Simon Hagemann, Atakan Sünnetcioglu, Tobias Fahse, Rainer Stark

Neural Network Hyperparameter Optimization for the Assisted Selection of Assembly Equipment

Hagemann, S., Sünnetcioglu, A., Fahse, T., & Stark, R. (2019). Neural Network Hyperparameter Optimization for the Assisted Selection of Assembly Equipment. 2019 23rd International Conference on Mechatronics Technology (ICMT). https://doi.org/10.1109/icmect.2019.8932099

Neural Network Hyperparameter Optimization for the Assisted Selection of Assembly Equipment

Simon Hagemann
Daimler AG
Digital Factory
Sindelfingen, Germany
simon.hagemann@daimler.com

Atakan Sünnetcioglu
Fraunhofer IPK
Division Virtual Product Creation
Berlin, Germany
atakan.suennetcioglu@ipk.fraunhofer.de

Tobias Fahse
Daimler AG
Digital Factory
Sindelfingen, Germany
tobias.fahse@daimler.com

Rainer Stark
Technische Universität Berlin
Department of Industrial Information Technology
Berlin, Germany
rainer.stark@tu-berlin.de

Abstract— **The design of assembly systems has been mainly a manual task including activities such as gathering and analyzing product data, deriving the production process and assigning suitable manufacturing resources. Especially in the early phases of assembly system design in automotive industry, the complexity reaches a substantial level, caused by the increasing number of product variants and the decreased time to market. In order to mitigate the arising challenges, researchers are continuously developing novel methods to support the design of assembly systems. This paper presents an artificial intelligence system for assisting production engineers in the selection of suitable equipment for highly automated assembly systems.**

Keywords— artificial intelligence, assembly system design, automotive, body-in-white, neural network, hyperparameter optimization

I. Introduction

The assembly system design (ASD) covers a wide spectrum of different tasks that are performed manually in industrial practice. Nevertheless, different fields of research have emerged to improve the automation or assistance of ASD, such as the automated product analysis or the robotic assembly line balancing. [1, 2] Currently, in industrial ASD two different design approaches are commonly applied: (i) conventional, manual ASD, which requires a substantial investment of time and effort (ii) template-based assembly system design, based on a virtual assembly system model, which however limits the solution space of product development significantly. [3–5] Optimally, the template does not require any changes. In case significant changes are required, the template-based approach is highly inefficient and depicts an error prone process.

The automated or assisted ASD is not yet widespread in industrial practice. However, it shows high benefits in accelerating the design process, increasing the optimality of the design solution and supporting the documentation consistency. Various authors have been emphasizing the high potential of the assisted ASD: Bauernhansl et. Al. show the potential of intelligent assistance software and call for the essential need of software support due to the increasing complexity of production systems [6]. Bracht et. al. discuss a fractional automation of ASD processes, for instance, the automated welding gun selection [7]. Moreover, Michalos et. al. refer to the large size and complexity of current assembly lines in automotive industry and demand methods to assist the balancing of assembly lines. [8] Therefore, the authors do not primarily see the major potential in reducing the work dedicated to the design, rather in achieving a higher quality and more optimal assembly system configurations, which engineers might not consider.

Please note the structure of this paper: Firstly, the state of the art in the field of assisted and automated assembly system design will be presented. Afterwards, the new method and software prototype for the assisted assembly equipment selection will be described in detail. Lastly, the authors outline the optimization process of the neural network (NN) and its results.

II. State of the Art and Assembly System Design Process

In the early phases of assembly system design, engineers need to complete the following activities in highly iterative processes:

(i) *Product Analysis and Assembly Sequence:*
The ASD starts with the product data analysis. The engineer classifies the joining elements under the consideration of their parameters such as the joining technology or material properties. The holistic product analysis enables the assembly planner to decide in which order the single parts and sub-assemblies are to be joined – the so-called assembly sequence. The assembly sequence depends on multiple factors, e.g. the accessibility of joining elements, the flange positions, etc. Previous publications have focused on the automated product analysis and the creation of possible assembly sequences such as [9, 10].

(ii) *Assembly System Configuration and Balancing:*
The joining sequence enables the derivation of the initial assembly process and rough configuration of the assembly system. The assembly system configuration is primarily defined by the number of stations and the number of robots (workers) and is based on the extensive station balancing. For the automated balancing and assembly system configuration, a vast number of publications are available such as [11–13].

(iii) *Equipment Selection and Specification:*

The assembly system configuration (see step ii) determines the number of resource types in each station. Resource types serve as placeholders, which need further specification. This manual equipment selection and specification problem has been analyzed extensively in literature. However, no systematic literature review and classification methodology exist for this problem. Various papers in literature focusing on the assisted selection of assembly equipment, such as [14] and [15], which present methods for the automated selection of industrial robots. Other authors, for instance [16, 17] concentrate on the assisted selection of machine tools and handling equipment. Nevertheless, none of the presented methods follow a generic approach able to suggest different kinds of assembly equipment.

(iv) *Assembly Facility Layout:*

After the equipment is selected and the assembly system configuration is set, the system needs to be positioned in the production facility – the assembly facility layout. A large number of research projects have focused on the search for suitable algorithms. Especially the search for suitable heuristics has been a major topic, such as in [18–20].

III. Assisted Selection of Assembly Equipment

This chapter begins with the description of the industrial scenario, which has been used to develop the hereby presented method and software prototype. Then, the method – i.e. the necessary activities for the assisted selection of assembly equipment – and the software architecture is introduced. Subsequently, the prototype implementation will be presented.

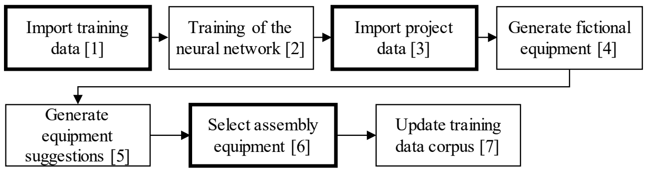
A. Industrial Scenario

It has been stated that the automated or assisted selection and specification problem has not been regarded sufficiently by research and industry. Exactly this gap will be tackled in this research paper. Therefore, real automotive assembly scenarios from the body-in-white (BiW) production stage have been used to develop the here presented method and software prototype. In industrial equipment selection, the input data depict an assembly system configuration and an assembly sequence enriched with joining element information. Assembly system configurations consist of different resource modules, which serve as placeholders for real assembly equipment. The resource modules in BiW are mainly of the following types: (i) joining robot, (ii) handling robot, (iii) fixture, (iv) carrier, (v) buffer. The target of the equipment selection is to replace those placeholders by real resources from a resource library in which several thousand resources are stored.

B. Method Description and Software Architecture

The method for the assisted selection of assembly equipment uses a neural network that can predict the values of the parameters of assembly equipment (output layer) based on the parameter values of the assembly process and pre-defined assembly station parameters (input layer).

The first stage of the method (see Fig. 1) is the training process, which begins with the import and interpretation of input data, composed of information about the assembly process and the corresponding assembly system (step 1).



Import training data [1]
Training of the neural network [2]
Import project data [3]
Generate fictional equipment [4]
Generate equipment suggestions [5]
Select assembly equipment [6]
Update training data corpus [7]

Fig. 1. Assembly system with assigned equipment

The process information contains data about the assembly operations that are necessary to join the chassis components of a vehicle. Among others, parts to join (e.g. part number, size), the assembly sequence, types of assembly process (e.g. spot weld or arc weld) and attributes of the assembly process such as the number of joined parts are included. On the other hand, information about the assembly system reveals the stations and assembly equipment of the production system that has been designed to perform the necessary assembly operations for the considered chassis components.

The training of the neural network (step 2) takes place based on the links between assembly stations and assembly operations, which represent the domain knowledge of production planners. After the training is completed, the next stage of the method can begin.

The assisted selection of assembly equipment begins with the import and interpretation of project data (step 3). The first necessary input is the process information with the same contents as given above. In contrast to the training stage, the process information addresses the assembly process of a newly developed chassis, for which the necessary assembly equipment needs to be selected. In other words, a specific assembly system for the given operations does not exist yet. The second input information provides a general framework for the designed production system, which consists of the number of stations and placeholder equipment for each station.

In step 4, the neural network begins to generate fictional equipment by defining the parameter of suitable equipment (output layer) based on the parameter values of the assembly process (input layer). The outcomes of the neural network are called fictional because there is no guarantee that the calculated parameters exactly match to the parameters of any existing equipment in the resource library of Daimler.

Due to the fact that the fictional equipment is not applicable in assembly system design, a filter function compares the parameters of fictional and existing equipment. Based on this comparison, equipment suggestions are generated which are regarded as suitable for the concerning assembly process (step 5). Subsequently, the production planner examines the suggested equipment for a given assembly process and selects the most suitable assembly equipment (step 6).

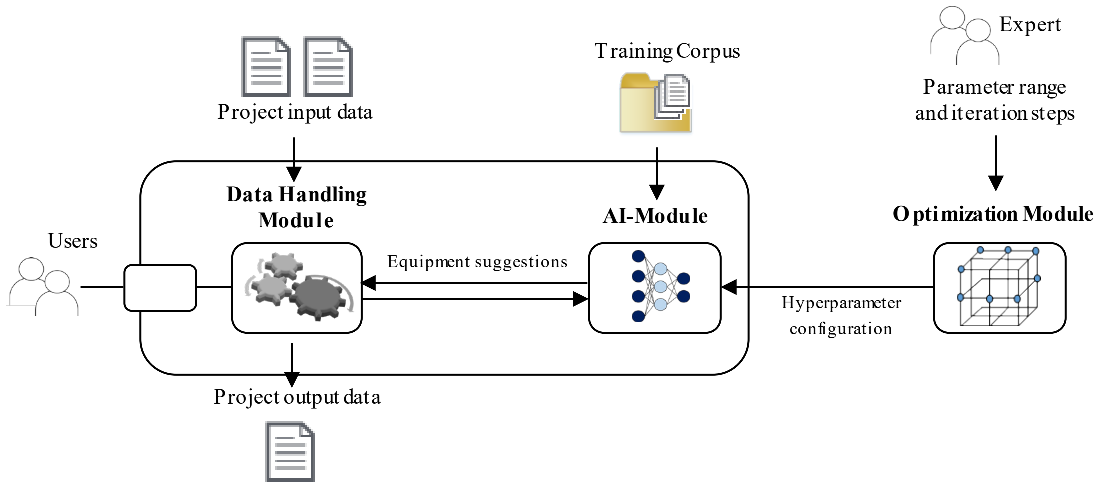


Fig. 2. The conception of the software architecture

During the assisted selection of equipment, the method updates the training data corpus based on the created link between the assembly process and equipment (step 7).

In Fig. 2, the conception of the software architecture is shown with all its elements required for enabling the aforementioned activities. The architecture consists of three modules. The core of the architecture depicts the data handling module and the AI-module. The third module comprises the hyperparameter optimization. Between the modules, arrows indicate the data flow. The functionality of the data handling and AI-modules and the corresponding data flow will become apparent in the following chapter. The optimization module and the hyperparameter configuration will be discussed in detail in Chapter IV.

C. Prototype Implementation

This section provides information about the software prototype, which implements the proposed method in order to provide a platform to evaluate the suitability of artificial intelligence (AI) in the context of assisted selection of assembly equipment.

The essential information regarding the assembly process, assembly system and general framework for projects are imported via the AutomationML (AML) interface. As stated in the previous section, the method requires two concurrent files as input for training and equipment selection modes. In order to avoid wrong data transfers, the software prototype allows importing ZIP files that contain two AML files. Note that all imported AML files are converted to input matrices in CSV format to enable the usage of data by the AI algorithms. In addition, an interface for Excel sheets is available to import the resource library.

As shown in Fig. 3, the graphical user interface (GUI) of the software prototype consists of two main workspaces. On the left side, the structure of the assembly system (i.e. stations and associated equipment) are shown. Placeholder equipment is visualized in red to differentiate from the already selected equipment by the planner. The right side of the GUI provides information about the selected equipment, placeholder equipment, fictional equipment and suggested best match and alternatives. Note that the suggestions are sorted by the calculated score, which is computed based on a comparison of the parameters of the fictional and existing equipment using the Manhattan distance metric. In case two or more suggestions have the same score value, they are sorted in accordance to the count of their occurrences in the training data set. In other words, suggestions that were preferred more often in previous projects are favored than the less frequently applied assembly equipment in case they have the same score.

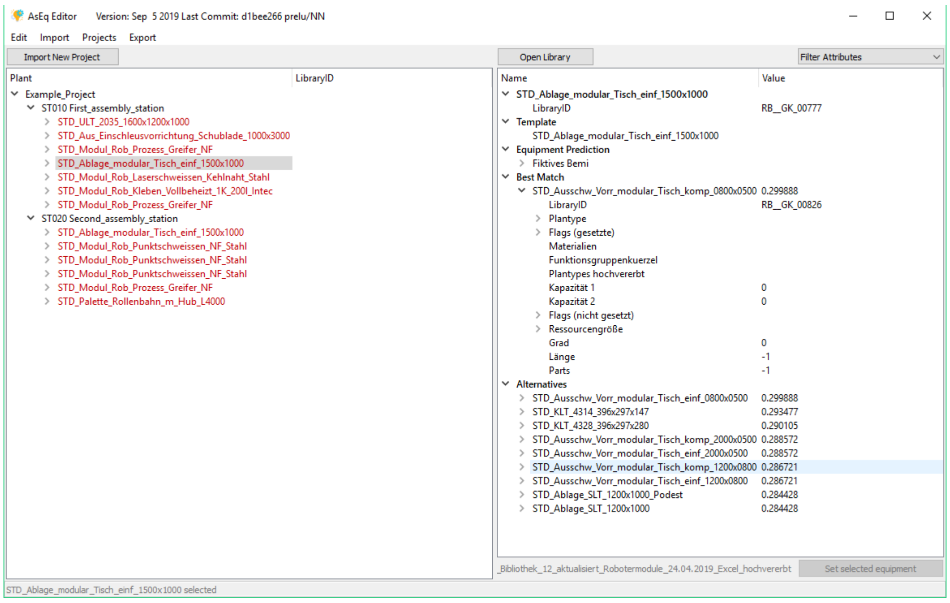
The entries on the right side are expandable to enable the analysis of parameters if necessary. As shown in Fig. 3, equipment parameters such as library ID, set flags and plantypes (to label the capabilities of equipment, such as handling, curve, input), capacity values (e.g. load capacity of a robot arm) and size of the equipment. If necessary, it is possible to hide undesired parameters to limit the shown information on the graphical user interface (see the dropdown menu “Filter Attributes”).

In addition to the assisted selection of assembly equipment, the software prototype also supports a complete manual selection to cover the cases that the planner does not agree to any of the suggested equipment. The manual equipment selection is performed by opening the resource library (see the button “Open Library”).

The important settings regarding the neural network and the filter function are managed using a configuration file that is saved in the same directory with the software prototype. For instance, the values of the neural network parameters such as the learning rate is controlled within the configuration file. In addition, the weighting coefficients for the score calculation can be changed flexibly to modify the filter function.

IV. Optimization of Predictions

The implementation of the method in a software prototype must be followed by the determination of the neural network parameters. The correct configuration of artificial intelligence algorithm has a crucial role on the quality of predictions. However, the theoretical knowledge about the AI parameters is not sufficient for the optimal configuration of the neural network with regard to the specific use case of authors’ research. Therefore, a parameter configuration tool was developed to support the identification of the optimal configuration.



```
Plant                                                      LibraryID
v Example_Project
  v ST010 First_assembly_station
    > STD_ULT_2035_1600x1200x1000
    > STD_Aus_Einschleusvorrichtung_Schublade_1000x3000
    > STD_Modul_Rob_Prozess_Greifer_NF
    > STD_Ablage_modular_Tisch_einf_1500x1000
    > STD_Modul_Rob_Laserschweissen_Kehlnaht_Stahl
    > STD_Modul_Rob_Kleben_Vollbeheizt_1K_200l_Intec
    > STD_Modul_Rob_Prozess_Greifer_NF
  v ST020 Second_assembly_station
    > STD_Ablage_modular_Tisch_einf_1500x1000
    > STD_Modul_Rob_Punktschweissen_NF_Stahl
    > STD_Modul_Rob_Punktschweissen_NF_Stahl
    > STD_Modul_Rob_Punktschweissen_NF_Stahl
    > STD_Modul_Rob_Prozess_Greifer_NF
    > STD_Palette_Rollenbahn_m_Hub_L4000
```

```
Name                                                  Value
v STD_Ablage_modular_Tisch_einf_1500x1000
    LibraryID                                         RB__GK_00777
v Template
    STD_Ablage_modular_Tisch_einf_1500x1000
v Equipment Prediction
  > Fiktives Bemi
v Best Match
  v STD_Ausschw_Vorr_modular_Tisch_komp_0800x0500    0.299888
      LibraryID                                       RB__GK_00826
    > Plantype
    > Flags (gesetzte)
      Materialien
      Funktionsgruppenkuerzel
      Plantypes hochvererbt
      Kapazität 1                                     0
      Kapazität 2                                     0
    > Flags (nicht gesetzt)
    > Ressourcengröße
      Grad                                            0
      Länge                                           -1
      Parts                                           -1
v Alternatives
  > STD_Ausschw_Vorr_modular_Tisch_einf_0800x0500    0.299888
  > STD_KLT_4314_396x297x147                          0.293477
  > STD_KLT_4328_396x297x280                          0.290105
  > STD_Ausschw_Vorr_modular_Tisch_komp_2000x0500    0.288572
  > STD_Ausschw_Vorr_modular_Tisch_einf_2000x0500    0.288572
  > STD_Ausschw_Vorr_modular_Tisch_komp_1200x0800    0.286721
  > STD_Ausschw_Vorr_modular_Tisch_einf_1200x0800    0.286721
  > STD_Ablage_SLT_1200x1000_Podest                   0.284428
  > STD_Ablage_SLT_1200x1000                          0.284428

_Bibliothek_12_aktualisiert_Robotermodule_24.04.2019_Excel_hochvererbt
```

```
STD_Ablage_modular_Tisch_einf_1500x1000 selected
```

Fig. 3. GUI of the software prototype

When executed, the configuration tool calculates the rate of overlap (i.e. hit rate) between correct assembly equipment and the suggested assembly equipment for each parameter configuration. The hit rate counts as the performance indicator for the evaluation of possible parameter configurations.

The applied optimization approach searches for the optimal configuration by varying the AI hyperparameters with equally sized steps within the predefined value ranges (see Table I). For instance, the parameter nn_limit is varied using the values 500, 1000 and 2000.

TABLE I. VALUES OF PARAMETERS FOR GRID SEARCH

Parameter	Value 1	Value 2	Value 3	Value 4
nn_limit	500	1000	2000	
max_epoch	100	200	300	
mini_batch_size	50			
learning_rate	0.0001	0.001	0.01	0.1
solver_weight_decay	0	1		
solver_momentum	0	1		

In addition to the hyperparameters, also the shape and the structure of the neural network is varied within this investigation. Shape is a hyperparameter that affects the structure of the NN and has two forms. The first form is a rectangular shape where the amount of neurons in each layer is the same. The second form is a pyramid shape, meaning that the amount of neurons decreases towards the output layer and hereby forming a triangular shape. In addition, the number of hidden layers are varied between 1 to 3, whereby structures without hidden layers are not taken into consideration in our experiments since such neural network configurations are not capable of learning non-linear input/output patterns [21]. A further factor that affects the structure of the neural network is the coefficient for the number of neurons on each layer, which is varied between 1 and 4.

The initial definition of the hyperparameters base on the common recommendations in the literature and previous experiences of the authors. It is aimed to limit the variation sparingly to reduce the necessary processing time to acceptable levels. (The configuration tool required around 60 hours to process 2880 different configurations on a typical business PC without GPU usage.)

Among all possible configurations, the rates for TOP 10 match (i.e. the correct equipment is among the first 10 suggestions according to their score values) vary between 41.84% and 96.45% while the average value equals to 80.84%. In total, three different configurations deliver the highest average values, which are shown with details in Table II. Note that the configurations are identical except the values for max_epoch, which means that the max_epoch does not cause any difference regarding the hit rate in the top configurations. Aside from max_epoch, the superior values for the hyperparameters are recognizable (e.g. nn_limit with 2000, only one hidden layer or rectangular NN shape).

TABLE II. RESULTS OF HYPERPARAMETER OPTIMIZATION WITH GRID SEARCH

Hyperparameter	C1	C2	C3
nn_limit	2000	2000	2000
max_epoch	100	200	300
mini_batch_size	50	50	50
learning_rate	0.1	0.1	0.1
solver_weight_decay	0	0	0
solver_momentum	0	0	0
Shape	Rectang.	Rectang.	Rectang.
No. of hidden layers	1	1	1
Coefficient	1	1	1
TOP 10 hit-rate	96.45%	96.45%	96.45%

A further analysis of the results reveals that a hit rate of 58.51% is possible for exact matches, i.e. the suggested equipment with the highest score is indeed the correct assembly equipment. This hit rate is especially important for the considerations regarding the applicability of the method for a fully automated selection of assembly equipment.

In the next chapter, a more advanced strategy for hyperparameter tuning is presented. The aim of this strategy is to be able to identify the best hyperparameter combinations in a shorter time due to the reduced number of combinations compared to grid search strategy.

V. Further Optimization Potential

In order to further improve the results of the software prototype, the initial neural network could be divided into individual neural networks for each output variable. Under this condition, the activation function of the output layer can be chosen under consideration of the variable type. Tailored activation functions allow more precise predictions. For continuous output variables, linear functions have been proven successful. For binary variables the sigmoid function. [22]

In that case, each neural network itself has to be tuned regarding its hyperparameters. This increases the amount of parameters to be tested by the number of predictable outputs. In addition, the parameter optimization approach introduced in chapter IV has shown that the computational effort increases with the number of steps and the number of parameters that are tested. The computational effort therefore increases vastly if the abovementioned optimization potential is exploited or more precise parameter value estimations are desired. Those facts demand for replacing the grid search strategy by a method with lower computational effort. A possibility is to use a method the authors call implicit search strategy. [23] In the following, this tuning process is described for a single neural network.

The used hyperparameters equal the ones enumerated in Chapter IV. All hyperparameters are varied in each optimization loop in order to detect interdependencies. A priorly created search tree determines the optimization room for each parameter. The general approach is iterative. Several iterations are executed in order to approximate the best possible value of each parameter. Hence, the advantage of the new calculation method is a heuristic approach to decrease the amount of combinations calculated. Therefore, the testing of all values and parameters is enabled in each iteration. The result is a unique configuration of parameter values for each neural network.

Fig. 4 shows a search tree for the parameter “Batch Size” as an example. Each iteration equals one level of the search-tree. In each iteration, the algorithm only expands the best-rated node of each iteration. In the shown case, the parameter “Batch Size” assumes in each iteration the grey value for one specific neural network. For predicting a different output, a different neural network is used, which possibly follows a different path.

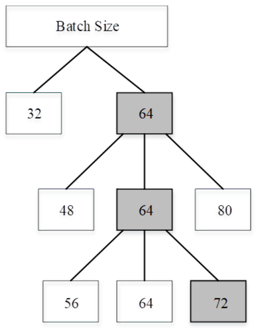


Fig. 4. Search Tree for parameter "Batch Size"

In order to force the algorithm to converge, the range of possible values for each parameter is decreased in each iteration. The authors have chosen to halve the range of possible values in each iteration and hereby the algorithm continuously searches on a more granular grid. Therefore, the delta of the analyzed hyperparameter values decreases in each iteration. For instance, in the first iteration the highest batch size delta between the different parameter configurations is 32. In the third iteration, the delta decreases to the value of 8. Eventually, the parameter would change slightly more in a higher number of iterations and lead to an increased result quality.

Nevertheless, a trade-off between number of iterations and the quality of the resulting neural networks exists. Initial calculations indicated that in the third iteration, the quality of the neural network does not change significantly anymore. If the method is applied in different environments, a higher number of iterations may be needed. Hence, an objective stop criterion is beneficial in order to justify a termination of the method.

The best-rated node is chosen by evaluating all possible parameter combinations in each iteration regarding a quality criterion. A possible quality criterion is the value of the loss function when applying the trained neural network on a test-set. In each iteration, the algorithm chooses between all offered specifications of a parameter by training each configuration of the neural network and evaluating the value of the loss function on the test-set for each configuration. The aforementioned stop criterion could therefore be the percentage change of the loss function.

Search trees similar to Fig. 4 are created for all parameters named in Chapter IV. The number of parameter configurations, which are to be tried, varies depending on the number of possible parameter values of each parameter in each iteration. Let $a_{j,i}$ be the number of possible values of parameter j in iteration i. If m parameters are tuned, this leads to $\prod_{j=1}^{m} a_{j,i}$ different configurations which are tested in the iteration i.

The benefit of the implicit search strategy is the reduction of parameter configurations that have to be tested while keeping a certain flexibility: Even if in the first iteration the lowest of all manifestations of a parameter is chosen, this choice can be adjusted towards higher values in further iterations. This ability of the implicit search strategy is very important due to interdependencies between different parameters: A change of one parameter can affect the optimal value for another parameter. The reduction of parameter configurations becomes apparent in

Fig. 4. During three iterations, the implicit search strategy is able to find a value for the parameter "Batch Size" with a precision of 8 (i.e. delta=8) while testing 2+3+3=8 parameter configurations. All configurations of the value set A= {8, 16, 32, 40, 48, 56, 64, 72, 80, 88} could have been identified. In order to achieve the same precision by using Grid Search, 11 parameter configurations would have to be tested, namely all configurations of the value set A. This equals a reduction of 27%. The percentage reduction increases with higher precision: If 6 iterations are used (precision of 1), implicit search tests 17 configurations, while Grid Search tests 95 configurations (reduction of 82%).

VI. Conclusion

In the previous chapter, the authors have presented the results of the performed experiments with the aim to find the best hyperparameter configuration and structure for the applied neural network. In this chapter, a short summary of the obtained results is given, followed by the recognized threats to the validity of the results and future research.

A. Summary of the Results

The obtained results from the grid search show that it is possible to reach a hit rate of 96.45% is for TOP 10 match. The analysis of the best performing configurations show that a rectangular shaped neural network with one hidden layer and no multiplicator for the number of neurons delivers the best result. The values of the varied AI hyperparameters within the best performing configuration are as follows: nn_limit (2000), learning_rate (0.1), solver_weight_decay (0), and solver_momentum (0). On the other hand, it is not possible to name a superior value for the hyperparameter max_epoch.

A possible strategy to improve the obtained results is the splitting of the neural network according to binary and continuous variables in the output layer. As a result, the output of the neural network is tailored to the desired output range, which is expected to lead to a higher precision. One disadvantage of this approach is the increased computational effort for the hyperparameter tuning process due to the increased count of neural networks. To overcome this issue, an implicit search strategy has been proposed instead of the grid search strategy.

An experiment that has been conducted to test the implicit search strategy has shown that in each iteration, the quality of the predictions improves. In order to compare the implicit search strategy to other hyperparameter tuning approaches like "Bayesian optimization" or gradient-based approaches, a comparison with an example dataset would be required in order to properly compare speed and precision of the different approaches for hyperparameter tuning.

B. Threats to Validity

The conducted experiments base strongly on the defined value ranges and the steps of the hyperparameters. Due to time restriction during research, the authors have limited the value ranges and steps sparingly. Therefore, it is not possible to exclude the possibility of achieving better results with the hyperparameter values that have not been taken into consideration in the experiments. Furthermore, the random separation of training and test data is a non-controllable factor in the experiments. To mitigate the undesired consequences of the random separation of data, it is recommended to repeat the experiments to gather multiple results set.

A further threat to the validity of the obtained results has a natural cause that all performed experiments are using test data from already existing assembly systems. That means, it is possible to optimize the hyperparameter configuration in accordance to the existing assembly systems (retrospective optimization), but this cannot guarantee that the configured neural network fits optimally for assembly systems that are yet to be created.

Lastly, it is not possible to guarantee that the used training data corresponds to the real assembly stations of the production site. If operational changes to the production plant are made after the project data has been used as training data, the AI system is trained on false assumptions.

C. Future Research

In this paper, it has been shown that the configuration and hyperparameter settings of the neural network have a considerable impact on the equipment selection results. Even though the system has been optimized significantly, there is still potential for improvement: Similar to the splitting of the neural network into multiple networks, it is conceivable to activate specific neural networks for each resource type, such as tools, grippers, et cetera.

Despite the possibilities to improve the AI system itself, the data basis has to be further improved. Analyses have shown that a high percentage of the equipment cannot be distinguished explicitly, based on their given attributes. In specific terms, the resource library contains many "similar" equipment, which makes our AI-driven selection of assembly equipment more difficult. In the same way, the given training basis for development proposes of our method needs to be sufficiently increased, especially if the proposed method needs to be prepared for pilot projects.

References

[1] N. Boysen and M. Fliedner, "A versatile algorithm for assembly line balancing," *European Journal of Operational Research*, vol. 184, no. 1, pp. 39–56, 2008.

[2] G. Michalos, A. Fysikopoulos, S. Makris, D. Mourtzis, and G. Chryssolouris, "21st International Conference on Engineering Design, ICED17 // Multi criteria assembly line design and configuration – An automotive case study," pp. 69–87.

[3] F. Biesinger, D. Meike, B. Kraß, and M. Weyrich, "A Digital Twin for the Production Planning based on Cyber Physical Systems: A Case Study for a Cyber-Physical System based Creation of a Digital Twin," in *Conference Proceedings CIRP Conference on Intelligent Computation in Manufacturing*, 2018, pp. 355–360.

[4] W. Walla, "Standard- und Modulbasierte digitale Rohbauprozesskette: Frühzeitige Produktbeeinflussung bezüglich Produktionsanforderungen im Karosserierohbau der Automobilindustrie: Frühzeitige Produktbeeinflussung bezügliche Produktionsanforderungen im Karosseriebau der Automobilindustrie," Karlsruher Institut für Technologie, Karlsruhe, 2015.

[5] Stanev Stilian, "Methodik zur produktionsorientierten Produktanalyse für die Wiederverwendung von Produktionssystemen 2REUSE: Konzept, Informationsmodell und Validierung am besonderen Beispiel des Karosserierohbaus in der Automobilindustrie," Karlsruher Institut für Technologie, Karlsruhe, 1979.

[6] T. Bauernhansl, M. t. Hompel, and B. Vogel-Heuser, Eds., *Industrie 4.0 in Produktion, Automatisierung und Logistik: Anwendung, Technologien, Migration*. Wiesbaden: Springer Vieweg, 2014.

[7] U. Bracht, D. Geckler, and S. Wenzel, *Digitale Fabrik: Methoden und Praxisbeispiele*. Berlin u.a.: Springer, 2011.

[8] G. Michalos, A. Fysikopoulos, S. Makris, D. Mourtzis, and G. Chryssolouris, “Multi criteria assembly line design and configuration – An automotive case study,” *CIRP Journal of Manufacturing Science and Technology*, vol. 9, pp. 69–87, 2015.

[9] Ulrike Thomas, “Automatisierte Programmierung von Robotern für Montageaufgaben,” Dissertation, Technische Universität Braunschweig, Braunschweig, 2008.

[10] J. Michniewicz, G. Reinhart, and S. Boschert, “CAD-Based Automated Assembly Planning for Variable Products in Modular Production Systems,” *Procedia CIRP*, vol. 44, pp. 44–49, 2016.

[11] G. Levitin, J. Rubinovitz, and B. Shnits, “A genetic algorithm for robotic assembly line balancing,” *European Journal of Operational Research*, vol. 168, no. 3, pp. 811–825, 2006.

[12] T. C. Lopes *et al.,* “Balancing a robotic spot welding manufacturing line: An industrial case study,” *European Journal of Operational Research*, vol. 263, no. 3, pp. 1033–1048, 2017.

[13] Adalberto Sato Michels, Thiago Cantos Lopes, Celso Gustavo Stall Sikora, and Leandro Magatão, “The Robotic Assembly Line Design (RALD) problem: Model and case studies with practical extensions,” in *Computers and Industrial Engineering*, 2018, pp. 320–333.

[14] V. Agrawal, V. Kohli, and S. Gupta, “Computer aided robot selection: the multiple attribute decision making approach,” *International Journal of Production Research*, vol. 29, no. 8, pp. 1629–1644, 1991.

[15] N. Rishi Kanth, A. Srinath, and J. Suresh Kumar, “Selection of Industrial Robots for Automation Applications in Multiple Attribute Decision Making Environment using the Analytical Network Process,” *International Journal of Engineering &Technology*, vol. 2018, no. 7, pp. 392–402, 2018.

[16] A. Kusiak, “The production equipment requirements problem,” in *International Journal of Production Research*, pp. 319–325.

[17] A. Kusiak and S. S. Heragu, “KBSES: A knowledge-based system for equipment selection,” *Int J Adv Manuf Technol*, vol. 3, no. 3, pp. 97–109, 1988.

[18] J. Merker, *Heuristiken in der Layoutplanung: Graphentheoretische Verfahren für das Nachbarschaftsproblem*. Wiesbaden, s.l.: Deutscher Universitätsverlag, 1998.

[19] T. C. Lueth, “Automated Planning of Robot Workcell Layouts,” *IEEE International Conference on Robotics and Automation*, 1992.

[20] A. Drira, H. Pierreval, and S. Hajri-Gabouj, “Facility layout problems: A survey,” *Annual Reviews in Control*, vol. 31, no. 2, pp. 255–267, 2007.

[21] S. J. Russell and P. Norvig, *Künstliche Intelligenz: Ein moderner Ansatz,* 3rd ed. München u.a.: Pearson Higher Education, 2012.

[22] Michael I. Jordan, “Why the logistic function? A tutorial discussion on probabilities and neural networks,” in *Computational Cognitive Science Technical Report*, pp. 1–13.

[23] Andrew Ng, *Hyperparameter Tuning*. Stanford University, 2019.