Proceedings of the 2022 Winter Simulation Conference
B. Feng, G. Pedrielli, Y. Peng, S. Shashaani, E. Song, C.G. Corlu, L.H. Lee, E.P. Chew, T. Roeder, and P. Lendermann, eds.

USING DATA FARMING AND MACHINE LEARNING TO REDUCE RESPONSE TIME FOR THE USER

Falk Stefan Pappert
Oliver Rose

Department of Computer Science
Universität der Bundeswehr München
Werner-Heisenberg-Weg 39
Neubiberg, 85577, GERMANY

ABSTRACT

Simulation in our area usually takes some time; even if a preexisting model just needs to be parameterized there is still the run time, which will usually take at least a few minutes if not hours. In our current case, a planner wanted to know for a given product mix situation and for an equipment group with specific characteristics how much he can utilize the equipment without violating flow factor targets. Since the user is usually asking the same question just with different parameters we are able to solve the waiting time problem while still giving good decision support. Instead of simulating every scenario at the time the user actually needs these answers, we use data farming to generate a large set of data points that are then used to train a neural network. This neural network substitutes for the simulation and responds to the user immediately.

1 INTRODUCTION

Capacity planning is a crucial task in industry. Robinson, Fowler, and Neacy (2003) point why accurate capacity planning is important and yet difficult to achieve in the highly sophisticated semiconductor industry. A planner faces numerous questions every day, being it short term operative question or long term strategic ones considering future investments. A necessary starting point to be able to answer these questions is knowledge about the available equipment capacity and how its utilization will influence the material flow. Flow factors are used as a key performance indicator for the whole production system as well as smaller subsections. They are calculated as

$$flow\ factor = \frac{actual\ cycle\ time}{raw\ process\ time}.$$

Flow factors are a good performance indicator to evaluate a system, especially when producing many different products with many different cycle times. In semiconductor industries it has become good practice to aim for certain flow factors as a target of control. Having shorter production cycle times, and therefore more production cycles, is highly important in the semiconductor industry, as it allows additional development cycles for new and improved products - although this may decrease throughput. Also, manufacturing products quickly is preferable as this provides additional flexibility and shorter delivery times to customers. Flow factors are dependent on the system itself and its utilization. Operating curves show this relation for a system, see Fayed and Dunnigan (2007) and Aurand and Miller (1997). An operating curve of a simple system can be seen in Figure 1. Usually, there is an almost flat area where production is quite robust, with a steep rise the closer utilization gets to 100%. At what point this rise starts and how steep it is very dependent on the system at hand. Other characteristics can also change the operating curve of a system. The operating curve of another still simple system is shown in Figure 2.

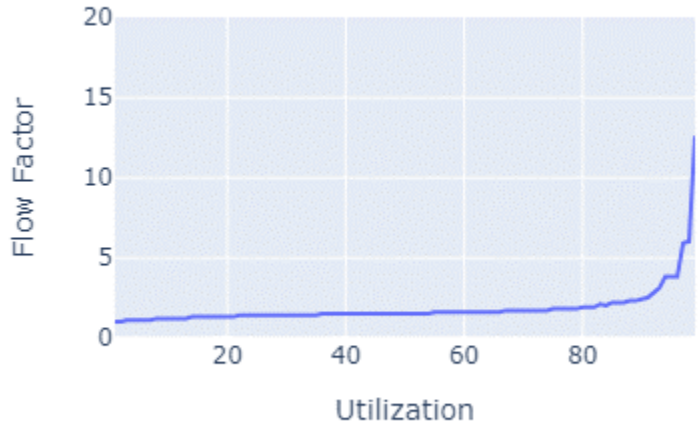

Figure 1: Operating curve of a simple equipment group without any special features.

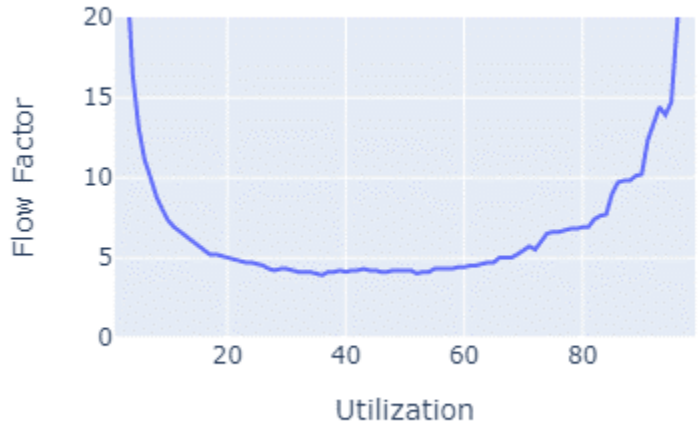

Figure 2: Operating curve of an equipment group with a single equipment, batching and rare, but long, breakdowns.

When comparing both systems operating curve, the most obvious change is the fall on the left side of the second operating curve. This will typically be the case when looking at systems using some form of batching. But there are also other differences, while the system represented by Figure 1 is relatively stable until about 95% utilization we already see a steeper rise of the flow factor much earlier in the second system. What is also quite interesting with the second system is the minimal possible flow factor. As we can see, it is not possible to have the flow factor better than approximately 5, regardless of the utilization point we would be running the system at. So, when trying to run the system at a lower flow factor, changing

utilization will have no sufficient effect. One would have to change the systems characteristics, e.g. change the number of machines or lax batching rules.

When doing production planning constrained by desired flow factor values, it is important to first see if these flow factor values are possible. If they are achievable, it is important to know how high a system can be utilized without having to expect the flow factor target being violated. Hence, what our planner needs is a way to determine the lowest possible flow factor of a given system, or at least if a given flow factor is possible within that system. Additionally, the planner needs the utilization values which present the limits under which a flow factor can still be achieved.

These two points of information limit the available capacity of the equipment for planning if flow factor targets should not be violated. Although most equipment groups behave quite differently, based on their characteristics, the underlying question stays the same. Traditionally, lookup tables were used to get these information. Although reasonably fast, they lacked consideration of several important equipment characteristics and therefore tended to be inaccurate. Simulating the system at several utilization points to generate an operating curve could be a solution, but simulation typically takes a lot of time. There are approaches to reduce the time to calculate operating curves by using only a few points (Byrne 2012). This would still involve a lot of simulation effort, so a response would take some time. Our goal is to reduce this time to an instant, so the planner does not need to wait for a result. At the same time, our system should provide the planner with reliable information for the planning decisions.

As traditional discrete event simulation seems to be to slow for the purpose at hand, a promising approach to replicate behavior within a manufacturing context are artificial neural networks. Bergmann et al. (2014) show the use of neural networks to replicate of manufacturing control strategies. Wörrlein et al. (2019) discuss the application of neural networks to predict energy consumption of production equipment. Detailed information on the implementation and technical realization of the system can be found in Pappert and Rose (2021).

In the following section we will give a general overview of our approach. In Section 3 we discuss the considered features in more detail, followed by an introduction to the model and evaluation, the training of the neural network and results. Finally, in Section 8, we will take a look at future steps.

2 APPROACH

In planning, some questions arise repeatedly during a common work day – and waiting for an answer would slow down work significantly These general questions, and the space of possible instances we could be asked for, are known. We aim to build a system that is able to answer very similar questions with a response time that is as low as possible while still providing sufficient answer quality to be useful.

A common approach to analyzing an equipment group would be to build a simulation model, include all its features and run some simulations, analyze the results and supply the answers to the user. Because of time criticality we have to change this common approach. The quality of the simulation response is acceptable for our purpose, just the time to get it needs to be reduced as much as possible.

In Figure 3 we give an overview on the approach we are using to solve this runtime issue.

The traditional approach of tackling a problem with simulation is shown in the “challenge”-column: i.e. design and implement a model, gather statistics and analyze the results. This approach can be speed up slightly by providing a general model, that only needs to be parameterized and run by the user. Then, most of the model building time would be moved to the time of building the system, when run time is not that important. As running automatically generated models based on parameter sets still takes some time, at least when looking at larger equipment groups with higher throughputs, automated model building can just be a first step.

The step towards very fast response times is to not only put the model building to a point in time before the application phase, but also move the running of the simulation before the application phase. With only a few possible combinations, it would be quite simple to prepare all possible system configurations, and look up the result from this data set. In our case the problem space is enormous, therefore running all

theoretically possible - or even practically feasible - configurations is not preparable in that way. What we can do is prepare a lot of data points and infer the areas between our prepared points. And this is

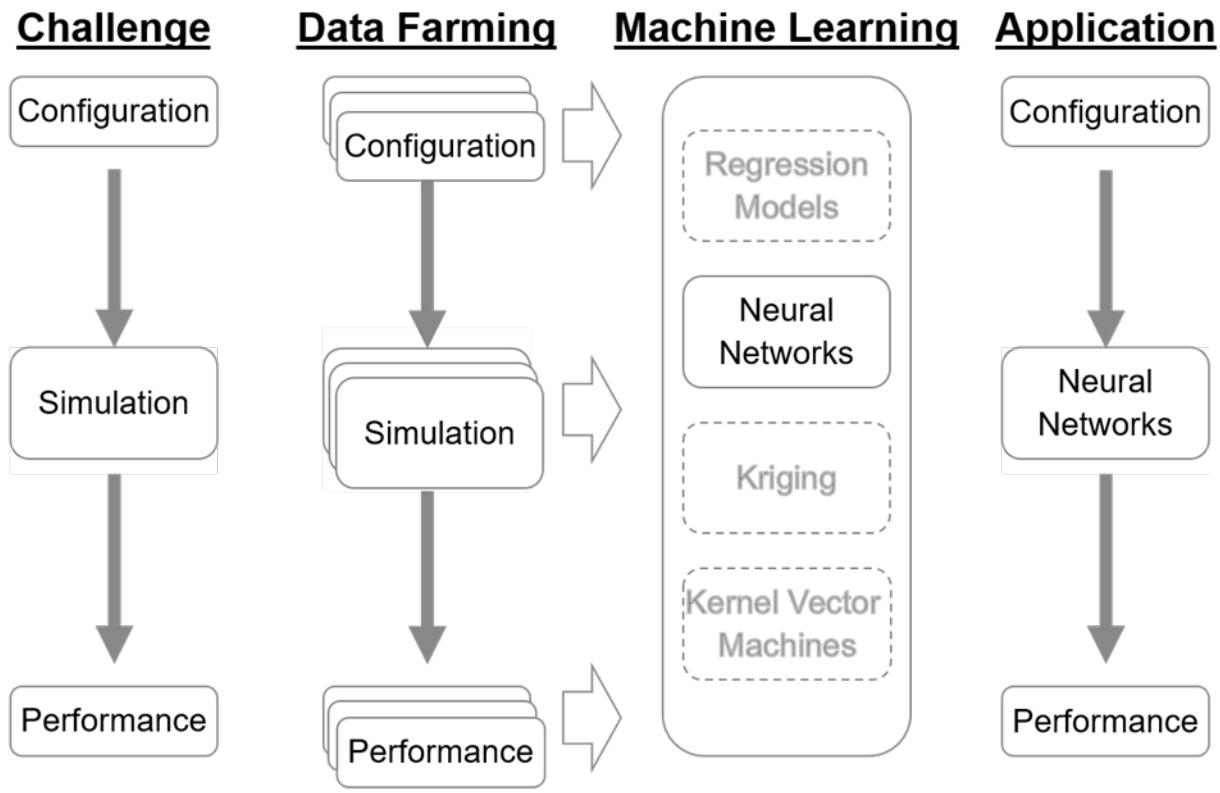

Figure 3: Overview of the general idea.

what our approach boils down to: We run data farming for a large number of supporting points to capture the problem space. Then we use these calculated data points to train a neural network. If the training is successful, the neural network can immediately respond to the user. Simulation and training may take a long time, but during the actual application, the response time of the system to user's requests will be very short.

3 FEATURES CONSIDERED

In this paper we use the terms feature, factor and level to describe characteristics of the equipment model. With feature we describe a general characteristic that is considered within the model, like batching, dedication or the number of tools in an equipment group. A features' implementation is represented by a number of factors. For batching this would be the maximum batch size, minimum batch percentage, and maximum waiting time. With levels we describe the actual instance value of the factor used in a given scenario or design point. In the example of "equipment count", the levels could be 1, 10 or 15.

Starting with Robinson et al. (2003), Hopp and Spearman (1996) and a review of the previous planning methods we defined the relevant features for our equipment group model. Values to represent our levels are based on a fab dataset of our industrial partner, where we looked for natural clusters and selected representatives of real instances to capture realistic working points. Table 1 gives an overview on our currently considered features and factors and their number of levels.

Initially each of the features we planned to investigate were represented by a single factor (Pappert et al. 2017). For features like the number of equipment in the group this is simply done as a numerical factor

Table 1: Overview on features, factors and levels.

Feature	Factor	# Levels	Type
Batching	MaxBatch	5	Quantitative
	MinBatchPercentage	3	Quantitative
	MaxWaitTime	1	Quantitative
Breakdown	BreakdownCycleLength	5	Quantitative
	BreakdownCapaLoss%	2	Quantitative
Dedication	Dedication	4	Categorical
Equipment #	ToolCount	6	Quantitative
Maintenance	MaintenanceCycleLength	3	Quantitative
	MaintCapaLoss%	2	Quantitative
Product Mix	ProductMix	3	Categorical
Rework	ReworkPercentage	3	Quantitative
Process Time	RPT	6	Quantitative
Setup	SetupDuration	3	Quantitative

with levels defining the values to be used in the simulation model. For more complex features we started with categorical levels, and Defined parameter sets for each level. The problem with having categorical levels defined in this way is the lack of a general way to find points between two levels. With our final system we aim to answer general questions on equipment groups and not only for cases that luckily fit exactly our precalculated scenarios. So instead of this one factor representing batching, we split them; For example in the case of batching, the categorical parameter of the batch [max=300, min =270, wait=12h] was split into three: the maximum batch size (300) defining the capacity of an equipment, a minimum batch percentage (90%) threshold under which processing a batch may not be economically viable, and the wait time (12h) to balance flow factor against economic viability for low volume products. With this change, each factor becomes numerical, which makes it easy to find additional valid points between two factor levels.

When splitting features into multiple factors, we aim for each factor representing a different aspect of the feature and try to keep sets of factors as generic and meaningful as possible. . Again, looking at batching, we moved from a fixed amount for the minimum to a percentage of the maximum. Not only does this simplify finding valid factor level combinations, as in all valid level combinations the maximum needs to be at least as large as the minimum. It also makes comparing combinations a lot easier, and thereby is offering us the opportunity to analyze our resulting dataset with regard to ideas on the performance influence of certain features. We also think this will provide us with a clearer idea of the cause for seen changes in performance. This differentiation into aspects of a feature can also be shown with the feature of breakdowns. Here we moved from a single factor with, each level representing a combination of a fixed value for mean time between failure (MTBF) and mean time to repair (MTTR), to two factors: one representing the cycle length of a breakdown cycle, the other the capacity loss due to the breakdown. This enables us not only to see the impact of fixed values on the systems performance, but see the impact of these concepts more clearly. With the change, we can now simply compare systems with different capacity losses at the same cycle length, or systems with the same capacity loss at different cycle lengths instead of looking at the feature as a whole. In the old system, a MTTR value almost exclusively had meaning in relation to its MTBF value. A MTTR of 2h would not tell us much by itself without having an idea of the MTBF. Breakdowns are also a good example of these aspects showing an effect on results. Imagine two systems with a similar amount of capacity loss, e.g. 20% due to breakdowns. One system breaking down every 4 hours for an hour, compared to a system with the same capacity loss, with breakdowns only happening every 4 weeks but lasting a whole week. The impact of the cycle length becomes much easier apparent compared to our previous approach.

In this paper we are representing all but two features with only quantitative factors: The remaining two features are the product mix and dedication. The challenge with these features is finding good numerical representations to capture the full complexity of the feature, while remaining individually meaningful. For product mix a good factor could be the number of products, but we are still working on an elegant way to describe the balance between the different volumes of the products.

4 THE MODEL

The simulation model is built in a factory simulator we created in-house. The simulator is written in java and provides us with the opportunity to adapt every aspect. The model itself is a configurable model of an equipment group. The model configuration depends on the factor setting of a given scenario. Different parts of the model are responsible for implementing the defined features of a scenarios equipment group. Figure 4 is giving a general overview of the model.

For each product there is a source, that generates lots with 24 wafers at a defined arrival rate. The rate is based on the utilization point and the percentage part of the product generated in the product mix.

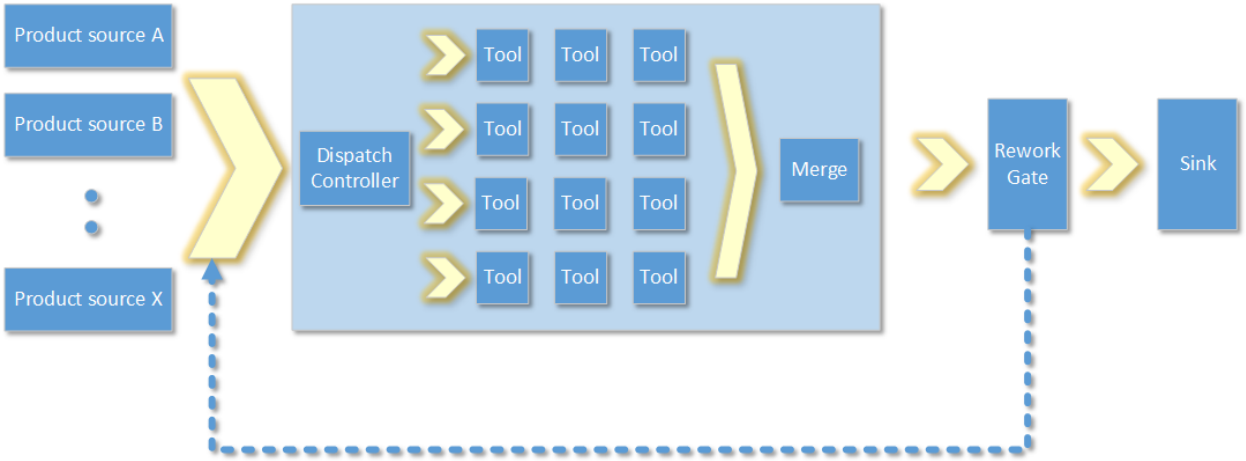

Figure 4: Model overview.

From the source, lots move to a queue controlled by a dispatching controller. This controller decides for each lot when and to which equipment to go. In the current setup of the model lots are basically dispatched using FIFO (First in first out), when possible. Since some of our models consider dedication, the controller is also responsible for sending only lots to an equipment that is defined as being able to handle the given product. Equally, for scenarios requiring different setups for different products, the controller decides when a product change should be done. The logic to change a current setup is enforcing a setup to be used for a minimum number of lots and then allowing to switch to another setup based on the current situation in the queue. We use this basic setup rule, as it needs to work reasonably well with small and large equipment numbers. Furthermore, it works with only a few products as well as many and on a wide range of utilization points. This approach does not perfectly balance capacity loss, due to setup times and the wish for small flow factors, but it is easily applicable to all scenarios. The last feature the controller implements is batching. Our batching considers three parameters: a maximum batch size, a minimum batch size and a maximal waiting time. A batch is only send if at least a minimal number of wafers participate in the batch. Since some of our scenarios have very unbalanced product mixes, we define a maximal waiting time after which a batch can be started even though the minimum threshold has not yet been reached.

From the queue, the lots or batches are send to a specific equipment dependent on the before mentioned decisions by the controller. The equipment does not differentiate between lots and batches and will simply process the arriving group of wafers as one. If the previous wafers needed a different setup state the

equipment will automatically change its setup. This is because the decision to change the setup state is implicit in the controller sending wafers of another product to this equipment. A feature directly handled by the equipment is the process duration, which is modelled as a fixed time, depending on the scenario and a small distributed offset to account for slight changes in processing time or an operator not being able to pick up material immediately. Capacity losses due to breakdowns and maintenance are also modelled at the equipment. Following the equipment a merge block will undo batching to regroup the wafers into their original lots. A rework gate is used to randomly send lots back into the queue of the dispatch controller. This models a test step which may send a certain percentage of lots for rework on an equipment using additional capacity.

Once the lots have successfully passed the rework gate they are processed for data collection in a sink.

5 EVALUATION AND SIMULATION

With the generic model described in Section 4 we could automatically generate a model representing a single point in the parameter space of our factor levels. But before we can actually run a simulation, we need to know the load required to reach a targeted utilization point. With the large parameter space we are looking at, it would not be feasible to simply try different loading amounts, and find good loading amounts iteratively. Reasonable loading heavily depends on the capacity of a given scenario. Think about a scenario with only a single machine, that can process a single lot at a time within an hour. This would leave us with an equipment group being utilized at 100% with just 24 lots per day. On the other end of the spectrum we are looking at equipment groups with 20 machines, running possible batches of up to 300 wafers and only needing 15 minutes per batch. For this equipment group 100% utilization would mean 24000 lots per day. Simply trying different loading points would cause us to waste quite a number of simulation runs just to find a good starting point. The load of the simulated model has a big impact on run time, and overloading will slow down the simulation speed even more. With more work in process (WIP) in the model, and longer queues, simulations tend to take longer. So overshooting, while trying to find a starting point of the model, gets even more expensive in computing time than running the scenario at a good utilization point.

To solve this issue we do a static capacity analysis when starting to analyze a new scenario. The impact on capacity of most features is fairly easy to calculate, as we have done with the short calculation of our example above. Looking at number of equipment, batch size and the processing time for a single production run determines the theoretical capacity limit. From this we have to subtract capacity losses of other features like breakdowns. The capacity loss due to a feature like setup is very hard to predict, as this is heavily dependent on the exact rule followed to issue setups. Two things played into our hand when trying to handle this issue, both coming from the application of the system. First, there are not that many types of equipment that need a consideration of setups. Therefore the impact of model inaccuracies does not invalidate the approach for the largest portion of the equipment park. Second, planners at our partner usually don't consider capacity loss due to setup when estimating utilization for similar reasons. Thus the capacity loss and the dynamic effects of setups are modeled in the system, and the returned values fulfill the expected purpose since their meaning is the same in the model and for the user of the system.

Based on the calculated capacity, we can calculate the needed load to reach specific utilizations of the model. And with these, we can start looking for interesting points in the operating curve. The first point we look for is the utilization point with the minimum possible flow factor. With this point found, we know if the FF thresholds we look for are possible with the current scenario. Additionally, we can limit our search for the thresholds to the right part of the operating curve. Although the flow factor may rise with lower utilizations, for example due to batching, we are usually aiming to utilize equipment as high as possible. An equipment group facing cycle time issues due to batches not filling up fast enough is not a hard limitation of the available equipment capacity, but more of a control issue.

After finding the point with the lowest flow factor a search for the thresholds is done. Each of these searches uses a search strategy akin to binary search. To reduce the number of unnecessarily calculated

points, all previously calculated points for a scenario are used to find the best next point to evaluate in the search for a threshold or the lowest point.

As we do have statistical influence in our model, we need to consider several simulation replications for a single utilization point. The number of replications per utilization point is determined on the fly during the evaluation, to avoid unnecessary simulation runs. For all evaluated utilization points we first run a small set of replications. Based on these, we calculate the half length of the confidence interval and mean and compare their quotient with the relative error we aim for, similarly to Law and Kelton (2000). If the error is still larger we start another set of replications. This is done until the relative error is smaller than the quotient.

On average this has led to 825 simulation runs to determine the location of the lowest point and 3 thresholds for a scenario. Based on our old single factor per feature approach we were looking at 460000 data points just to generate the supporting data for our training. Which meant about 380 million simulation runs. While evaluating single design points on a normal office PC is still feasible, it took our server several weeks just for this data set. With the switch to multi factor features, we are able to continue using this data set as a base set. Additionally, we get a lot of new level combinations. This change gives us the opportunity to address many more data points as were previously possible With the increasing number of design points we cannot simply calculate all points, but have to choose the most valuable ones for our training. We have to decide which areas of our design are represented sufficiently and which areas may benefit from further supporting points.

6 TRAINING

With the numerous data points we generate a trained neural network. In the beginning we were using neural networks in R, for more information on R see Verzani (2014). But training took quite long, usually hours or even days. With larger networks we got to a point, where training would run into to memory issues and the training basically stopped working reliably. To tackle these issues we decided to switch our training environment to Keras (Géron 2019) using the Tensorflow library (Abadi et al. 2016). With this change we were able to utilize two major benefits. First, changing the environment from R to Python allowed us to use a lot of other libraries from the quite active machine learning community in Python. The second big advantage came with the option to use graphic processors for training. So, without much additional effort beyond rewriting our training script in Python, we reduced training times to the area of seconds and minutes. A big benefit of these highly reduced training times is the opportunity to test a lot more network configurations to find good architectures working well for our topic.

When training, we generally aim to reduce mean squared error (MSE). Although this is able to compare the success of different training runs it does not tell much about the applicability of our results for the actual use case. To evaluate this, we defined a histogram showing training results graphically: it shows us the distance of our test set from the actual simulated values. These graphs are a representation of the training's success and help us by enabling us to immediately evaluate a network with regard to its applicability in practice. What we want to get from the trained network is the utilization threshold for a targeted flow factor for a given equipment configuration. So what we expect is a value between 0 and 100 representing utilization in percent. On the x-axis we display the distance from the prediction of the trained network to the one from the simulation. Results are put into bins of 1%. So the first bar represents up to 1% deviation, the second bin up to 2% deviation and so on. On the y-axis we count the number of test scenarios that have a specific deviation. An ideal result would have a very large bar at 0 and show no additional data points.

Figure 5 is an example of one resulting graph, from a – as we thought at the time – quite successfully trained network compared to other networks based on MSE. Looking at the training results in this representation easily shows that the results would not be usable at all. some deviation occurring in few cases might be acceptable. But a deviation occurring regularly of 10% to 20% would make the prediction not usable for any planning or investment decision, let alone the predictions with more than 60% deviation.

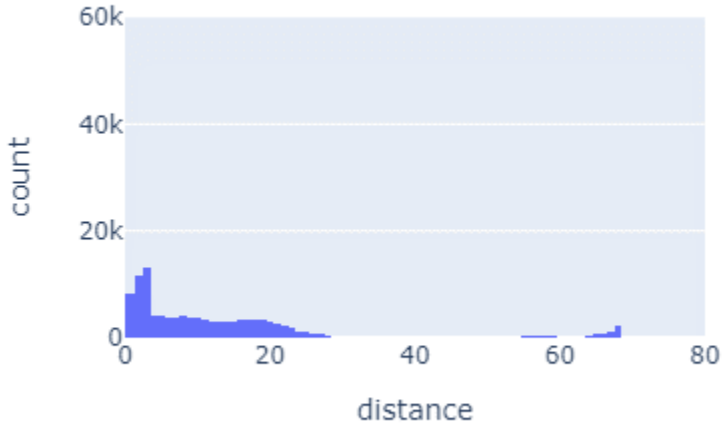

Figure 5: Training result graph example.

Although we managed to reduce training times with our switch to Keras, finding good network architectures has still been a very labor intensive iterative process, which beckons the question of automation. Our manual results before starting the search for automation can be seen in Figure 6. Almost half of the test set is perfectly predicted and about a third is at most 2% off. This leaves about a sixth of the test cases that are not predicted as good as we would like.

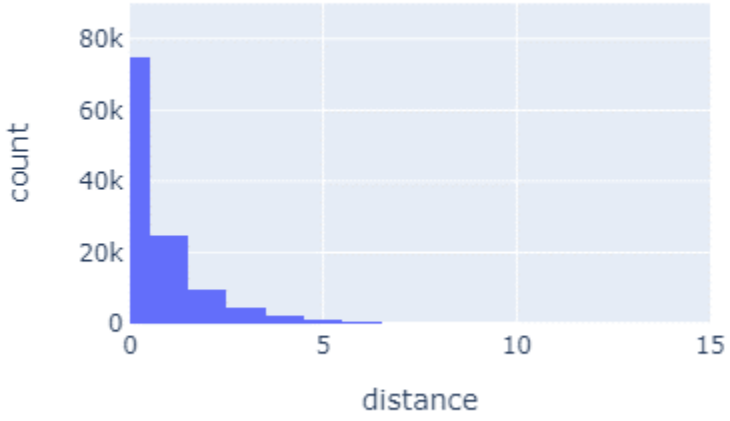

Figure 6: Manual training results.

We started exploring two approaches of automating this process. The first approach we looked at was neuro-evolution. Here, the basic idea is to have an evolutionary algorithm do the parameter search for our network architecture search. More information on this approach can be found in Stanley et al. (2019).

Although being a promising approach at first, results differed from run to run and we were not able to even replicate the quality of results from our manual process. Of course, there are many ways and parameters for this approach to improve search results and to adjust for a given problem. But this just seemed to move the issue of the parameter search one meta level higher, without improving our initial problem.

The second approach we looked at for automating the network search is using AutoKeras. AutoKeras is an AutoML system based on Keras introduced in Jin et al. (2019). The goal is to automate most steps getting from the data to the trained model. First results are quite promising as shown in Figure 7. We are looking to further improve our previous results. As AutoML systems provide a large set of parameter options we see the opportunity of future improvements.

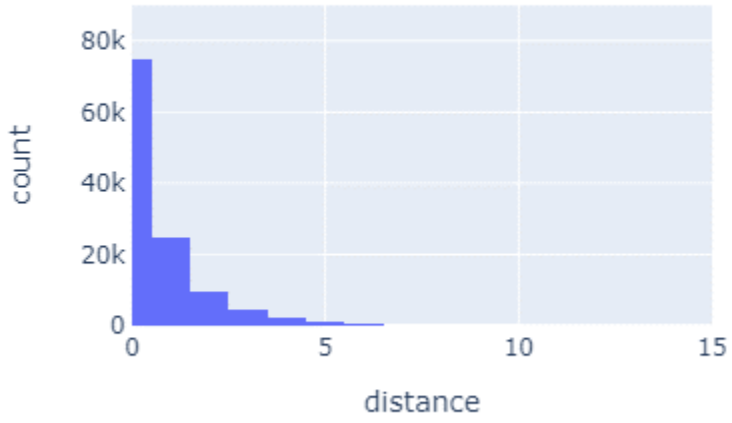

Figure 7: Training results using AutoKeras.

7 RESULTS

With our research, we are trying to reach three goals. The first is to be able to use the compiled data for future analyses, to better understand the effects of different factors and their combinations on an equipment group. With our changes to the representation of features and the way we are splitting features into factors with independent meaning, we have taken a large step in this direction and are now looking at a growing data set we can use for future analysis.

The major goals of our research have been to provide the planners with sufficiently high quality responses to their questions within a very short time frame. We can reach the goal of very short response times with our current approach. Querying the trained network for a response takes less than a second on an office computer with a modern graphics card. Even for systems that cannot support modern graphics cards, the trained network could be put on specialized hardware that can store a trained network and be easily plugged in at an USB port for a quite low cost. The goal of sufficiently good responses is still not entirely reached. Although our predictions are better than the historical system, we still see some room for improvement.

8 OUTLOOK

As we have mentioned in the last section we are already very happy with the response time of our system and do not see much need to improve on its current performance. Still, there are several things where we

still see room for future research. Although we already get quite good results we would like to improve the result quality. To this end, we see several paths to be taken in the future.

First and foremost is an analysis of the points we could not predict very well. This analyses comes with two purposes. First, from a practical point, it is important to know if the cases that are hard to predict represent cases that are likely to be seen in a real fab situation. With our huge space of possible equipment group configurations not all configurations are similarly likely to be considered. If these cases are unlikely fringe cases, the usefulness of the (prediction) system would be reduced much less than if they are highly likely or even currently existing/ used in the fab of our industrial partner. Second is information on whether there is any form of systematic issue that causes prediction problems for a certain kind of configurations. Finding a systematic issue may help us remove the systematic issue and thereby improve prediction quality significantly.

There are two general options to improve the system. The first option is to improve the data base used for training. The simplest way, just adding more and more levels and train with an even larger design, has very hard limits that we have nearly reached. Another way would be to find areas with less than ideal prediction and add very few additional points in these regions with the hope of a local improvement.

The second option is to improve responses by having a more successful training approach. As we have mentioned before there are numerous parameters for AutoKeras that could be tweaked, and there are other approaches to find good network layouts and training parameter sets that may fit better to our problem.

ACKNOWLEDGMENTS

We would like to thank Heiderose Stein for her valuable and constructive suggestions during the writing of this paper.

REFERENCES

Abadi, M., P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng. 2016. "TensorFlow: a system for Large-Scale machine learning". In *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, 265–283. Savannah, GA: USENIX Association.

Aurand, S. S., and P. J. Miller. 1997. "The operating curve: a method to measure and benchmark manufacturing line productivity". In *1997 IEEE/SEMI Advanced Semiconductor Manufacturing Conference and Workshop ASMC 97 Proceedings*, 391–397. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Bergmann, S., S. Stelzer, and S. Strassburger. 2014. "On the use of artificial neural networks in simulation-based manufacturing control". *Journal of Simulation* 8(1):76–90.

Byrne, N. M. 2012. *A framework for generating operational characteristic curves for semiconductor manufacturing systems using flexible and reusable discrete event simulations*. Ph.D. thesis, Dublin City University. https://doras.dcu.ie/16922/1/PhDThesisNeillByrneBoundVersion20120427.pdf.

Fayed, A., and B. Dunnigan. 2007. "Characterizing the Operating Curve—how can semiconductor fabs grade themselves?". In *2007 International Symposium on Semiconductor Manufacturing*, 1–4. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Géron, A. 2019. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*. 2nd ed. Sebastopol: " O'Reilly Media, Inc.".

Hopp, and Spearman. 1996. *Factory Physics*. 3rd ed. New York: McGraw-Hill.

Jin, H., Q. Song, and X. Hu. 2019. "Auto-keras: An efficient neural architecture search system". In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 1946–1956. New York, USA: Association for Computing Machinery.

Law, A. M., and W. D. Kelton. 2000. *Simulation Modeling & Analysis*. 3rd ed. New York: McGraw-Hill, Inc.

Pappert, F. S., and O. Rose. 2021. "Reducing Response Time with Data Farming and Machine Learning.". *Simulation Notes Europe* 31(2):87–94.

Pappert, F. S., O. Rose, F. Suhrke, and J. Mager. 2017. "Simulation based approach to calculate utilization limits in Opto semiconductor frontends". In *2017 Winter Simulation Conference (WSC)*, 3888–3898. Institute of Electrical and Electronics Engineers, Inc.

Robinson, J., J. Fowler, and E. Neacy. 2003. "Capacity Loss Factors in Semiconductor Manufacturing". http://www.fabtime.com/abs_CapPlan.shtml, accessed 27.04.2022.

Stanley, K. O., J. Clune, J. Lehman, and R. Miikkulainen. 2019. "Designing neural networks through neuroevolution". *Nature Machine Intelligence* 1(1):24–35.

Verzani, J. 2014. *Using R for introductory statistics*. 2nd ed. CRC Press.

Wörrlein, B., S. Bergmann, N. Feldkamp, S. Straßburger, M. Putz, and A. Schlegel. 2019. "Deep-Learning-basierte Prognose von Stromverbrauch für die hybride Simulation". *Simulation in Produktion und Logistik 2019*:121–131.

AUTHOR BIOGRAPHIES

FALK STEFAN PAPPERT is Research Assistant and PhD student at Universität der Bundeswehr as a member of the scientific staff of Prof. Dr. Oliver Rose at the Chair of Modeling and Simulation. His focus is on conceptual modelling approaches to simulation-based scheduling and optimization of production systems. He has received his M.S. degree in Computer Science from Dresden University of Technology. He is a member of GI. His email address is falk.pappert@unibw.de.

OLIVER ROSE holds the Chair for Modeling and Simulation at the Department of Computer Science of the Universität der Bundeswehr Munich, Germany. He received a M.S. degree in applied mathematics and a Ph.D. degree in computer science from Würzburg University, Germany. His research focuses on the operational modeling, analysis and material flow control of complex manufacturing facilities, in particular, semiconductor factories. He is a member of INFORMS Simulation Society, ASIM, and GI. His email address is oliver.rose@unibw.de.