NAM-CAM: Neural-Additive Models for Semi-Analytic Descriptions of CAM Simulations

Konstantin Ditschuneit$^{1[0000-0002-1120-1030]}$, Adem Frenk^{1}, Markus Frings$^{2[0000-0001-9096-939X]}$, Viktor Rudel^{3}, Stefan Dietzel$^{1[0000-0002-4192-0327]}$, Johannes S. Otterbach$^{1[0000-0002-7404-2321]}$

^{1} Merantix Momentum, Berlin, Germany
^{2} ModuleWorks, Aachen
^{3} Fraunhofer IPT, Aachen
{`konstantin.ditschuneit, johannes.otterbach`}`@merantix.com`

Abstract. Computer-Aided Manufacturing (CAM) is an iterative, time- and resource-intensive process involving high computational costs and domain expertise. The exponentially large CAM configuration space is a major hurdle in speeding up the CAM iteration process. Existing methods fail to capture the complex dependency on CAM parameters. We address this challenge by proposing a new element for the engineer's design workflow based on an explainable artificial intelligence method. Using Neural-Additive Models (NAMs), we create a semi-analytic model that improves guided search through the configuration space and reduces convergence time to an optimal CAM parameter set. NAMs allow us to visualize individual parameter contributions and trivially compute their sensitivity. We demonstrate the integration of this new element into the CAM design process of a blade-integrated disk (blisk). By visualizing the learned parameter contributions, we successfully leverage NAMs to model the dependency on CAM parameters.

Keywords: Computer-Aided Manufacturing, blisk, interpretable machine learning, XAI, Neural-Additive Models

1 Introduction

With the advance of modern manufacturing technologies, designing new technical components becomes increasingly more complex. While planning the machining process of these components, engineers make extensive use of Computer-Aided Manufacturing (CAM) systems. Typical CAM-based simulation workflows consist of multiple subsequent steps, such as (1) tool path calculation; (2) tool engagement simulation; and (3) cutting force simulation. CAM designers visually inspect the tool path w.r.t. tool accelerations, trajectory and tool orientation smoothness, and general machinability of the part and adapt the CAM parameters accordingly. However, a typical CAM parameter space consists of around 50 parameters. When testing just 3 independent settings for each parameter, the total number of configurations to calculate is $3^{50} \approx 7.2\text{e}23$, rendering an exhaustive search of this configuration space infeasible. A major limiting factor is the simulation of relevant process variables, such as the cutting force, using CAM-integrated technology models. Especially, the dexel-based tool engagement simulation makes the workflow expensive, as it requires high tool path and part resolutions. Consequently, human intuition plays a significant role in guiding the exploration. However, the individual steps in the CAM workflow commonly run for multiple hours on modern computers, and engineers often spend long hours optimizing CAM parameters by hand and abort once a satisfactory result is achieved.

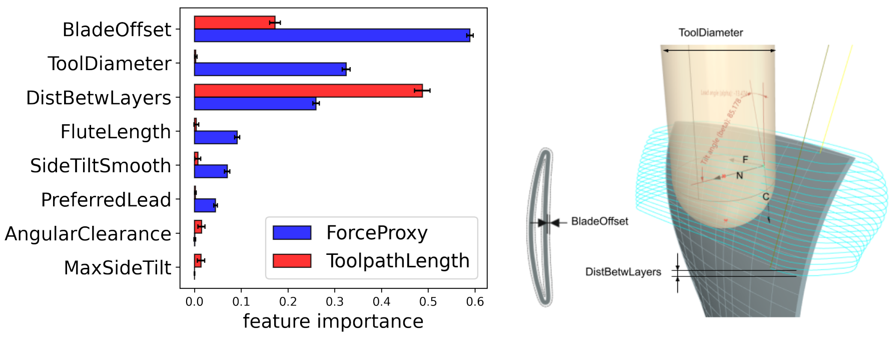

Fig. 1. (*Left*) Learned parameter importance α_d (see Equation 1) for targets toolpath length (red) and force proxy (blue). The plots reveal that some parameters have negligible contributions overall, e.g., MaxSideTilt, while BladeOffset and DistBetweenLayers are influential for both targets. (*Right*) 3D sketch of a blisk with the toolpath in blue and the machining tool head in yellow.

In recent years, the success of advanced Machine Learning (ML) techniques, have been extended from traditional applications, e.g., computer vision and natural language processing, to new domains, such as physical system [3,8,23] or engineering [17,19,26]. We add to this research by introducing a new design step that can be integrated into the engineer's workflow, based on NAMs [1,15], an explainable artificial intelligence technique. Studying the challenging machine planning process of blade-integrated disks, or blisks (see Fig. 1 left), we show that NAMs can assist in the guided search of the CAM parameter space. NAMs constitute a semi-analytic model of the CAM simulation, due to leveraging modern auto-differentiation frameworks. The architecture of NAMs allows to easily visualize the parameter-contributions to the quantity of interest and facilitates further analyses, such as sensitivity analysis, intuition checks, and insights into the simulation dynamics. The interpretability features enable engineers to assess the quality of the final model and guide them in their experimentation and search for optimal CAM parameter settings. Finally, NAMs allow for further automation and multi-target optimization when turning the quantities of interest into a corresponding optimization metric.

2 Related Work

Artificial Intelligence (AI) in manufacturing. The study of AI-based methods to optimize manufacturing processes is not new [22]. Most works are based on a variety of zero-order optimization routines [14,25,27], while artificial neural networks have received considerably less attention and are focused on path optimization problems [29].

Interpretable models for tabular data. The parameter dependency of the CAM simulation can be expressed as tabular data. In practice, tabular data are often modeled using linear models, such as regressions or discriminant analysis [13]. Linear models require hand-engineered features to achieve good performance, making them time-consuming to build. To increase the expressivity of such models, Generalized Additive Models (GAMs) [9] have been introduced to model non-linear but still univariate regressions. The non-linear dependencies are often modeled with tree-based models, such as xgboost [16]. While these tree-based GAMs are powerful, they are not differentiable, making any analysis cumbersome. In contrast, NAMs [1] cover the full expressivity of GAMs while also being fully differentiable. This allows for different types of analyses and optimizations. Moreover, NAMs, being neural network-based algorithms, benefit from modern accelerators, such as Graphics Processing Units (GPUs), due to their matrix-multiplication parallelism.

3 Neural-Additive Models

NAMs are instances of GAMs [9]. This model class uses non-linear univariate functions to approximate the quantity of interest. Let $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1,\ldots,N}$ be a dataset of N samples, with $\mathbf{x}_i \in \mathbb{R}^D$ denoting the independent variables and $y_i \in \mathbb{R}^K$ the target variable. For a simple regression, $K = 1$, but we also investigate multi-target regressions with $K > 1$, which enables us to re-use parameters in a multi-target regression. We express the relationship between dependent and independent variables as

$$\tilde{y}_i = \alpha_0 + \sum_{d=1}^{D} \alpha_d \, \phi_d \left(\frac{x_{i,d} - \mu_d}{\sigma_d} \right) , \tag{1}$$

where ϕ_d is a non-linear function, the *shape function*, of the d-th component of vector $\mathbf{x}_i$, and α_d are parameters that are determined numerically. Note that we normalize the input data using a component-wise shift μ_i and scale σ_i. The tilde indicates that this is the approximated value and not the ground truth value. In the NAM setting, we replace the functions ϕ_d with neural networks, i.e., $\phi_d(\cdot) = \mathrm{NN}[\boldsymbol{\theta}_d](\cdot)$, where $\boldsymbol{\theta}_d$ are the trainable parameters of the d-th component. To train the NAM, we make use of the mean squared error (MSE) defined as $\mathcal{L}_{\text{MSE}} = \mathbb{E}_{\mathbf{x}_i \in \mathcal{D}} \left[\|\tilde{y}_i - \frac{y_i - \mu_y}{\sigma_y}\|^2 \right]$, where we also scale the target variable for numerical stability. Minimizing $\mathcal{L}_{\text{MSE}}$ w.r.t. the family of parameters $\{\boldsymbol{\theta}_d\}_{d=1,\ldots,D}$ and $\{\alpha_d\}_{d=0,\ldots,D}$ results in shape functions that capture the influence of the input variables onto the target quantity. A generalization of model 1 to multivariate shapes for higher-order interactions is straight-forward [15]. This increases the expressivity of the model but reduces the interpretability due to higher-dimensional shape functions.

Sensitivity analysis. Using neural networks as shape functions ϕ_d makes model 1 fully differentiable. This allows us to compute and visualize local sensitivity measures defined via the first derivative

$$s_d(\mathbf{x}_0) = \left| \frac{\partial y}{\partial x_d} \right|_{\mathbf{x}_0} . \tag{2}$$

We can define a global sensitivity measure by computing, for instance, the maximum local sensitivity over a bounded value range. This assumption is acceptable, as most input parameters in a CAM system have a finite range.

Uncertainty assessment for the shape functions. The interpretability properties of univariate NAMs offer the added benefit of visualizing the uncertainty of the shape functions. For small- to medium-sized datasets, fitting a NAM is fast. This allows for several strategies to create uncertainty estimates, e.g., bootstrapping [13], ensemble learning [13], Bayesian dropout [11], or randomized NAM initialization. Independent of the specific choice, any of these methods results in varying shape functions to assess uncertainty. Using human assessment, the engineer can determine whether additional experiments are needed.

4 Experiments

4.1 CAM Toolpath Calculation and Force Predictions

For the CAM workflow, we use the ModuleWorks SDK. The toolpath is calculated with the MultiBlade component, while the engagement simulation utilizes the CutSim component. Simulations can be done using mesh-based [12] or dexel-based approaches, such as tri-dexel models [4] used here. Compared to mesh-based methods, memory consumption grows moderately with growing complexity of the mechanical part. However, due to its discrete representation, the dexel-method misses features smaller than the dexel distance and requires an increased dexel resolution. To save computational costs, a buffered-cuts simulation approach is used. Therefore, the tool movement over several steps is summed up before updating the in-process workpiece (IPW) by intersecting the swept volume of the tool and the old IPW.

To calculate forces on the tool, the cutter-workpiece engagement (CWE) is required in combination with a mechanistic cutting force model [2,6,7]. We follow a strategy based on a detailed 3D tool shape model [6]. However, accurate force estimates are at odds with the buffering, as fine-grained geometry updates are needed. This makes the computation slow.

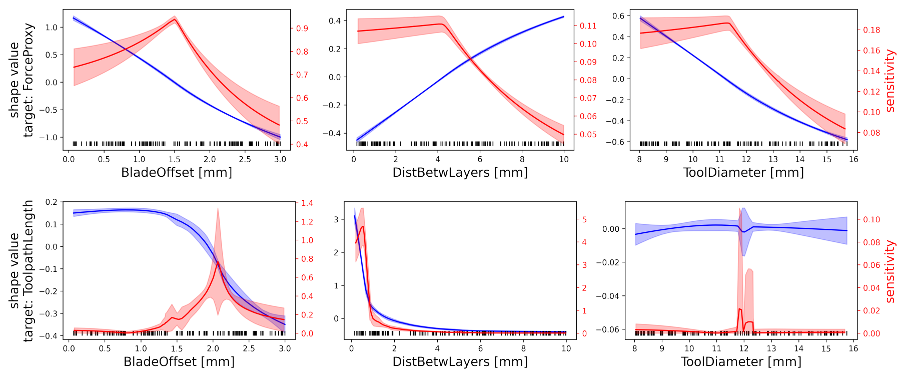

Fig. 2. Shape functions of the learned NAMs for both targets (top: force proxy, bottom: toolpath length). Shapes for the force proxy show almost linear correspondence, while the shapes for the toolpath length are non-linear. NAMs are clearly capable of simultaneously learning both simple and complex correlations. The uncertainty bands are derived from ensemble learning and correspond to 1σ standard deviation for shape functions and sensitivity. Please note the differences in scale for the different curves. The black stripes mark individual feature values in the dataset to indicate the density of data values.

4.2 Dataset Generation and NAM Training

To generate the dataset, we use Docker [21] to containerize the ModuleWorks SDK. Alternatively, the CAM simulator can also be accessed via an internet-based API endpoint. Either setup allows abstracting the underlying operating system and running jobs on any hardware. We use Kubernetes [20] to orchestrate the workflow and automate it using Flyte [10] in conjunction with a Hydra entrypoint [28].

With this setup, we simulate the blisk toolpath using a set of initial parameters. We parallelize the dataset simulation by horizontally scaling the environment. For each simulation, we record the complete trajectory of the path, including the tool immersion angles, slices, and so forth. We are interested in minimizing the average absolute force between the tool and the workpiece during the cutting process over all points on the toolpath. Since we do not have access to the exact force calculation, we use an empirically tested proxy metric based on the sum of the cut-intersection surface area $a_{t,k}$ between the component and cutting edge of the tool t: $\tilde{F}_t = \sum_{k=1}^{3} a_{t,k}$.

4.3 Experiment Design

Dataset. We ran the CAM workflow for 128 different parameter configurations sampled uniformly at random. We collect the parameters and the corresponding target metrics as rows in our tabular dataset. The data consists of 8 independent parameters and 2 target metrics. We randomly split the dataset into a training

(80%), a validation (10%), and a test (10%) dataset. To fit the NAM, we normalize the target metrics to be mean-centered and to have a unit standard deviation and the input variables to lie within the $[0, 1]$ range. To estimate the shift and scale values, we use the training set to avoid leakage of the test set.

Models and training. The NAMs are implemented with PyTorch [24] and use multi-target regression with $K = 2$ to model both metrics simultaneously. The networks are trained using mini-batch gradient descent using the AdamW optimizer [18]. Furthermore, we perform hyperparameter optimization using a Tree-structured Parzen Estimator [5] to maximize the $R2$ scores [13] of the model. We train 256 different models for both targets, force proxy and toolpath length and rank models based on their $R2$ scores. The best ones predict the normalized targets with a mean absolute error (MAE) [13] of 0.24 and 0.13, respectively.

4.4 Discussion

We demonstrate the usefulness of the NAMs in Fig. 1, where we show the importance of each CAM parameter on the different target quantities. As can be seen clearly, different parameters influence the quantities of interest differently. For instance, the tool diameter has a more significant impact on the force due to increased surface area but has a vanishing impact on the path length due to a constant layer distance.Examples of the shape functions underlying the models are depicted in Fig. 2. We show the mean of the shape in blue with the 1σ standard deviation depicted by the shaded area and computed over the best ten runs. The sensitivity in red is computed per individual run, then averaged. Here the shaded region shows 1σ standard deviation as the error propagation makes the signal noisy. We can see the different influences of the parameters on the final target metrics due to the different shape functions. At the same time, the sensitivity dependence is similar in shape but not magnitude.

5 Conclusion

The application of interpretable AI models to manufacturing processes enables new insights during the design process. We showed that by using NAMs, we can understand detailed parameter influences of a CAM design system, which subsequently can be used for guided design exploration, uncertainty assessments, and sensitivity analyses. We demonstrated the power of NAMs using a blisk design and showed how we can predict toolpath length and tool force simultaneously while also extracting the sensitivities of the metrics w.r.t. these parameters using the differentiable nature of neural networks.

We also highlight that the method is use-case agnostic and can be applied to all industrial applications that leverage CAM to improve the manufacturing process. We hypothesize that it will speed up the design process by enabling power users and users with limited domain knowledge to better and faster understand the simulation tools and their input settings.

Future work will extend the NAM framework to include higher-order interactions and investigate the use of uncertainty metrics for automated design optimization using Bayesian learning techniques. Moreover, we are interested in the automation and integration of such tools into the engineering design process.

Acknowledgment. We kindly acknowledge funding by the German Federal Ministry of Education and Research (BMBF) within the project "CAM2030: Entwicklung einer innovativen Lösung für das Advanced Systems Engineering der computergestützten Prozessplanung der Zukunft" (#02J19B080, #02J19B082, #02J19B084).

Contributions. KD and JSO designed the experiments and NAMs and performed the analysis. AF and MF built the experimentation framework and simulation environment. VR provided the blisk setup. SD and JSO coordinated and oversaw the project. All authors contributed to the manuscript.

References

1. Agarwal, R., Frosst, N., Zhang, X., Caruana, R., Hinton, G.E.: Neural Additive Models: Interpretable Machine Learning with Neural Nets. arXiv (2020)
2. Altintas, Y., Kersting, P., Biermann, D., Budak, E., Denkena, B., Lazoglu, I.: Virtual process systems for part machining operations. CIRP Annals **63**(2), 585–605 (2014)
3. Belbute-Peres, F.d.A., Economon, T.D., Kolter, J.Z.: Combining Differentiable PDE Solvers and Graph Neural Networks for Fluid Flow Prediction. arXiv (2020)
4. Benouamer, M.O., Michelucci, D.: Bridging the gap between csg and brep via a triple ray representation. In: Proceedings of the fourth ACM symposium on Solid modeling and applications, pp. 68–79 (1997)
5. Bergstra, J., Bardenet, R., Bengio, Y., Kégl, B.: Algorithms for hyper-parameter optimization. In: J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, K. Weinberger (eds.) Advances in Neural Information Processing Systems, vol. 24. Curran Associates, Inc. (2011). URL https://proceedings.neurips.cc/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf
6. Boess, V., Ammermann, C., Niederwestberg, D., Denkena, B.: Contact zone analysis based on multidexel workpiece model and detailed tool geometry representation. Procedia CIRP **4**, 41–45 (2012)
7. Boz, Y., Erdim, H., Lazoglu, I.: A comparison of solid model and three-orthogonal dexelfield methods for cutter-workpiece engagement calculations in three-and five-axis virtual milling. The International Journal of Advanced Manufacturing Technology **81**(5), 811–823 (2015)
8. Brandstetter, J., Worrall, D., Welling, M.: Message Passing Neural PDE Solvers. arXiv (2022)
9. Cao, L., Zhang, C., Joachims, T., Webb, G., Margineantu, D.D., Williams, G., Caruana, R., Lou, Y., Gehrke, J., Koch, P., Sturm, M., Elhadad, N.: Intelligible Models for HealthCare. Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining pp. 1721–1730 (2015)
10. Flyte: Flyte: The Workflow Automation Platform for Complex, Mission-Critical Data and Machine Learning Processes at Scale (2022). URL https://github.com/flyteorg/flyte

11. Gal, Y., Ghahramani, Z.: Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. arXiv (2015)
12. Gong, X., Feng, H.Y.: Cutter-workpiece engagement determination for general milling using triangle mesh modeling. Journal of Computational Design and Engineering **3**(2), 151–160 (2016)
13. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. Springer Series in Statistics (2009)
14. Karuppusamy, N.S., Kang, B.Y.: Minimizing airtime by optimizing tool path in computer numerical control machine tools with application of A* and genetic algorithms. Advances in Mechanical Engineering **9**(12), 1687814017737,448 (2017)
15. Kim, M., Choi, H.S., Kim, J.: Higher-order Neural Additive Models: An Interpretable Machine Learning Model with Feature Interactions. arXiv (2022)
16. Krishnapuram, B., Shah, M., Smola, A., Aggarwal, C., Shen, D., Rastogi, R., Chen, T., Guestrin, C.: XGBoost: A Scalable Tree Boosting System. arXiv pp. 785–794 (2016)
17. Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., Anandkumar, A.: Neural Operator: Graph Kernel Network for Partial Differential Equations. arXiv (2020)
18. Loshchilov, I., Hutter, F.: Fixing weight decay regularization in adam. CoRR **abs/1711.05101** (2017). URL http://arxiv.org/abs/1711.05101
19. Lötzsch, W., Ohler, S., Otterbach, J.S.: Learning the Solution Operator of Boundary Value Problems using Graph Neural Networks. arXiv (2022)
20. Martin, P.: Kubernetes (2021)
21. Merkel, D.: Docker: Lightweight linux containers for consistent development and deployment. Linux J. **2014**(239) (2014)
22. Narooei, K.D., Ramli, R.: Application of Artificial Intelligence Methods of Tool Path Optimization in CNC Machines: A Review. Research Journal of Applied Sciences, Engineering and Technology **8**(6), 746–754 (2014)
23. Ohler, S., Brady, D., Lötzsch, W., Fleischhauer, M., Otterbach, J.S.: Towards Learning Self-Organized Criticality of Rydberg Atoms using Graph Neural Networks. arXiv (2022)
24. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: PyTorch: An Imperative Style, High-Performance Deep Learning Library. arXiv (2019)
25. Pezer, D.: Efficiency of Tool Path Optimization Using Genetic Algorithm in Relation to the Optimization Achieved with the CAM Software. Procedia Engineering **149**, 374–379 (2016)
26. Pfaff, T., Fortunato, M., Sanchez-Gonzalez, A., Battaglia, P.W.: Learning Mesh-Based Simulation with Graph Networks. arXiv (2020)
27. Tsagaris, A., Mansour, G.: Path planning optimization for mechatronic systems with the use of genetic algorithm and ant colony. IOP Conference Series: Materials Science and Engineering **564**(1), 012,051 (2019)
28. Yadan, O.: Hydra - a framework for elegantly configuring complex applications. Github (2019). URL https://github.com/facebookresearch/hydra
29. Zuperl, U., Cus, F.: Optimization of cutting conditions during cutting by using neural networks. Robotics and Computer-Integrated Manufacturing **19**(1-2), 189–199 (2003)