the rest of the field. Therefore, these results give a good orientation how the algorithm will behave in use at a big heliostat field.

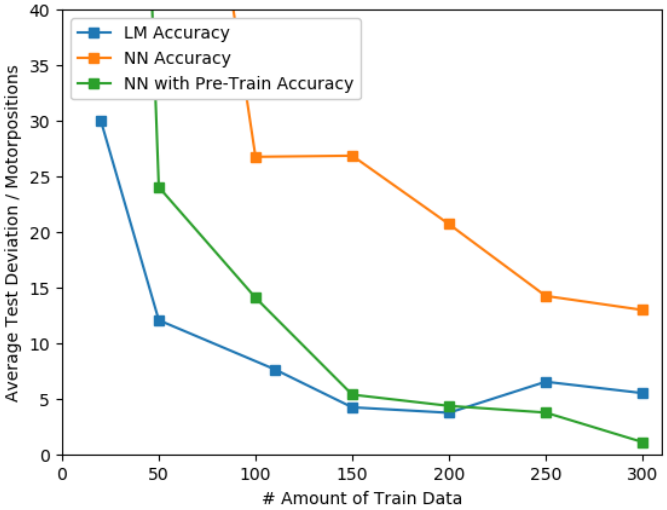


Figure 8: Summary over all tested algorithms. The values can vary considerably for all algorithms used, as they are located on a complex convergence band. However, it was tried to optimize all algorithms equally for the given data set

5. Conclusion

Based on the results shown in this paper, neural networks are most likely the best choice for heliostat calibration, if a large dataset is available. Though this is not the case for calibration techniques, where the measurement takes many seconds to minutes or each heliostat has to be moved individually, neural networks can still be a reasonable choice if a suitable pretraining has taken place. With an example of the Stone-Method it was found, that the function template underlying the used regression algorithm can be used to create a database large enough to pretrain neural networks. By the help of this pretraining it was possible to increase the prediction accuracy of one heliostat at the solar tower Jülich by a factor of 3 compared to the best geometry model results calculated by the classical approach, already starting with a training dataset containing less than 300 measurement points.
Furthermore, first estimations could be made regarding an optimization of the training dataset with respect to the long-term use of neural networks at the solar tower.

6. Outlook

With the classical calibration approach each heliostat has to be calibrated very regularly. the aim is to extend the time between two calibrations as much as possible, while keeping the tracking accuracy high. To ensure this for the NN approach, the next step is to run a long term test at the solar tower in Jülich to validate its prediction accuracy throughout the year. If it is not maintained the training/pretraining has to be adapted to ensure that this is the case.

In connection with this the pretraining should be studied more, in context of included extreme values or choosing another algorithm to create the artificial dataset e.g. using a grid instead of a random distribution. At least, for an unsupervised application at a complete solar tower field a mathematical network optimization, like Google's Morph Net (Gordon et al., 2018) or a Bayesian optimization, have to be implemented.

References

A. Pfahl, J. Coventry, M. Röger, F. Wolfertstetter, J. F. Vásquez-Arango, F. Gross, M. Arjomandi, P. Schwarzbözl, M. Geiger, P. Liedke, Progress in heliostat development, Solar Energy 152 (2017) 3–37.

Z. Lu, H. Pu, F. Wang, Z. Hu, L. Wang, The expressive power of neural networks: A view from the width, in: Advances in neural information processing systems, 2017, pp. 6231–6239.

Q. Xie, M.-T. Luong, E. Hovy, Q. V. Le, Self-training with noisy student improves imagenet classification, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 10687–10698.

S. Feng, H. Zhou, H. Dong, Using deep neural network with small dataset to predict material defects, Materials & Design 162 (2019) 300–310.

S. Pidhorskyi, D. Adjeroh, G. Doretto, Adversarial latent autoencoders, arXiv:2004.04467v1 (2020). arXiv:2004.04467v1.

A. Noguchi, T. Harada, Image generation from small datasets via batch statistics adaptation (2019). arXiv:1904.01774v4.

B. Y. . H. G. LeCun, Y., Deep learning, Nature 521 (2015) 436–444.

G. Klambauer, T. Unterthiner, A. Mayr, S. Hochreiter, Self-normalizing neural networks, CoRR abs/1706.02515 (2017). URL: http://arxiv.org/abs/1706.02515. arXiv:1706.02515.

Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, Greedy layer-wise training of deep networks, in: Advances in neural information processing systems, 2007, pp. 153–160.

K. W. Stone, Automatic heliostat track alignment method, 1986. US Patent 4,564,275.

J. C. Sattler, M. Röger, P. Schwarzbözl, R. Buck, A. Macke, C. Raeder, J. Göttsche, Review of heliostat calibration and tracking control methods, Solar Energy 207 (2020) 110–132.

J. A. Carballo, J. Bonilla, M. Berenguel, J. Fernández-Reche, G. García, New approach for solar tracking systems based on computer vision, low cost hardware and deep learning (2018). doi:10.1016/j.renene.2018.08.101. arXiv:1809.07048v1.

P. J. Huber, Robust estimation of a location parameter, in: Breakthroughs in statistics, Springer, 1992, pp. 492–518.

D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).

L. N. Smith, Cyclical learning rates for training neural networks (2015). arXiv:1506.01186v6.

A. Gordon, E. Eban, O. Nachum, B. Chen, H. Wu, T.-J. Yang, E. Choi, Morphnet: Fast & simple resource-constrained structure learning of deep networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 1586–1595.

Appendix A. LM-parameters

In table A.1 all used LM parameters are listed. The function template used at the solar tower in Jülich assumes a model heliostat with 2 axes. Thereby each axis can move the suspension point ($\vec{O}_1$) by a constant vector ($\vec{v}_{12}$,$\vec{v}_{2M}$). Without rotations the suspension point (the mirror center) would be: $\vec{O}_M = \vec{O}_1 + \vec{v}_{12} + \vec{v}_{2M}$. Each rotation R is described by a rotation matrix depending on the angles $\alpha, \beta, \delta, \gamma, \tau_M, \theta_M$ and is acting on either the first or on both translation vectors. Where $\alpha, \beta, \delta, \gamma$ are constant angles of misalignment (which have to be determined by the regression algorithm) and τ_M, θ_M are the actual controllable alignments of the heliostat, measured in motor positions. Beside these rotational misalignments, for each axis the gear ratio ($GR1, GR2$) and an offset (τ_k, θ_k) are considered for the control parameters τ_M and θ_M

Symbol	Explanation	Degrees of Freedom	Dependencies	Explanation
R_{01}	Alignmet 1. axis (incl. tilting)	2 (rotation)	α	Rotational displacement
			β	Rotational displacement
R_θ	Rotation 1. axis	1 (rotation) +1 (gear ratio)	θ_M	Tracking angle azimuth
			θ_k	Axis offset
			GR1	Gear ratio
$\vec{v}_{12}$	Distance between axes	1 (translation)	-	-
R_{12}	Axis tilt	1 (rotation)	γ	Rotational displacement
R_τ	Rotation 2. axis	1 (rotation) +1 (gear ratio)	τ_M	Tracking angle elevation
			τ_k	Axis offset
			GR2	Gear ratio
$\vec{v}_{2M}$	Displacement between 2. axis and mirror center	1 (translation)	-	-
R_{2M}	Mirror tilt	1 (rotation)	δ	Rotational displacement
$\vec{O}_1$	Heliostat suspension point (without translation)	3 (translation)	-	-
$\vec{O}_M$	Heliostat suspension point (mirror center)	3 (translation)	All above	-

Table A.1: Matrices and vectors used for the function template underlying the Levenenberg-Marquardt regression algorithm.