Machine Learning Surrogate Models for Performance Prediction with Architectural Models

Sebastian Weber
sebastian.weber@fzi.de
FZI Research Center for Information Technology

Vincenzo Pace
vincenzo.pace@student.kit.edu
Karlsruhe Institute of Technology

Thomas Weber
thomas.weber@kit.edu
Karlsruhe Institute of Technology

Jörg Henß
henss@fzi.de
FZI Research Center for Information Technology

Robert Heinrich
robert.heinrich@uni-ulm.de
Ulm University

Abstract

Predicting the performance of software systems enables software architects to predict the fulfillment of quality requirements such as latency or throughput during design and operation. Simulation approaches like the Palladio approach provide accurate estimates, but become costly in highly explorative design processes. We investigate machine learning surrogate models that approximate simulation outputs with reduced computational effort. To this end, we generate synthetic architectural models in the Text-Based Palladio Component Model format, simulate them with Palladio approach to obtain performance metrics, and train ML models on textual embeddings of these architectures. The surrogate models predict average response times faster than a full simulation and exceed trivial baselines but have room for further improvements in predictive accuracy.

1 Introduction

Estimating the performance of software systems before implementation allows architects to explore design alternatives and to identify potential bottlenecks early. Approaches such as the Palladio approach [1] provide accurate predictions of latency, throughput, or scalability, but repeated simulations become costly in settings where large numbers of alternatives need to be analyzed, for example during automated design space exploration or in continuous integration pipelines.

This work is based on a master's thesis [10] investigating whether machine learning (ML) surrogate models can complement simulation by providing faster approximations of performance metrics. In line with the vision of hybrid approaches that combine ML and simulation [9], surrogate models are not intended to replace simulation entirely, but to accelerate evaluations in scenarios where a large number of design alternatives can be explored. We implemented a fully automated pipeline, visualized in Figure 1, for dataset generation and surrogate model training. The term model refers to two different kinds of models, namely architectural models in the first half of the pipeline and surrogate ML models in the second part. The synthetic architectural models were created in the Text-Based Palladio Component Model (TPCM)[1] format and converted into Palladio Component Model (PCM) instances for simulation. Using the Palladio simulator, more than 15000 architecture models were simulated, producing performance metrics, i.e., average response time. This data formed the basis for training and comparing multiple regression models, namely linear baselines, support vector regression, random forests, and neural networks.

The textual input of the models was transformed into numerical features through TF-IDF (Term Frequency-Inverse Document Frequency) [6] embeddings to serve as a simple baseline, while transformer-based embeddings based on CodeBERT [5] were applied to capture richer structural patterns. The surrogate models were then evaluated against the Palladio results to assess their accuracy, generalizability, and runtime. The evaluation shows that surrogate models consistently outperform trivial baselines, e.g., prediction based on averaged response times over all input models, in terms of mean absolute error, but that prediction quality in terms of the coefficient of determination R^2 varies substantially between methods and embeddings. These findings demonstrate that surrogate-based prediction from architectural models is feasible, but further research is needed to improve robustness and generalization.

2 Model Training

The training of surrogate models is realized through a multi-stage pipeline, which is summarized in Fig-

[1] https://github.com/PalladioSimulator/Palladio-Addons-TextBasedModelGenerator

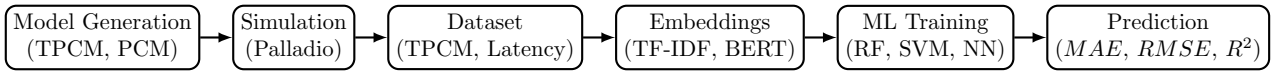

Figure 1: Overview of the pipeline from model generation to ML-based prediction.

ure 1. The pipeline begins with model generation, where synthetic architectural models were automatically created in the TPCM format. This domain-specific language allows the structural definition of software components, interfaces, and system compositions in a compact and textual representation. The models contained varying numbers of TPCM model elements that were randomly and syntactically correctly connected. The generated models are not based on existing systems. For simulation purposes, the TPCM instances were converted into PCM instances, which can be processed by the Palladio simulator. In the subsequent simulation step, the PCM models were simulated using Palladio to produce quantitative performance data. The main performance indicator considered in this study is latency, more specifically, the average response time of the modeled system under simulated workloads.

The generated models and their corresponding simulation results were stored in a dataset. Each entry consists of a TPCM description paired with its simulated latency value. In total, more than 15,000 architecture–latency pairs were collected, covering different system sizes and interaction patterns. This dataset provides the empirical foundation on which the surrogate learning was performed. As machine learning algorithms require numerical input features, the textual architecture models were transformed through two different embedding techniques. The first is Term Frequency–Inverse Document Frequency (TF-IDF) [6], a classical method from information retrieval that represents documents as weighted vectors based on the occurrence of tokens. The second is a transformer-based approach using CodeBERT [5], which generates contextual embeddings designed to capture structural and semantic information in source code and related technical languages.

With these embeddings, four types of regression models were trained:

- Linear Models, i.e., ridge regression and lasso regression, which provide a simple and interpretable baseline
- Random Forests (RF), which combine multiple decision trees into an ensemble predictor
- Support Vector Machines (SVM), which fit regression hyperplanes in high-dimensional feature spaces
- Neural Networks (NN), which can approximate complex nonlinear mappings through stacked layers of neurons

Each of these model families offers different trade-offs in terms of interpretability, computational efficiency, and generalization ability. The last step is the usage of the trained surrogate models for prediction, which is detailed in the following Section 3.

3 Evaluation

The dataset was split into training, validation, and test sets (80/10/10), and the predictor of the mean latency was used as a baseline. The trained surrogates were assessed using standard regression metrics: the Mean Absolute Error (MAE), which is shown in Table 1, to measure the average deviation between predicted and simulated latencies, the Root Mean Squared Error (RMSE) to emphasize larger errors, and the coefficient of determination (R^2) to quantify how well the predictions explain the variance in the data. Each model type was trained with both TF-IDF and CodeBERT embeddings and evaluated under different hyperparameter configurations. For linear models these were variations of regularization strength, while for random forests the number and depth of trees were tuned. Support vector regression models were tested with different kernel functions and penalty parameters, and neural networks were trained with varying numbers of layers, hidden units, and learning rates.

TF-IDF embeddings often outperformed CodeBERT, possibly due to the sliding-window chunking required by limited computing resources. Among regressors, linear models such as ridge regression and support vector regression achieved the most stable results. Random forests (RF) performed comparably but with higher variance, possibly due to the large range of the hyperparameter used. Neural networks showed no clear improvement and occasionally overfitted. A key limitation of the approach and the results presented lies in the characteristics of the training data and the available resources. The dataset, while comprising more than 15,000 samples, was still limited in size relative to the possible complexity of the architectures and covered only a small part of this complexity. In addition, it exhibited a skewed distribution of latency values, leading to imbalance and difficulties in learning rare cases. This affected the ability of the models to generalize, particularly in scenarios with extreme performance outliers. In addition, training was constrained by limited computational resources, which required compromises such as sliding-window chunking for transformer embeddings and restricted the scope of hyperparameter optimization.

Embedding	Metric	Lasso	Ridge	RF	SVM	Neural Net
CodeBERT	Mean MAE	0.4955	0.5288	0.4389	0.3418	0.2719
	Median MAE	0.4672	0.4818	0.6289	0.4158	0.4127
	n (samples)	36	36	71	96	118
TF–IDF	Mean MAE	0.2097	0.2705	0.3973	0.4916	0.1611
	Median MAE	0.1582	0.2197	0.6447	0.4575	0.1378
	n (samples)	39	39	41	165	119

Table 1: High level comparison of different models across all model run configurations and embeddings

Overall, surrogate models outperformed the baseline in terms of MAE and RMSE, but R^2 values were often negative, indicating difficulty in explaining the variance of simulated response times. The skewed distribution of latency values with several outliers contributed to this instability. These results suggest that surrogate prediction from TPCM input is feasible, but improvements in representation and data quality are needed to achieve robust generalization.

4 Related Work

Machine learning has been applied to performance prediction in several domains. In configurable systems, testing across large configuration spaces is infeasible, which motivated surrogates such as DeepPerf [3] and Perf-AL [8]. Both showed that deep neural architectures and adversarial learning can provide accurate predictions from sparse samples. In high-performance computing, ML has been used to predict execution times and scalability. Malakar et al. [2] compared ensemble methods and neural networks, while Mankodi, Bhatt, and Chaudhury [7] studied shallow and deep networks, illustrating trade-offs between model complexity and training data size. Code-level approaches such as DeepTLE [4] predict runtime performance from source code tokens without execution, but they require code and thus target later development stages. In contrast, our work explores surrogate modeling directly from architectural descriptions in the TPCM, enabling performance estimation before implementation [10].

5 Conclusion

We explored machine learning surrogate models for predicting the performance of software architectures described in the TPCM format in this paper. Based on more than 15,000 simulated models, we trained different regressors on TF-IDF and CodeBERT embeddings. The results showed that surrogates outperform a trivial baseline in terms of MAE and RMSE, but their R^2 remained limited. TF-IDF combined with linear models provided the most stable performance, while neural networks and transformer embeddings did not yield clear benefits.

Future work should focus on larger and more balanced datasets, improved representations such as graph-based embeddings, and more extensive hyperparameter optimization. In addition, the integration of surrogates into design space exploration and continuous integration pipelines could demonstrate their practical applicability in real-world development processes.

Acknowledgements

This work has received funding from the European Chips Joint Undertaking under Framework Partnership Agreement No 101139789 (HAL4SDV) including the national funding from the German Federal Ministry of Research, Technology and Space (BMFTR) under grant number 16MEE0468. The responsibility for the content of this publication lies with the authors. This work was funded by the DFG (German Research Foundation) – project number 499241390 (FeCoMASS), by the Topic Engineering Secure Systems of the Helmholtz Association (HGF) and supported by KASTEL Security Research Labs, Karlsruhe and supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - SFB 1608 - 501798263.

References

[1] R. H. Reussner et al. *Modeling and simulating software architectures: The Palladio approach*. MIT Press, 2016.

[2] P. Malakar et al. "Benchmarking Machine Learning Methods for Performance Modeling of Scientific Applications". In: *2018 IEEE/ACM Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS)*. Nov. 2018, pp. 33–44.

[3] H. Ha and H. Zhang. "DeepPerf: Performance Prediction for Configurable Software with Deep Sparse Neural Network". In: *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. May 2019, pp. 1095–1106.

[4] M. Zhou et al. "Deeptle: Learning code-level features to predict code performance before it runs". In: *2019 26th Asia-Pacific Software Engineering Conference (APSEC)*. IEEE. 2019, pp. 252–259.

[5] Z. Feng et al. "Codebert: A pre-trained model for programming and natural languages". In: *arXiv preprint arXiv:2002.08155* (2020).

[6] P. Koehn. *Neural machine translation*. Cambridge University Press, 2020.

[7] A. Mankodi, A. Bhatt, and B. Chaudhury. "Evaluation of Neural Network Models for Performance Prediction of Scientific Applications". In: *2020 IEEE REGION 10 CONFERENCE (TENCON)*. Nov. 2020, pp. 426–431.

[8] Y. Shu et al. "Perf-AL: Performance Prediction for Configurable Software through Adversarial Learning". In: *Proceedings of the 14th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. Bari Italy: ACM, Oct. 5, 2020, pp. 1–11.

[9] L. von Rueden et al. "Combining Machine Learning and Simulation to a Hybrid Modelling Approach: Current and Future Directions". In: *Advances in Intelligent Data Analysis XVIII*. Springer, 2020, pp. 548–560.

[10] V. Pace. "Evaluating the suitability of ML-based surrogate models for performance prediction with design-time software architecture models". Unpublished. Master's Thesis. Karlsruher Institut für Technologie (KIT), 2025.