

AUTOMATED NEURAL ARCHITECTURE SEARCH FOR SIMULATION METAMODELLING OF CONWIP SYSTEMS

Mozolova, L.; Grznar, P.; Mozol, S. & Gregor, M.

Department of Industrial Engineering, Faculty of Mechanical Engineering, University of Žilina, Slovakia

E-Mail: lucia.mozolova@fstroj.uniza.sk, patrik.grznar@fstroj.uniza.sk, stefan.mozol@fstroj.uniza.sk, milan.gregor@fstroj.uniza.sk

Abstract

In this article, we present a methodology that enables the automatic creation of metamodels based on artificial neural networks. We use a neural network architecture search technique, known as NAS. Work in this area so far has mostly relied on manual design of the network architecture, which is time-consuming and the results depend on the researcher's experience. Our framework combines several interconnected steps: generating data using Latin Hypercube Sampling, automatically checking its quality, finding the best architecture using the Tree-structured Parzen Estimator (TPE) algorithm, and finally building a set of models that work together to make the final prediction. We have verified the entire procedure on a real production line controlled by the Constant Work-In-Progress (CONWIP) system. We achieved a prediction accuracy of $R^2 = 0.9955$ and an average percentage error of 1.34 %. Compared to direct simulation, our metamodel is 312 times faster, enabling optimization and scenario analysis directly during decision-making.

(Received in January 2026, accepted in April 2026. This paper was with the authors 1 week for 2 revisions.)

Key Words: Discrete Simulation, Neural Metamodel, Network Architecture Search, CONWIP, Multi-Criteria Optimization

1. INTRODUCTION

Today, manufacturing companies face strong competitive pressure. Markets are changing rapidly, customers expect short lead times, and the ability to react flexibly has become key [1, 2]. That is why the importance of tools that help to understand the behaviour of a production system under different conditions is growing. Discrete simulation allows for the creation of a virtual model of the production line, entering various parameters and monitoring the results without interfering with the actual production [3]. The problem arises with optimization – algorithms need to try thousands of combinations of parameters. If one run lasts a minute, a million runs last almost two years [4]. Added to this is the increasing complexity of production systems using advanced automation [5, 6], producing hundreds of products [7], with heterogeneous workers whose skill distribution affects line performance [8] and requiring sophisticated inventory management [9]. The solution is metamodels – simpler mathematical models that approximate the behaviour of the simulation. After training, they can estimate the results almost immediately [10]. Among the most successful are artificial neural networks capable of capturing complex nonlinear relationships [11, 12]. The problem remains the design of the network architecture – how many layers, neurons, what activation function. Traditionally, this has been solved by trial and error, which is lengthy and depends on experience [13]. The new Neural Architecture Search approach lets the algorithm automatically find the best architecture [14]. Modern methods such as TPE significantly speed up this process [15]. In this article, we present a complete procedure that combines data generation, quality control, automatic architecture search and the creation of a robust metamodel. The paper presents an integrated methodology that combines Latin Hypercube Sampling and data validation using Neural Architecture Search (NAS) to create a metamodel composed of ten networks. This verification approach achieved an accuracy of $R^2 = 0.9955$

and demonstrated a 312-fold acceleration compared to the conventional simulation. The idea of replacing a complex simulation model with a simpler mathematical expression is not new [10]. Barton laid the theoretical foundations in 1992 [16], Kleijnen elaborated on the methodological aspects of the experimental design [10]. When choosing a metamodeling technique, the number of parameters and the complexity of the relationships matter [17, 18]. Polynomial regression is interpretable but limited. Kriging offers an estimate of uncertainty, but it scales poorly. Neural networks provide flexible approximation with good scalability. Hurion used a neural network in 1997 to optimize a kanban system [19]. Fonseca and colleagues summarised the findings into practical recommendations [11]. Sabuncuoglu and Touhami have shown that networks outperform classical methods in nonlinear relationships [12]. Santos and Santos addressed efficient data collection [20]. Despite the advances, the question of the optimal architecture remained unanswered – most authors experimented with several configurations, which is unsystematic and time-consuming. In 2017, Zoph and Le introduced Neural Architecture Search – instead of manually designing the architecture, an algorithm searches for the architecture [14]. This area is developing rapidly, as documented by a review study [21]. Pham and colleagues proposed a more efficient method using weight sharing [15]. For practical applications, Optuna implementing the TPE algorithm has proven itself [22]. TPE builds models for both good and bad configurations and designs new ones based on them. Recent extensions such as sample-based dynamic hierarchical transformers combined with contextual bandits further improve adaptive architecture selection in deep learning systems [23]. The application of NAS in production simulation is a new field – our paper is one of the first to systematically apply these techniques to metamodels of production systems.

2. METHODOLOGY

The methodology consists of five main steps. These steps follow each other, but at the same time contain feedback – if we find a problem in any step, we can return to the previous one. The whole procedure is shown in Fig. 1.

In the first step, we create a simulation model of the investigated production system and define the inputs and outputs of the metamodel. In the second step, we generate data using Latin Hypercube Sampling. In the third step, we check the quality of the data with seven statistical tests. In the fourth step, we look for the optimal architecture using the TPE algorithm. In the fifth step, we create the final metamodel as a set of ten networks.

2.1 Data generation with Latin Hypercube Sampling

The quality of a metamodel depends primarily on the quality of the data on which it has been trained. We need data that covers the full range of possible conditions. At the same time, we want this data to be obtained efficiently [24]. Latin Hypercube Sampling is a technique that meets these requirements. It works by dividing the range of each parameter into equal intervals and ensuring that there is exactly one sample in each interval. Unlike random sampling, there are no clusters or empty areas.

The simulation model is built on the SimPy framework in Python. For each combination of parameters, we run the simulation several times with a warm-up phase to obtain steady values of the indicators and eliminate random fluctuations.

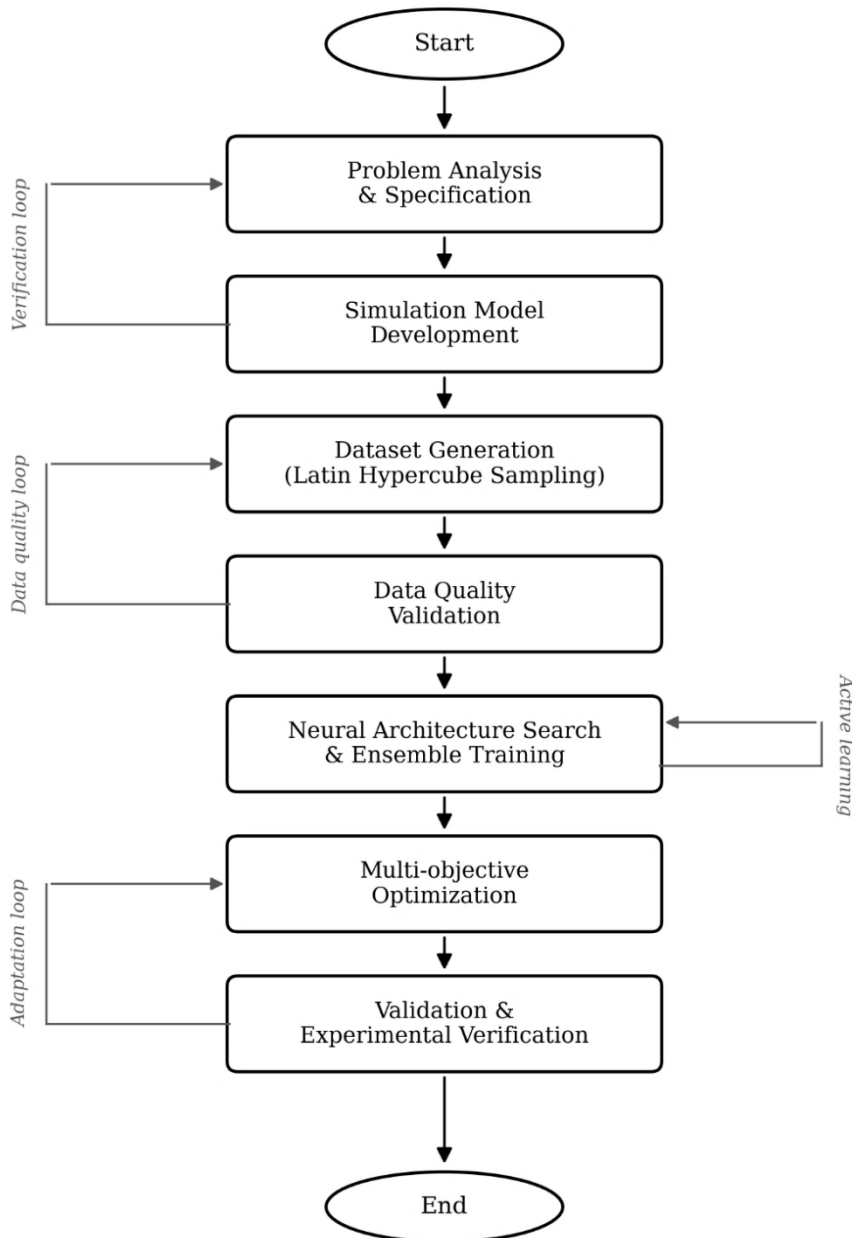


Figure 1: Overview of the proposed methodology for automated ANN metamodel development.

2.2 Data quality control

Before we let the data into the process of finding an architecture, we subject it to a series of checks. We defined seven tests: sample size (min. 95 % of valid values), output variability (coefficient of variation > 0.3), input-output correlation (mean > 0.1), outlier detection (max. 5 %), input coverage, multicollinearity ($VIF < 5$), and normality of distribution.

2.3 Network architecture search

We use the TPE algorithm from the Optuna library [22]. The search space includes: number of hidden layers (2–6), number of neurons per layer (128–1024), activation function (ReLU, Swish, GELU), dropout rate (0–0.3), learning rate (0.0001–0.01), and optimizer (Adam, Nadam, AdamW). The number of hidden layers was not fixed a priori; it was included in the NAS search space in the range of 2 to 6 layers, and the TPE algorithm selected the optimal depth automatically based on the validation R^2 metric (see Section 4.1, Table II, for the converged value). We evaluate each configuration by triple cross-validation. We use the loss

function of *MSE* and a maximum of 500 epochs with a premature stop after 20 epochs without improvement. The MedianPruner algorithm terminates unpromising attempts prematurely, saving computational time.

2.4 Creating a set of models

When we find the best architecture, we create ten networks with different initial weights from it [25]. We train each of them independently. The final prediction is the average of their outputs. The variance between predictions provides information about uncertainty – if the networks match, we can be more sure of the result.

3. CASE STUDY: CONWIP PRODUCTION LINE

3.1 System description

We tested the methodology on a five-station production line of an engineering company producing large, welded structures. The line consists of workplaces performing cutting, welding, machining, assembly and inspection operations. Each product goes through all operations in a fixed order. Line management is based on the CONWIP principle [26]. This system limits the total number of pieces in progress using authorization cards. When a new product wants to enter the line, it must take a free card. It carries this card with it through all workplaces. Only when it leaves the line, the card is released for the next product. CONWIP thus keeps the production in progress under control, regardless of changes in other conditions. We have defined ten input parameters, which are presented in Table I.

Table I: Input parameters of the CONWIP production system.

Category	Parameter	Range	Unit
Control	CONWIP limit (x_1)	[5, 40]	pieces
Control	Batch size (x_2)	[1, 5]	pcs/batch
Capacity	Operators WS1–WS5 (x_3 – x_7)	[1, 3]	persons
External	Daily demand (x_8)	[150, 350]	pcs/day
Stochastic	<i>MTBF</i> (x_9)	[50, 200]	hours
Stochastic	<i>MTTR</i> (x_{10})	[0.5, 2.0]	hours

The first two parameters are related to the management of the production flow. The other five parameters determine the capacity of the workplaces. The daily demand parameter represents an external input. The last two parameters characterize the reliability of the equipment – *MTBF* is the average time between failures and *MTTR* is the average repair time. The outputs of the metamodel consist of five indicators: *TPT* lead time (from card acquisition to completion), *WIP* work in progress (average number of pieces in the system), throughput *TH* (completed pieces per hour), utilization *Util* (average utilization of workplaces) and 95th percentile of *TPT* (indicator of delivery reliability).

3.2 Data generation and preparation

Using Latin Hypercube Sampling, we generated 14,058 different combinations of input parameters. With ten parameters, this means approximately 1,400 samples per parameter, which is a sufficient amount. We simulated each combination for 720 hours of simulated time. The first 120 hours serve as a warm-up phase; we collect data over the next 600 hours. We ran 10 replications for each configuration. The total generation time was 59 minutes on the Intel i9-12900K processor. We divided the data into three parts: 70 % for training (9,840 samples), 10 % for validation (1,406 samples) and 20 % for testing (2,812 samples). The data

successfully passed all seven control tests. The *VIF* was 1.00 for all inputs, confirming that LHS produced uncorrelated samples. The correlation analysis showed the expected connections: the CONWIP limit is strongly correlated with *WIP* ($r = 0.92$) and with throughput ($r = 0.78$).

4. RESULTS AND DISCUSSION

4.1 Neural Architecture Search results

The TPE algorithm ran a total of 200 attempts in 3 hours 11 minutes on the NVIDIA RTX 3090 graphics card. During these experiments, the algorithm tried a wide range of configurations – from simple networks with two layers and 128 neurons to complex structures with six layers and 1024 neurons. The progress of the search is shown in Fig. 2.

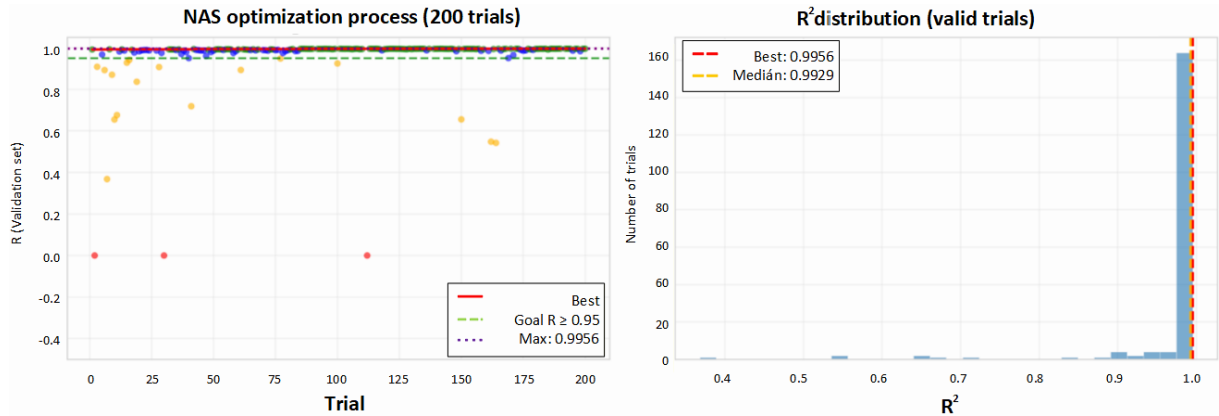


Figure 2: NAS optimization trajectory (left) and distribution of R^2 across 200 trials (right).

On the left, we can see the development of the best R^2 value during the trials. In the first attempts, the algorithm was still mapping the space, so the results were scattered. Around the 30th attempt, configurations with an R^2 above 0.99 appeared. Gradually, the best value improved with significant jumps around the 50th and 180th attempts. Convergence occurred after about 180 attempts. On the right, the histogram shows the distribution of R^2 values across all 200 attempts. Most of the attempts reached a value above 0.95. The best single attempt was $R^2 = 0.9956$. The final set of models achieved a slightly lower but more robust $R^2 = 0.9955$ – by averaging more models, the variance is reduced, which makes the model more reliable. The optimal architecture found by TPE on the 183rd attempt is described in Table II.

Table II: Optimal neural network architecture from NAS.

Hyperparameter	Optimal value
Hidden layers / Neurons	4 layers \times 256 neurons
Activation function	Swish ($\beta = 1$)
Dropout / Batch normalization	0.8 % / Yes
L2 regularization	1.65×10^{-6}
Optimizer / Learning rate	Nadam / 0.0099
Total trainable parameters	329,477

The network has four hidden layers, each containing 256 neurons. It uses Swish as an activation function, which has proven to be an effective alternative to traditional ReLU in recent years. The low dropout rate (0.8 %) and weak L2 regularization confirm that the model did not need strong regularization due to sufficient data.

4.2 Metamodel performance evaluation

We tested a set of ten models on a test set of 2,812 samples that were not used during training or architecture search. The results of the comparison of actual and predicted values are shown in Fig. 3.

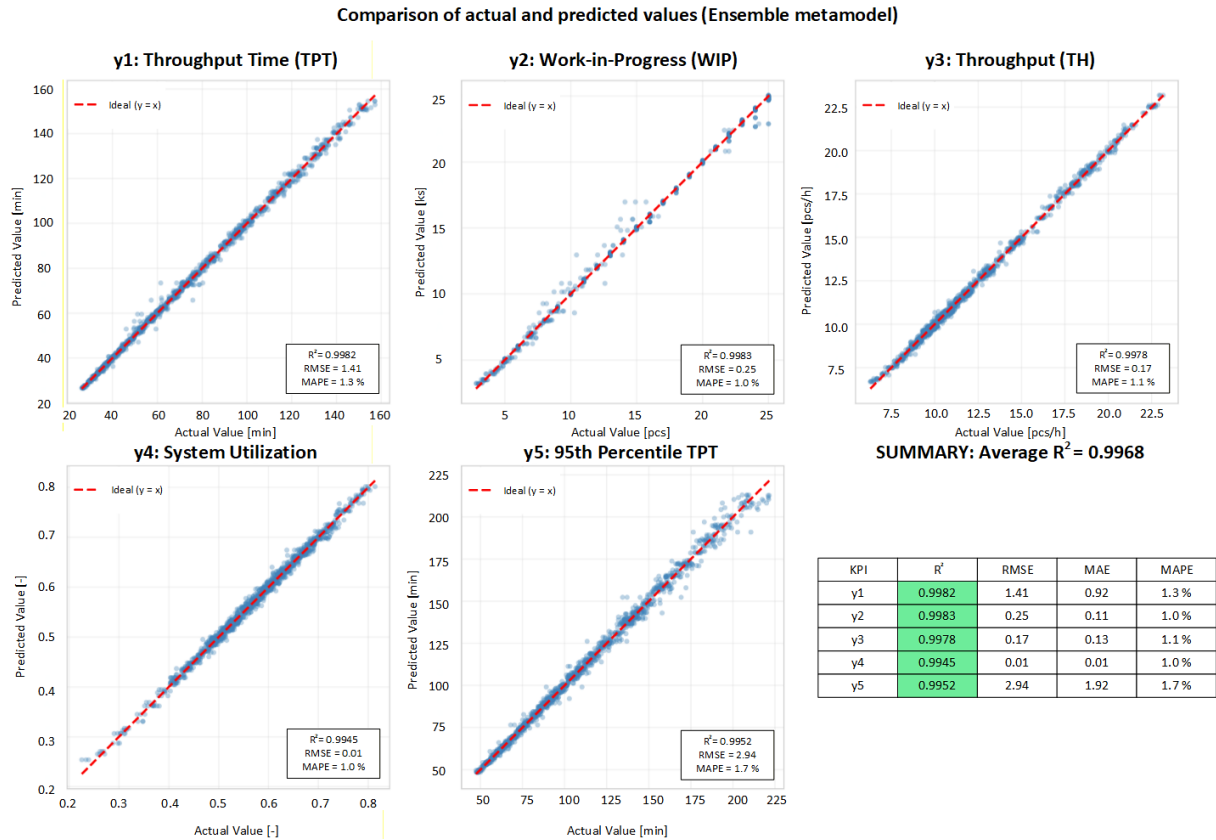


Figure 3: Comparison of predicted and actual values for all KPIs on test set ($n = 2,812$).

The figure contains five scatter plots, one for each indicator. On the horizontal axis is the actual value from the simulation, on the vertical axis is the prediction from the metamodel. The red line shows the ideal match case. For all five indicators, the points tightly encircle this line with no visible systematic deviation. We achieve the most accurate forecasts for *WIP* work in progress ($R^2 = 0.9979$, error of 1 %). This is good because *WIP* is a key parameter for CONWIP management. The *TH* throughput and *TPT* lead time reach an R^2 of around 0.997 and an error of 1.2 to 1.5 %. Utilization ($R^2 = 0.9931$) and the 95th percentile of *TPT* ($R^2 = 0.9918$) have slightly lower accuracy. The aggregated results are in Table III.

Table III: Ensemble metamodel performance metrics on test set.

KPI	R^2	RMSE	MAE	MAPE (%)
<i>TPT</i> (y_1)	0.9972	1.72 min	1.06 min	1.5
<i>WIP</i> (y_2)	0.9979	0.28 pcs	0.11 pcs	1.0
<i>TH</i> (y_3)	0.9973	0.19 pcs/h	0.14 pcs/h	1.2
<i>Util</i> (y_4)	0.9931	0.008	0.006	1.1
<i>TPT</i> 95 % (y_5)	0.9918	3.75 min	2.21 min	1.9
Average	0.9955	-	-	1.34

The average R^2 value is 0.9955 and the average percentage error is 1.34 %. In practice, this means that when the metamodel predicts, for example, a lead time of 50 minutes, the

actual value will most likely be between 49 and 51 minutes. In industrial practice, R^2 above 0.95 is considered acceptable accuracy – our metamodel significantly exceeds this limit.

4.3 Computational efficiency analysis

We compared the time needed to evaluate a different number of scenarios by simulation and metamodel. The results are shown in Fig. 4.

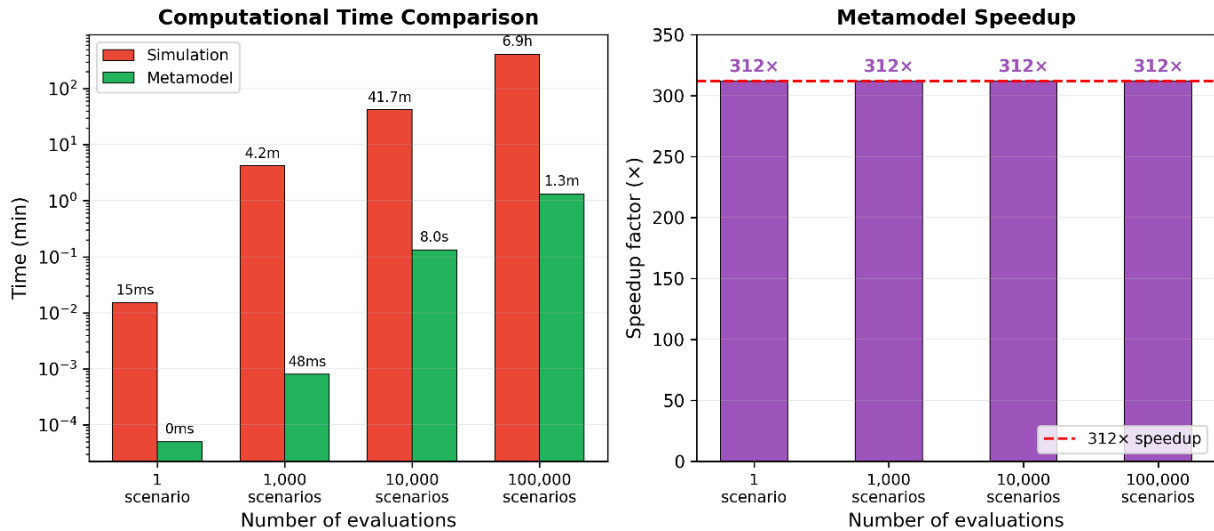


Figure 4: Computational efficiency: simulation vs. metamodel time (left) and speedup factor (right).

The consistent speedup ratio confirms that metamodel overhead is negligible compared to evaluation time. For 100,000 scenarios, the metamodel requires only 80 seconds compared to 6.9 hours for direct simulation, transforming simulation-based optimization from hours to seconds.

4.4 Multi-objective optimization results

To demonstrate practical use, we performed a multi-criteria optimization using the algorithm of differential evolution [27] with a population of 50 individuals and 30 generations (1,500 evaluations). The whole process took 1.2 seconds. The results are shown in Fig. 5.

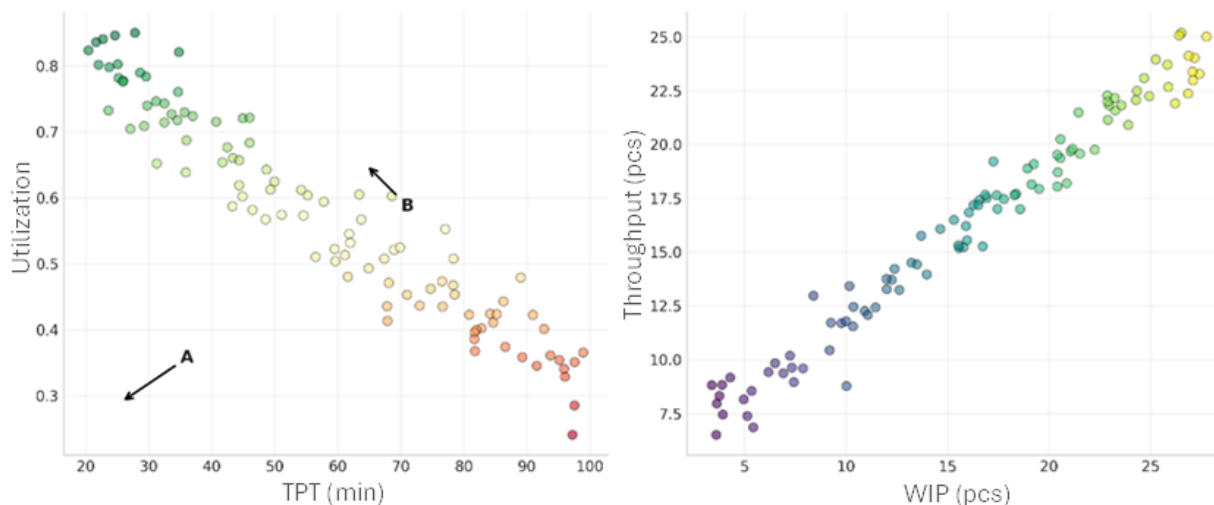


Figure 5: Trade-offs between KPIs: *TPT* vs. Utilization (left) and *WIP* vs. Throughput (right).

The left graph shows the relationship between *TPT* lead time and *Util* utilization. These indicators are in conflict: short lead times mean little work in progress and low utilization. High utilization requires more work-in-progress and therefore longer lead times. The points form a typical Pareto front – a curve along which we can move, but we cannot improve both indicators at the same time. The right graph shows the relationship between *WIP* and *TH* throughput. Here we see a positive correlation: more work in progress means higher throughput. This relationship corresponds to Little’s law ($WIP = TH \times TPT$).

Table IV: Optimal configurations for different operational objectives.

Objective	CONWIP	TPT (min)	WIP (pcs)	TH (pcs/h)	Util
Min <i>TPT</i>	12	25.6	4.0	9.9	0.29
Max <i>TH</i>	25	68.5	23.1	22.0	0.65
Balanced	18	52.1	15.1	17.5	0.73

The first configuration minimizes lead time (25.6 min) at the cost of low utilization (29%). The second maximizes throughput (22 pcs/h) with a longer lead time (68.5 min). The third is looking for a balance with the highest utilization (73 %).

4.5 Validation on scenarios

We prepared four scenarios simulating typical operational events and compared the metamodel’s predictions with the simulation. With a 30 % increase in demand, CONWIP effectively absorbed the shock. *WIP* remained unchanged, with throughput down only 0.9 %. With an operator downtime at the WS3 site, which is a bottleneck (longest processing times), lead time increased by 77 % and throughput decreased by 43 %. The change in batch size from 3 to 5 had an almost neutral impact – the lead time improved slightly (-0.6 %) due to fewer setups. The metamodel correctly identified different optimal configurations for speed-versus-efficiency modes. In all scenarios, the prediction error was below 2 %, confirming the reliability of the metamodel.

5. CONCLUSION

In this article, we introduced a procedure that enables the automatic creation of neural metamodels for production simulations. The methodology consists of five steps: creating a simulation model, generating data using Latin Hypercube Sampling, quality control with seven tests, finding an architecture using the TPE algorithm and creating a set of collaborative models. On the CONWIP production line, we achieved an accuracy of $R^2 = 0.9955$ with an error of 1.34 %, which significantly exceeds the industry requirements. The metamodel is 312 times faster than a direct simulation, allowing for interactive optimization and real-time decision-making. We have verified the practical applicability on multi-criteria optimization and scenario testing. The accuracy of the metamodel depends on the quality of the original simulation model. Based on the presented procedure, the following general guidelines can be derived for practitioners applying the methodology to other production systems: (i) the number of LHS samples should be approximately 1,000 to 1,500 per input parameter to cover the design space adequately; (ii) a warm-up phase of 15 to 20 % of the total simulated time is recommended for reliable steady-state estimation; (iii) the NAS search space should include 2 to 6 hidden layers and 128 to 1024 neurons per layer as a reasonable starting range for manufacturing metamodels; (iv) an ensemble of 8 to 10 networks balances predictive robustness and training cost; (v) data quality validation (sample size, *VIF*, correlation, outliers) should always precede architecture search to avoid wasted computation; (vi) the methodology is not limited to CONWIP systems and is applicable to any simulation-based

production model where inputs and KPIs are well defined. In the future, we plan to extend the methodology to more complex types of production systems and connect the metamodel with production information systems for adaptive management.

ACKNOWLEDGEMENT

This work was supported by the Slovak Research and Development Agency under the contract no. APVV-21-0308. This article was created with support of VEGA project: VEGA 1/0150/24. This work was supported by the KEGA Agency under the contract no. 017ŽU-4/2025.

REFERENCES

- [1] Micieta, B.; Staszewska, J.; Kovalsky, M.; Krajcovic, M.; Binasova, V.; Papanek, L.; Antoniuk, I. (2021). Innovative system for scheduling production using a combination of parametric simulation models, *Sustainability*, Vol. 13, No. 17, Paper 9518, 20 pages, doi:[10.3390/su13179518](https://doi.org/10.3390/su13179518)
- [2] Law, A. M. (2024). *Simulation Modeling and Analysis*, 6th ed., McGraw-Hill Education, New York
- [3] Fu, M. C. (Ed.) (2015). *Handbook of Simulation Optimization*, Springer, New York
- [4] Barton, R. R.; Meckesheimer, M. (2006). Metamodel-based simulation optimization, Henderson, S. G.; Nelson, B. L. (Eds.), *Simulation – Handbooks in Operations Research and Management Science*, Vol. 13, Amsterdam, 535-574
- [5] Müller, J.; Broum, T.; Malaga, M. (2023). Evaluating the reliability of a machine vision system for collaborative robots: an experimental study in the Industry 4.0 environment, *TEM Journal*, Vol. 12, No. 4, 1929-1938, doi:[10.18421/TEM124-02](https://doi.org/10.18421/TEM124-02)
- [6] Kováč, J.; Jenčík, R.; Andrejko, P.; Hajduk, M.; Pilat, Z.; Tomči, P.; Varga, J.; Bezák, M. (2020). Integrated Palletizing Workstation with an Industrial Robot and a Cobot, Berns, K.; Görges, D. (Eds.), *Advances in Service and Industrial Robotics (RAAD 2019)*, Springer, Cham, 202-209, doi:[10.1007/978-3-030-19648-6_24](https://doi.org/10.1007/978-3-030-19648-6_24)
- [7] Mleczo, J.; Dulina, L. (2014). Manufacturing documentation for the high-variety products, *Management and Production Engineering Review*, Vol. 5, No. 3, 53-61, doi:[10.2478/mper-2014-0027](https://doi.org/10.2478/mper-2014-0027)
- [8] Pasupa, T.; Suzuki, S. (2019). Impact of work-sharing on the performance of production line with heterogeneous workers, *International Journal of Industrial Engineering and Management*, Vol. 10, No. 4, 284-302, doi:[10.24867/IJIEM-2019-4-247](https://doi.org/10.24867/IJIEM-2019-4-247)
- [9] Krajčovič, M.; Plinta, D. (2012). Comprehensive approach to the inventory control system improvement, *Management and Production Engineering Review*, Vol. 3, No. 3, 34-44, doi:[10.2478/v10270-012-0022-0](https://doi.org/10.2478/v10270-012-0022-0)
- [10] Kleijnen, J. P. C. (2015). *Design and Analysis of Simulation Experiments*, 2nd ed., Springer, New York
- [11] Fonseca, D. J.; Navarrese, D. O.; Moynihan, G. P. (2003). Simulation metamodeling through artificial neural networks, *Engineering Applications of Artificial Intelligence*, Vol. 16, No. 3, 177-183, doi:[10.1016/S0952-1976\(03\)00043-5](https://doi.org/10.1016/S0952-1976(03)00043-5)
- [12] Sabuncuoglu, I.; Touhami, S. (2002). Simulation metamodeling with neural networks: an experimental investigation, *International Journal of Production Research*, Vol. 40, No. 11, 2483-2505, doi:[10.1080/00207540210135596](https://doi.org/10.1080/00207540210135596)
- [13] Vosniakos, G.-C.; Teifakis, A.; Bernardos, P. (2006). Neural network simulation metamodels and genetic algorithms in analysis and design of manufacturing cells, *The International Journal of Advanced Manufacturing Technology*, Vol. 29, Nos. 5-6, 541-550, doi:[10.1007/BF02729107](https://doi.org/10.1007/BF02729107)
- [14] Zoph, B.; Le, Q. V. (2017). Neural architecture search with reinforcement learning, *Proceedings of the 5th International Conference on Learning Representations (ICLR 2017)*, 16 pages, doi:[10.48550/arXiv.1611.01578](https://doi.org/10.48550/arXiv.1611.01578)
- [15] Pham, H.; Guan, M. Y.; Zoph, B.; Le, Q. V.; Dean, J. (2018). Efficient neural architecture search via parameter sharing, *Proceedings of ICML 2018*, 11 pages, doi:[10.48550/arXiv.1802.03268](https://doi.org/10.48550/arXiv.1802.03268)
- [16] Barton, R. R. (1992). Metamodels for simulation input-output relations, *Proceedings of the 1992 Winter Simulation Conference*, 289-299

- [17] Robinson, S. (2014). *Simulation: The Practice of Model Development and Use*, 2nd ed., Palgrave Macmillan, London
- [18] Li, Y. F.; Ng, S. H.; Xie, M.; Goh, T. N. (2010). A systematic comparison of metamodeling techniques for simulation optimization in decision support systems, *Applied Soft Computing*, Vol. 10, No. 4, 1257-1273, doi:[10.1016/j.asoc.2009.11.034](https://doi.org/10.1016/j.asoc.2009.11.034)
- [19] Hurrión, R. D. (1997). An example of simulation optimisation using a neural network metamodel: finding the optimum number of kanbans in a manufacturing system, *Journal of the Operational Research Society*, Vol. 48, No. 11, 1105-1112, doi:[10.1057/palgrave.jors.2600468](https://doi.org/10.1057/palgrave.jors.2600468)
- [20] Dos Santos, M. I. R.; dos Santos, P. M. R. (2008). Sequential experimental designs for nonlinear regression metamodels in simulation, *Simulation Modelling Practice and Theory*, Vol. 16, No. 9, 1365-1378, doi:[10.1016/j.simpat.2008.07.001](https://doi.org/10.1016/j.simpat.2008.07.001)
- [21] Elsken, T.; Metzen, J. H.; Hutter, F. (2019). Neural Architecture Search: a survey, *Journal of Machine Learning Research*, Vol. 20, 1-21
- [22] Akiba, T.; Sano, S.; Yanase, T.; Ohta, T.; Koyama, M. (2019). Optuna: a next-generation hyperparameter optimization framework, *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD 2019)*, 2623-2631, doi:[10.1145/3292500.3330701](https://doi.org/10.1145/3292500.3330701)
- [23] Meng, F.; Zhang, L.; Chen, Y.; Wang, Y. (2024). Sample-based dynamic hierarchical transformer with layer and head flexibility via contextual bandit, *Proceedings on Engineering Sciences*, Vol. 6, No. 2, 439-452, doi:[10.24874/PES06.02.001](https://doi.org/10.24874/PES06.02.001)
- [24] McKay, M. D.; Beckman, R. J.; Conover, W. J. (1979). A comparison of three methods for selecting values of input variables in the analysis of output from a computer code, *Technometrics*, Vol. 21, No. 2, 239-245, doi:[10.2307/1268522](https://doi.org/10.2307/1268522)
- [25] Lakshminarayanan, B.; Pritzel, A.; Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using Deep Ensembles, *31st Conference on Neural Information Processing Systems (NIPS 2017)*, 12 pages
- [26] Spearman, M. L.; Woodruff, D. L.; Hopp, W. J. (1990). CONWIP: a pull alternative to kanban, *International Journal of Production Research*, Vol. 28, No. 5, 879-894, doi:[10.1080/00207549008942761](https://doi.org/10.1080/00207549008942761)
- [27] Storn, R.; Price, K. (1997). Differential Evolution – a simple and efficient heuristic for global optimization over continuous spaces, *Journal of Global Optimization*, Vol. 11, No. 4, 341-359, doi:[10.1023/A:1008202821328](https://doi.org/10.1023/A:1008202821328)