Journal of Machine Engineering, 2021, Vol. 21, No. 2, 47–59
ISSN 1895-7595 (Print) ISSN 2391-8071 (Online)

Received: *30 November 2020* / Accepted: *14 March 2021* / Published online: *10 June 2021*

artificial neural network,
robot calibration,
hyperparameter optimization

Eckart UHLMANN[1,2]
Mitchel POLTE[1,2]
Julian BLUMBERG[1*]
Zhoulong LI[1,3]
Adrian KRAFT[1]

HYPERPARAMETER OPTIMIZATION OF ARTIFICIAL NEURAL NETWORKS TO IMPROVE THE POSITIONAL ACCURACY OF INDUSTRIAL ROBOTS

Due to the rising demand for individualized product specifications and short product innovation cycles, industrial robots gain increasing attention for machining operations as milling and forming. Limitations in their absolute positional accuracy are addressed by enhanced modelling and calibration techniques. However, the resulting absolute positional accuracy stays in a range still not feasible for general purpose milling and forming tolerances. Improvements of the model accuracy demand complex, often not accessible system knowledge on the expense of realtime capability. This article presents a new approach using artificial neural networks to enhance positional accuracy of industrial robots. A hyperparameter optimization is applied, to overcome the downside of choosing an appropriate artificial neural network structure and training strategy in a trial and error procedure. The effectiveness of the method is validated with a heavy-duty industrial robot. It is demonstrated that artificial neural networks with suitable hyperparameters outperform a kinematic model with calibrated geometric parameters.

1. INTRODUCTION

Industrial robots (IR) are widely used in industry for automation of pick-and-place as well as assembly tasks. In order to widen the application range to machining processes as milling and forming, the deficiency of IR regarding their low absolute positional accuracy T needs to be improved [1]. Such approaches mainly focus on modelling the geometric and non-geometric error sources of the serial chain, as joint angle offsets $\Delta\boldsymbol{\theta}$, angle and link length errors as well as joint and link stiffness [2–5]. The model parameters are in general identified experimentally. Uncertainty remains due to phenomena as e.g. gear backlash, friction and nonlinearities. Even though the modelling techniques are continuously improving, all

[1] Institute for Machine Tools and Factory Management IWF, TU Berlin, Germany
[2] Institute for Production Systems and Design Technology IPK, Fraunhofer, Germany
[3] School of Mechanical Engineering, Shanghai Jiao Tong University, China
[*] E-mail: blumberg@iwf.tu-berlin
https://doi.org/10.36897/jme/134275

the models capturing the physics behaviour of IR are still not feasible to meet general purpose milling and forming tolerances [6]. As a further improvement of the positional accuracy T of IR is so far limited by the model accuracy, the parameter identification method or the real time capability, recent research includes data-based modelling techniques. The IR kinematics as well as the error sources are partially or fully captured by algorithms, which relate the joint angle vector $\boldsymbol{\theta}$ to the pose error vector $\boldsymbol{e}$ or vice versa. Due to their capability of approximating arbitrary nonlinear relations as well as their simple structure, flexibility and learning ability, especially artificial neural networks (ANN) gain increasing attention in a wide range of applications [7]. The integration of ANN models in the process of accuracy improvement of IR is schematically shown in Fig. 1.

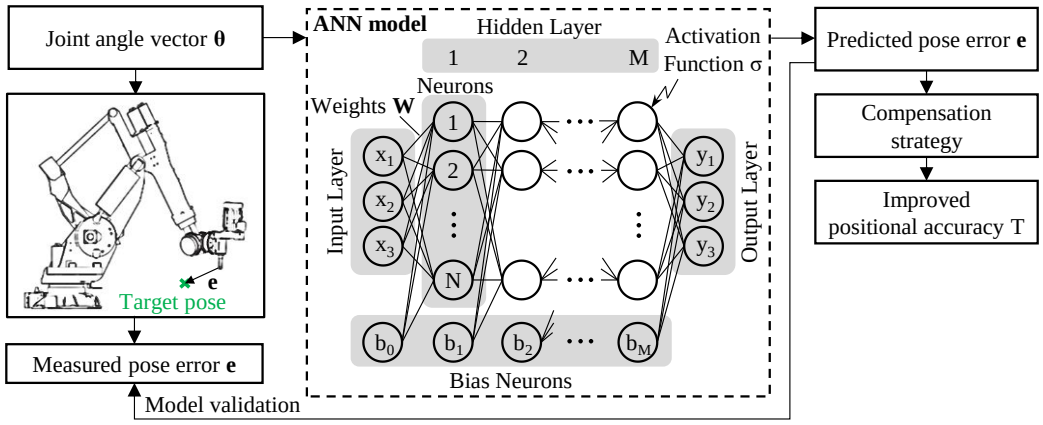

Fig. 1. Improvement of the positional accuracy T of IR by means of an ANN model

The generic ANN model with fully connected layers maps the input vector $\boldsymbol{x}$ to the output vector $\boldsymbol{y}$ by a composition of nonlinear functions g in the form [8]:

$$\boldsymbol{y} = f(\boldsymbol{x}) = g^{(M)} \circ g^{(M\text{-}1)} \circ \ldots \circ g^{(1)}(\boldsymbol{x}), \text{ with } g(\boldsymbol{x}) = \sigma(\boldsymbol{W} \cdot \boldsymbol{x} + \boldsymbol{b}) \tag{1}$$

In Eq. 1 the parameter M denotes the number of hidden layers, σ the activation function, $\boldsymbol{W}$ the weight matrix and $\boldsymbol{b}$ the bias. The weight matrices $\boldsymbol{W_i}$ and the biases b_i are adjusted in the training phase by an algorithm called backpropagation. As backpropagation computes the gradient of a loss function with respect to the ANN parameters, the activation function must be differentiable. Commonly used activation functions are RELU (rectified linear unit) and ELU (exponential linear unit). For a broad and concise overview of ANN it is referred to Fan et al. [7].

ANN models are mainly used for classification problems as e.g. face recognition. Only a few researches deal with the application of ANN models to the regression problem in robot calibration. Angelidis and Vosniakos [8] used three ANNs to estimate the three components of the relative position error vector $\boldsymbol{e_p}$ on a circular path. As the ANN model is trained for a specific task, it is not applicable to improve the positional accuracy T in a general manner. Nguyen et al. [9] used an ANN model to relate the joint angle vector $\boldsymbol{\theta}$ to the position error vector $\boldsymbol{e_p}$ of an IR. They used the first three joint angles θ_1, θ_2 and θ_3 of a six axes IR as input parameters to capture the non-geometric error sources after geometric calibration. Recently,

Nguyen et al. [6] adopted this method and implemented joint stiffness parameters in the physics model of the IR. The ANN model is composed of six input and three output nodes corresponding to the six joint angles θ and the three error components after geometric and joint stiffness calibration, respectively. In both works they used a fixed and a priori defined ANN architecture regarding number of hidden layers M, number of neurons per hidden layer N, activation function and learning rate l. Aoyagi et al. [10] used a similar but inverted approach to compensate the remaining error after geometric calibration for a seven axes robot. The three-dimensional position data served as the input to calculate the joint angle compensation values $\Delta\boldsymbol{\theta}$. They did not consider the impact of the choice of the ANN architecture upon the resulting positional accuracy T. Zhu et al. [11] used a radial basis function (RBF) neural network with a Gauss function as the activation function. They used the three components of simulated position data as the input to calculate the position error vector $\boldsymbol{e_p}$. The simulated training data is free from noise and does only contain geometric error sources. In order to bypass the difficulty of choosing an appropriate learning rate l for ANN training, Cai et al. [12] and Yua et al. [13] proposed to use an extreme learning machine (ELM) algorithm with one hidden layer and a sum function as the a priori defined activation function. They obtained an appropriate number of nodes N in the hidden layer by a trial and error procedure. However, the conclusions are highly dependent on the system and training data investigated and therefore not directly applicable to other scopes. Le and Kang [14] proposed to use an invasive weed optimization algorithm instead of backpropagation to determine the weight matrices $\boldsymbol{W_i}$ and the biases b_i. They as well did not consider the impact of the fixed and a priori chosen number of hidden layers M, number of neurons N and activation function upon the resulting positional accuracy T.

The proposed models show promising results for pose error compensation of IR in terms of ANN. However, the ANN models are not investigated regarding model structure, activation function and training strategy, even though these factors can significantly impact the ANN model performance. In this paper, the hyperparameters of an ANN model are optimized algorithm-based in terms of a neural architecture search (NAS), in order to find the best suitable ANN architecture to predict the pose error vector $\boldsymbol{e}$ of IR [15]. These hyperparameters include the number of hidden layers M, number of neurons per hidden layer N, activation function, bias, batch size B, learning rate l and dropout probability d. Further, the dependency upon the number of training data, the scope of validity as well as the interpolation and extrapolation capabilities of the proposed ANN models are investigated. With the proposed method the downside of choosing an appropriate ANN architecture and training strategy in a trial and error procedure can be overcome. The heavy-duty IR Fanuc M-900iB/700 by the company Fanuc K.K., Oshino, Japan, serves as a demonstrator throughout the present research.

2. ARTIFICIAL NEURAL NETWORK MODEL

In order to achieve high accuracy, the ANN architecture should be chosen depending on the characteristics of the relation to be approximated. Due to the vast number of hyper-parameters the ANN architecture is often chosen in a trial and error procedure and therefore

based on the experience of the user. To overcome this downside Elsken et al. [15] proposed the method NAS, which is separated in the following three steps:

- definition of a hyperparameter search space,
- a search strategy / hyperparameter optimization, as well as,
- a performance estimation strategy.

The general and fixed ANN architecture throughout this paper is composed of six input nodes corresponding to six joint angles θ_i of the IR as well as three output nodes corresponding to the absolute position or orientation (error), respectively. By training two distinct ANN models, which output the position and orientation (error), respectively, the problem of computing a loss for different units can be bypassed. All layers are fully connected. Each hidden layer consists of a nonlinear activation function followed by a dropout module. In accordance with Akiba et al. [16], we denote a "study" as a hyperparameter optimization with a fixed search space as well as a "trial" as an ANN model with a fixed set of hyperparameters sampled from the search space.

2.1. HYPERPARAMETER SEARCH SPACE

It is assumed that the user of the ANN is unexperienced regarding the ANN hyperparameters and their influence on the ANN performance. The lower and upper bounds of the numeric hyperparameters are weakly based on preliminary tests as well as related literature. The hyperparameter search space is summarized in Table 1.

Table 1. Hyperparameter search space

Hyperparameter	Sampling distribution	Bounds / parameter set
Number of hidden layers *M*	0	[1, 21]
Number of neurons per hidden layer *N*	Uniformly	[5, 100]
Activation function	-	["ReLU", "ELU"]
Bias	-	["True", "False"]
Batch size *B*	Uniformly	[2, 20]
Learning rate *l*	Log-uniformly	$[10^{-5}, 10^{-2}]$
Dropout probability *d*	Uniformly	[0.0, 0.5]

2.2. SEARCH STRATEGY / HYPERPARAMETER OPTIMIZATION

The search strategy should be computationally efficient with regard to the computing time required to find trials that minimize the score s_{trial} in a fixed hyperparameter search space. A possible implementation of NAS is based on a grid search method, which represents a full factorial investigation. With m hyperparameters and n_i discretization points of each hyperparameter a total number of $n = n_1 \cdot \ldots \cdot n_m$ trials need to be performed. The more hyperparameters m and discretization points n_i are chosen, the higher the confidence in finding an optimal set of hyperparameters. However, this search strategy becomes

computationally expensive with an increasing number of hyperparameters m and discretization points n_i. Therefore, it is not advisable to use a grid search for finding an optimal set of hyperparameters [17]. Instead, we propose to use a model-based search strategy. Bergstra et al. [18] introduced an algorithm called tree-structured parzen estimator (TPE) for optimizing hyperparameters of ANN models. The TPE is a sequential model-based optimization (SMBO) approach. SMBO methods fit a regression model based on the history of past trials in order to estimate the performance of hyperparameters and to choose promising hyperparameters for the following trials. Sequential means that it iterates between evaluating a new trial and sampling a new set of hyperparameters from the regression model. Promising hyperparameters are selected by repetitively sampling hyperparameters from the search space and keeping the set of hyperparameters, which maximizes a history-dependent criterion called "Expected Improvement" [18].

To further reduce the computational costs, it is common to stop the evaluation of unpromising trials, which perform significantly worse regarding the same training iteration. Such methods are called pruning, which aim at the allocation of computational resources (budget G) to more promising trials. Jamieson and Talwalkar [19] proposed a bandit-based pruning method called Successive Halving (SHA). The SHA method has an additional hyperparameter n_t, describing the number of trials to examine. With the SHA method a given budget G, e.g. number of total iterations, is allocated uniformly to a set of trials n_t for a predefined number of iterations. It compares iteratively the performance of each trial and discards the worst half of the set of trials n_t, until a single trial remains. It can be shown that using SHA is often significantly faster compared to a standard approach in identifying the best trials [19]. In practice, there is a trade-off between the available budget G and the set of trials n_t. In a possible scenario the validation loss of a trial may stagnate for some iterations or decrease slowly compared to other trials, even though the final validation loss at the end of the iteration is lower. In such cases it would be beneficial to use a fixed budget G with a reduced set of trials n_t. To overcome the difficulty of choosing a reasonable pair of budget G and set of trials n_t, we propose to use the pruning method introduced by Li et al. [20]. The so-called Hyperband Method starts multiple instances of SHA each with a different set of trials n_t for a fixed budget G.

2.3. PERFORMANCE ESTIMATION STRATEGY

In order to find an optimal set of hyperparameters, the trials are compared regarding a score s_{trial} describing the performance estimation. This is achieved by training the ANN model with a training dataset for a predefined number of epochs n_E and evaluating a loss function for a validation set. Consider one ANN output to be $\boldsymbol{o}_p \in \mathbb{R}^{3x1}$ and one validation data point to be $\boldsymbol{v}_p \in \mathbb{R}^{3x1}$. The loss function for one batch of the validation set is calculated by means of the mean square error (*MSE*) according to Eq. 2.

$$^{V}MSE = \frac{1}{3Q}\sum_{p=1}^{Q}\sum_{i=1}^{3}(o_{i,p} - v_{i,p})^2 \tag{2}$$

In Eq. 2 the parameter Q denotes the size of the validation set. The simplest method is to calculate the validation loss ${}^{V}MSE$ for a fixed validation set. However, to avoid overfitting and therefore achieve a higher generality of the trained ANN model with better performance on random and unknown input data, a cross validation (CV) is preferable. We use a five-fold CV, where the total training dataset is splitted randomly in five subsets of the same size. Four of the subsets are used to train and one to validate the ANN model. This process is repeated, in such way, that each subset is used once to validate the ANN model. The validation loss ${}^{V}MSE$ is calculated after each training epoch. For the score of a trial follows Eq. 3. In Eq. 3 the index j denotes the considered number of CV and n_E the total number of epochs. It is important to note that with Eq. 3 the mean minimum error of the validation dataset is used for ANN model selection. As the validation dataset is a subset of the training dataset, overfitting of the remaining and unknown test dataset can be avoided. Fig. 2 shows the proposed overall hyperparameter optimization scheme.

$$s_{trial} = \frac{1}{5}\sum_{j=1}^{5} \min(\{ {}^{V}MSE_{j,1}, \ldots, {}^{V}MSE_{j,n_E}\}) \tag{3}$$

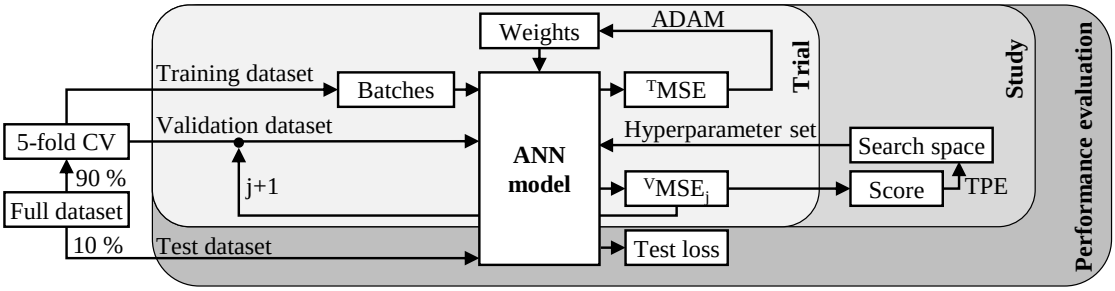

Fig. 2. Schematic overview of the hyperparameter optimization and performance evaluation process

2.4. IMPLEMENTATION

The ANN model is realized using Python 3.8 and the open source library PyTorch 1.6.0. NAS is implemented by using the open source library Optuna, which offers different efficient search strategies including TPE and recent pruning mechanisms including Hyperband. It supports distributed computations and uses a powerful *define-by-run* API (application programming interface) to describe the search space [16].

3. DATA ACQUISITION AND PREPARATION

A cuboid with a height of l_h = 300 mm, a width of l_w = 600 mm and a length of l_l = 600 mm is defined as the relevant workspace. The workspace is equally divided with z_h = 7, z_w = 7 and z_l = 9 positions in the respective directions. At each position z_o = 5 different

orientations are defined, resulting in a total number of $z = 2{,}205$ measurement poses. The position and orientation of the IR end-effector are measured by the lasertracker Leica AT 960 of the company Hexagon AB, Stockholm, Sweden, with the 6D Leica T-Mac sensor. The measurement setup is illustrated in Fig. 3.

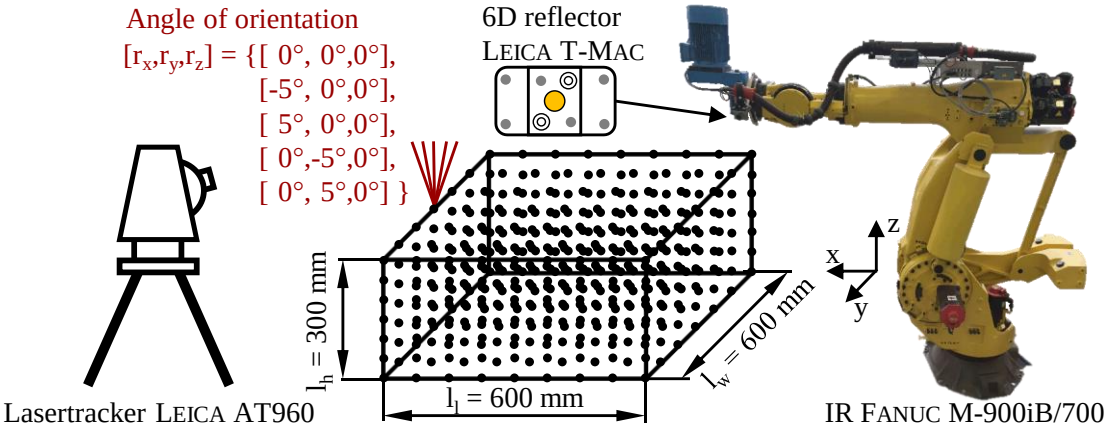

Fig. 3. Measurement setup for data acquisition

An analytical inverse kinematic algorithm is used to calculate the corresponding joint angle vector $\boldsymbol{\theta}$. The measurements are fully automated, resulting in a measurement time of $t_m = 120$ min, and are hence reasonable for industrial purposes. Each measured pose is denoted by a matrix ${}^{S}_{W}\boldsymbol{T}_m$, meaning the transformation from the world frame $\{W\}$ to the sensor frame $\{S\}$. The transformation between the world and the base frame ${}^{0}_{W}\boldsymbol{T}_m$ as well as between the hand and the sensor frame ${}^{S}_{6}\boldsymbol{T}_m$ are identified using the closed-form solution given by Wu and Ren [21]. As only the base and the hand frame are fixed to the IR structure, the ANN should be trained with the transformed data according to Eq. 4.

$$ {}^{6}_{0}\boldsymbol{T}_m = ({}^{0}_{W}\boldsymbol{T})^{-1} \cdot {}^{S}_{W}\boldsymbol{T}_m \cdot ({}^{S}_{6}\boldsymbol{T})^{-1} = \begin{pmatrix} \boldsymbol{R}_m & \boldsymbol{p}_m \\ \boldsymbol{0} & 1 \end{pmatrix} \tag{4} $$

The position data $\boldsymbol{p}_m$ as well as the Euler angles calculated from the rotation matrix $\boldsymbol{R}_m$ are extracted for each pose resulting in the dataset D_1. Thus, dataset D_1 represents the forward kinematics including all error sources.

Dataset D_2 is defined as the errors between the measured and calculated poses extracted from Eq. 4 and Eq. 5, respectively.

$$ {}^{6}_{0}\boldsymbol{T}_c = \prod_{i=1}^{6} {}^{i}_{i\text{-}1}\boldsymbol{T} \tag{5} $$

Given the nominal Denavit Hartenberg (DH) parameters joint angle θ_i, x-translation a_i, z-translation d_i and x-rotation α_i as well as the Hayati-parameter β_i Eq. 5 can be solved using the transformation convention according to Eq. 6.

$$ {}^{i}_{i\text{-}1}\boldsymbol{T} = \mathrm{Rot}(z,\theta_i) \cdot \mathrm{Trans}(x,a_i) \cdot \mathrm{Trans}(z,d_i) \cdot \mathrm{Rot}(x,\alpha_i) \cdot \mathrm{Rot}(y,\beta_i) \tag{6} $$

The error between the measured ${}^{6}_{0}\boldsymbol{T_m}$ and calculated pose ${}^{6}_{0}\boldsymbol{T_c}$ follows according to Eq. 7.

$$ \mathrm{err}\{{}^{6}_{0}\boldsymbol{T_m}, {}^{6}_{0}\boldsymbol{T_c}\} = ((\boldsymbol{p_m} - \boldsymbol{p_c})^T, \mathrm{vec}(\boldsymbol{R_m}\boldsymbol{R_c}^{-1})^{\mathrm{T}})^{\mathrm{T}} \tag{7} $$

The operator vec($\boldsymbol{A}$), $\boldsymbol{A} \in \mathbb{R}^{3x3}$ is expressed as vec($\boldsymbol{A}$) = 0.5(A_{32}-A_{23}, A_{13}-A_{31}, A_{21}-A_{12})^T. Dataset D_2 represents all error sources without capturing the forward kinematics.

A kinematic calibration procedure for a heavy-duty IR with parallelogram-mechanism is performed according to Jingfu et al. [4]. Therefore, the working space is subdivided into eight equally distributed cuboids. From each cuboid ten random poses – resulting in a total number of $z_c = 80$ poses – are used for the kinematic calibration. The remaining errors according to Eq. 7 between the measured ${}^{6}_{0}\boldsymbol{T_m}$ and calculated poses $({}^{6}_{0}\boldsymbol{T_c})_{\mathrm{calibrated}}$ using the calibrated model parameters form dataset D_3. Thus, dataset D_3 represents the remaining error after geometric calibration. Table 2 summarizes the definitions of the three datasets.

Table 2. Investigated datasets

Dataset	Definition	Explanation
D_1	${}^{6}_{0}\boldsymbol{T_m}$	Captures the forward kinematics including all error sources
D_2	$\mathrm{err}\{{}^{6}_{0}\boldsymbol{T_m}, {}^{6}_{0}\boldsymbol{T_c}\}$	Captures all error sources
D_3	$\mathrm{err}\{{}^{6}_{0}\boldsymbol{T_m}, ({}^{6}_{0}\boldsymbol{T_c})_{\mathrm{calibrated}}\}$	Captures the remaining error sources after geometric calibration

In order to speed up convergence of ANN model training, it is advisable to normalize the data of the input and output layer to the range of the activation function. In this work the input and output data of each neuron are normalized to a mean value of zero and a standard deviation of one.

4. ANN TRAINING AND PERFORMANCE EVALUATION

As the optimal hyperparameter set can differ between the three datasets D_1, D_2 and D_3, it is reasonable to perform a hyperparameter optimization for each dataset, individually. The datasets are randomly splitted in a train and a test set with a ratio of 90% to 10%. In each training run a training loss ${}^{T}MSE$ is evaluated for each output neuron according to Eq. 8, where $\boldsymbol{t}_p \in \mathbb{R}^{3x1}$ is a training data point, $\boldsymbol{o}_p \in \mathbb{R}^{3x1}$ the ANN output and B the batch size.

$$ {}^{T}MSE_i = \frac{1}{B}\sum_{p=1}^{B}(o_{i,p} - t_{i,p})^2, \quad i = 1, 2, 3 \tag{8} $$

The adaptive moment estimation optimizer (ADAM) is used to update the weights in each run based on the training loss ${}^{T}MSE$. In order to stabilize the ANN training and to reach convergence, a learning rate scheduler is used to reduce the learning rate l by 25%, if the validation loss ${}^{V}MSE$ does not improve for $n_E = 30$ epochs in a row. The total number of epochs is set to $n_E = 1{,}000$. For each study the ten best trials regarding the validation loss ${}^{V}MSE$ after the training phase are stored. Considering the ANN complexity and the respecttive computation time, the best ANN model is chosen based on the lowest number of degrees of freedom (total number of weights and biases). The ANN model with the identified optimal hyperparameter set is retrained for $n_E = 3{,}000$ epochs. The weights and biases at the epoch, where the ANN model reaches the lowest validation loss ${}^{V}MSE$ on the test set, are stored. Table 3 shows the hyperparameter sets of the optimized ANN models *A*1, *A*2 and *A*3, trained on the position data of dataset D_1, D_2 as well as D_3, respectively.

Table 3. Hyperparameter sets of optimized ANN models

ANN model	*A*1	*A*2	*A*3
Dataset	D_1	D_2	D_3
Number of hidden layers *M*	2	3	2
Number of neurons per hidden layer *N*	92	76	83
Activation function	ELU	ELU	ELU
Bias	true	true	true
Batch size *B*	6	18	20
Learning rate *l*	2.0×10^{-4}	36.6×10^{-4}	32.5×10^{-4}
Dropout probability *d*	0.1×10^{-4}	132.6×10^{-4}	512.9×10^{-4}
Degrees of freedom	18.035	18.319	14.777

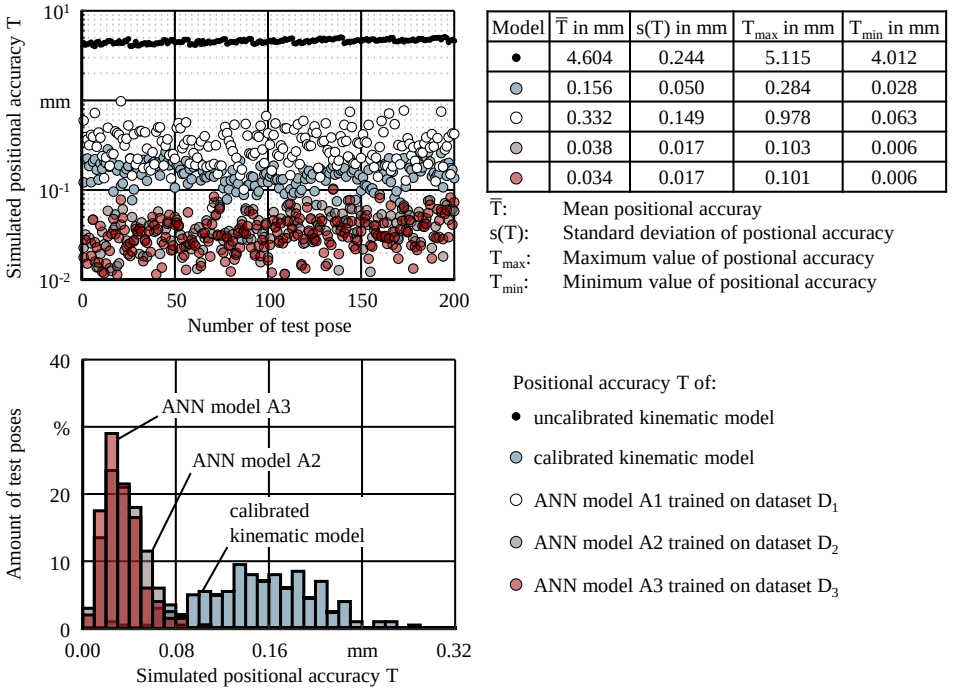

Fig. 4. Simulation results of the ANN models

The ANN model performance is evaluated with the test data. The simulation results of the three ANN models as well as the uncalibrated and calibrated kinematic model are shown in Fig. 4. It can be seen, that the ANN models *A*2 and *A*3 trained on the position error vector $\boldsymbol{e_p}$ outperform the calibrated kinematic model with a mean positional accuracy of $\overline{T} < 0.04$ mm over $\overline{T} = 0.16$ mm. The performance increase regarding the mean positional accuracy of the ANN model *A*3 compared to *A*2 is $\Delta\overline{T} < 0.005$ mm, leading to the conclusion that with a well-trained ANN model a geometric calibration becomes redundant. However, the mean positional accuracy of the ANN model *A*1, which captures the IR-kinematics including all error sources, is $\overline{T} = 0.33$ mm. Therefore, it is concluded that a combination of a nominal kinematic model and an ANN model trained on the uncalibrated error vector $\boldsymbol{e_p}$ leads to the best possible mean positional accuracy $\overline{T}$ of IR.

A hyperparameter optimization is further performed with the orientation data of the three datasets D_1, D_2 and D_3 leading to the ANN models $A1_O$, $A2_O$ and $A3_O$, respectively. Table 4 shows the performance of the ANN models in comparison to the uncalibrated and calibrated kinematic models. Similar to the results obtained with the position data, the ANN model $A2_O$ trained on the uncalibrated orientation error vector $\boldsymbol{e_O}$ performs distinctly better regarding the absolute orientation accuracy *O* than the calibrated kinematic model.

Table 4. Comparison of the simulated orientation accuracy

ANN model	Uncalibrated kinematic model	Calibrated kinematic model	$A1_O$	$A2_O$	$A3_O$
Mean value $\overline{O}$	9.54’	0.60’	2.66’	0.22’	0.24’
Standard deviation $s(O)$	0.35’	0.21’	2.34’	0.10’	0.12’
Maximum value O_{max}	10.48’	1.24’	21.77’	0.59’	0.80’
Minimum value O_{min}	8.74’	0.06’	0.17’	0.01’	0.04’

4.1. DEPENDENCY OF THE ANN PERFORMANCE UPON THE SIZE OF THE TRAINING DATASET

It is time-wise expensive to generate a large dataset to train, validate and test the ANN model. In order to investigate the influence of the size of the training dataset, the ANN model *A*2 is additionally trained with 3.125%, 6.25%, 12.5%, 25% and 50% of the total training dataset with $z_{train} = 1{,}985$ points, respectively. The test dataset remained constant for all investigations.

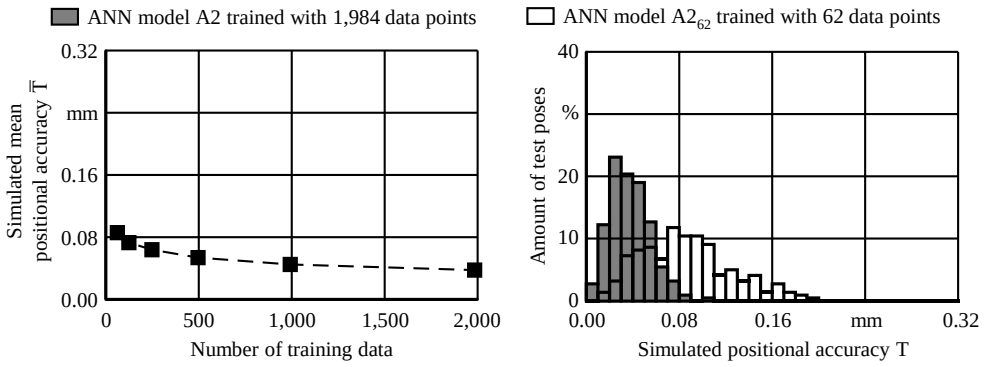

Fig. 5. Dependency of the ANN performance on the number of training data

Fig. 5 shows the mean positional accuracy $\overline{T}$ depending on the number of training data. As expected, the mean positional accuracy $\overline{T}$ becomes better with an increasing number of training data. However, with z_{train} = 62 training data (3.125%) the mean positional accuracy calculated with the total test dataset is already $\overline{T}$ = 0.09 mm and would therefore outperform the calibrated kinematic model.

4.2. INTERPOLATION AND EXTRAPOLATION CAPABILITY OF THE ANN MODELS

In the previous investigations, the hyperparameter optimization as well as the performance evaluation were conducted by using train and test data distributed in the whole considered workspace of the IR. Further, the dependency of the ANN performance upon the size of the considered workspace is investigated by subdividing the workspace into eight equally sized sub-workspaces. A hyperparameter optimization is performed with the training data of the cuboid $Q1$, see Fig. 6. The ANN models $A2$ and $A2_{Q1}$ are compared regarding the test data of cuboid $Q1$ in Fig. 6a. The simulated mean positional accuracy is $\overline{T}$ = 0.030 mm and $\overline{T}$ = 0.032 mm for the ANN models $A2$ and $A2_{Q1}$, respectively. As a reduction in the scope of validity of the ANN model does not improve the mean positional accuracy $\overline{T}$, it can be assumed that the ANN models are capable to capture the error sources in a general manner. Further, the extrapolation capability of the ANN model $A2_{Q1}$ is tested by using the total test dataset distributed over the whole considered workspace. The results are shown in Fig. 6b.

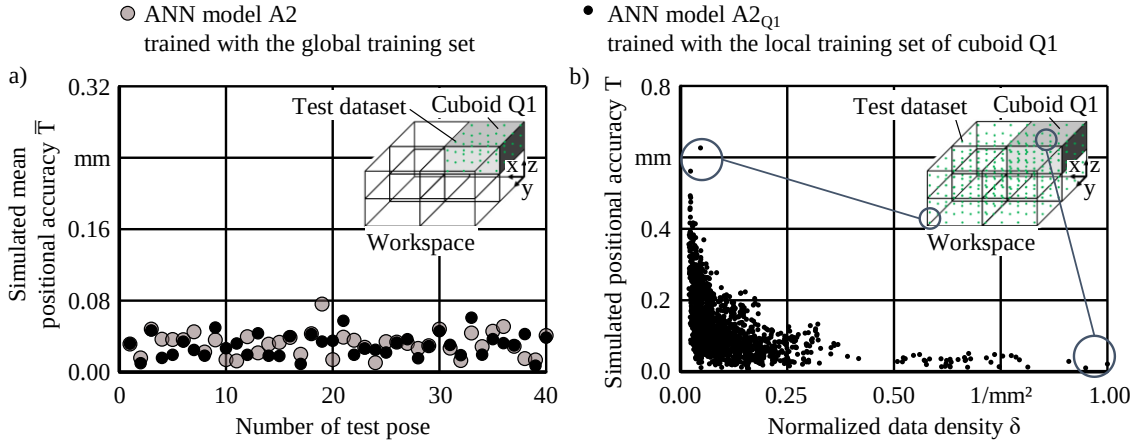

Fig. 6. Performance of the ANN models $A2$ and $A2_{Q1}$ evaluated as a) an interpolation problem within the cuboid $Q1$, b) an extrapolation problem outside the cuboid $Q1$

The data density δ is calculated according to Eq. 9:

$$\delta = \sum_i \frac{1}{R_i^n} \quad \text{, with } R_i = |\boldsymbol{p}_{\text{train},i} - \boldsymbol{p}_{\text{test}}|,\ n \in N \tag{9}$$

where R_i is the Euclidean distance between a fixed test and the i-th train data point. In Fig. 6b the parameter n is chosen to be n = 2. From Fig. 6b it can be seen that even with a low data density δ the ANN model $A2_{Q1}$ is capable to predict the position error vector $\boldsymbol{e_p}$

quite accurately, e.g. for $\delta < 0.1$ the simulated mean positional accuracy is $\overline{T} = 0.16$ mm. Especially with an increasing data density δ, the positional accuracy T is rapidly improving. The extrapolation capability of the ANN model $A2_{Q1}$ supports the conclusion that a well-suited ANN model can capture the relation between the joint angle vector of an IR $\boldsymbol{\theta}$ and the position error vector $\boldsymbol{e_p}$ in a general manner. It should, however, be emphasized, that a large variation of the input parameters improves the generality of the model.

5. SUMMARY

The proposed method deals with an algorithm-based choice of hyperparameters of ANN models such as number of hidden layers, number of neurons, activation function, batch size, learning rate and dropout probability, in order to improve the positional accuracy of IR. Only a few researches deal with ANN models in robot calibration and none of them explicitly uses a neural architecture search to improve the positional accuracy, although a bad choice of hyperparameters can significantly affect the ANN performance. Within this paper it could be demonstrated that with a suitable choice of hyperparameters the simulated positional accuracy can be improved to $T < 0.05$ mm by means of an ANN model. The results show that a classical geometric calibration of the IR-kinematic model becomes redundant, if a suitable ANN model is used to predict the pose error.

ACKNOWLEDGEMENTS

The research presented in this paper is funded by the German Federal Ministry for Economic Affairs and Energy (BMWi).

REFERENCES

[1] ELATTA A.Y., GEN L.P., ZHI F.L., DAOYUAN Y., FEI L., 2004, *An Overview of Robot Calibration*, Information Technology Journal, 3/1, 74–78.

[2] KHALIL W., DOMBRE E., 2004, *Modeling, Identification and Control of Robots*, Butherworth-Heinemann, Oxford.

[3] VEITSCHEGGER W., WU C.H., 1987, *A Method for Calibrating and Compensating Robot Kinematic Errors*, Proceeding IEEE International Conference on Robotics and Automation, 39–44.

[4] JINGFU P., YE D., GANG Z., HAN D., 2019, *An Enhanced Kinematic Model for Calibration of Robotic Machining Systems with Parallelogram Mechanisms*, Robotics and Computer Integrated Manufacturing, 59, 92–103.

[5] KLIMCHIK A., 2012, *Enhanced Stiffness Modeling of Serial and Parallel Manipulators for Robotic-Based Processing of High Performance Materials*, PhD Thesis, Ecole des Mines de Nantes, France.

[6] NGUYEN H.N., LE P.N., KANG H.J., 2019, *A New Calibration Method for Enhancing Robot Position Accuracy by Combining a Robot Model–Based Identification Approach and an Artificial Neural Network–Based Error Compensation Technique*, Advances in Mechanical Engineering, 11/1, 1–11.

[7] FAN J., MA K., ZHONG Y., 2019, *A Selective Overview of Deep Learning*, arXiv preprint.

[8] ANGELIDIS A., VOSNIAKOS G.C., 2014, *Prediction and Compensation of Relative Position Error Along Industrial Robot End-Effector Paths*, Int. J. Precis. Eng. Manuf., 15/1, 63–73.

[9] NGUYEN H.N., ZHOUG J., KANG, H.J., 2015, *A Calibration Method for Enhancing Robot Accuracy Through Integration of an Extended Kalman Filter Algorithm and an Artificial Neural Network*, Neurocomputing, 151, 996–1,005.

[10] AOYAGI S., KOHAMA A., NAKATA Y., HAYANO Y., SUZUKI M., 2010. *Improvement of Robot Accuracy by Calibrating Kinematic Model Using a Laser Tracking System – Compensation of Non-Geometric Errors Using Neural Networks and Selection of Optimal Measuring Points Using Genetic Algorithm*, IEEE RSJ International Conference on Intelligent Robots and Systems, 5,660–5,665.

[11] ZHU Q., LI J., YUAN P., SHI Z., LIN M., CHEN D., WANG T., 2016, *Accuracy Compensation of a Spraying Robot Based on RBF Neural Network*, International Conference on Advanced Robotics and Mechatronics (ICARM), 414–419.

[12] CAI Y., YUAN P., CHEN D., GAO D., WU X., XUE L., WANG T., 2017, *A Calibration Method of Industrial Robots Based on ELM*, 2nd International Conference on Advanced Robotics and Mechatronics (ICARM), 70–75.

[13] YUAN P., CHEN D., WANG T., CAO S., CAI Y., XUE L., 2018, *A compensation Method Based on Extreme Learning Machine to Enhance Absolute Position Accuracy for Aviation Drilling Robot*, Advances in Mechanical Engineering, 10/3, 1–11.

[14] LE P.N., KANG H.J., 2020, *A Robotic Calibration Method Using a Model-Based Identification Technique and an Ivasive Weed Optimization Neural Network Compensator*, Applied Sciences, 10/20, 1–14.

[15] ELSKEN T., METZE J.H., HUTTER F., 2019, *Neural Architecture Search: A Survey*, Journal of Machine Learning Research, 20, 1–21.

[16] AKIBA T., SANO S., YANASE T., OHTA T., KOYAMA M., 2019, *Optuna: A Next-generation Hyperparameter Optimization*, Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Anchorage AK USA, 2,623–2,631.

[17] BERGSTRA J., BENGIO Y., 2012. *Random Search for Hyper-Parameter Optimization*, Journal of Machine Learning Research, 13, 281–305.

[18] BERGSTRA J., BARDENET R., BENGIO Y., KÉGL B., 2011. *Algorithms for Hyper-Parameter Optimization*, Proceedings of the 24th International Conference on Nerual Information Processing Systems, 2,545–2,554.

[19] JAMIESON K., TALWALKAR A., 2016. *Non-Stochastic Best Arm Identification and Hyperparameter Optimization*, Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, 51, 240–248.

[20] LI L., JAMIESON K., DESALVO G., ROSTAMIZADEH A., TALWALKAR A., 2018, *Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization*, Journal of Machine Learning Research, 18, 1–52.

[21] WU L., REN H., 2013, *Finding the Kinematic Base Frame of a Robot by Hand-Eye Calibration Using 3D Position Data*, IEEE Transactions on Automation Science and Engineering, 14/1, 314–324.