Robust Development of Active Learning-based Surrogates for Induction Motor

Janne Keränen[1], Mikko Tahkola[1], Peter Råback[2], Álvaro González[2], Victor Mukherjee[3], and Jenni Pippuri-Mäkeläinen[1]

[1]VTT Technical Research Centre of Finland Ltd, FI-02044 VTT, Finland
[2]CSC–IT Center for Science Ltd, FI-02101 Espoo, Finland
[3]ABB Motors and Generators, Technology Center, FI-00380 Helsinki, Finland

A robust open-source cloud-based workflow is developed for finite element (FE) data generation for active learning (AL) -based surrogate modelling. Special attention is paid to making the FE solution procedure as robust and fast as possible without human intervention by, e.g., implementing special convergence criteria, reliable parallel computation, and variable timestep length. In AL, a surrogate model automatically improves itself by iteratively querying more FE data. Using AL and large datasets generated with parallelised cloud FE simulations, we develop a surrogate model to rapidly predict induction machine steady-state torque, torque ripple, total losses, and current harmonic distortion, as a function of motor frequency, voltage, and slip. Results show that AL performs better than grid sampling and on average works as well as random sampling, but with some outputs, the results vary less with AL. In addition, accurate ripple estimation requires a much larger training dataset than the other variables.

***Index Terms*—Cloud computing, data-driven modeling, finite element analysis, induction motors, machine learning.**

I. Introduction

DATA-DRIVEN MODELS offer new ways of modelling and simulating electrical machines (EMs). For long, ML has been utilised in the analysis of field data for fault diagnosis purposes. Lately, also different surrogate [1] and reduced order models [2] have been developed. The idea of these surrogate models is to distil the accuracy of a high-fidelity physics-based model, e.g., a finite element (FE) model, into a model that is significantly faster to run. In EM application, surrogate models can be utilised in, e.g., design optimisation, anomaly detection in condition monitoring, control, and digital twins [1]. To develop such a fast and accurate surrogate model based on physics simulation data, a vast number of simulations could be needed to run.

If the physical simulation is slow, it is desirable to minimise the number of such runs. One modern way of doing this is to allow the machine learning (ML) algorithm to decide new data points to improve the model instead of using tradition design of experiments (DoE). This is called *active learning* (AL), which is a subtype of machine learning where a learning algorithm can interactively query new data points from a user or another software [3]. In electrical machine surrogate modelling, the data source can be, e.g., an FE model of the EM. Previously, AL surrogates have been utilised, e.g., in material design [4], optimisation [5], and engineering [6]. To the best of our knowledge, our study is the first to apply active learning surrogates to any electrical engineering application.

In this paper, we develop a workflow using open-source tools for surrogate development combining active learning, a finite element solver, and cloud computation. The aim is to perform the heavy FE computation in containerised cloud, which can be interfaced from a Python environment run on a local computer. We demonstrate the workflow with a case where a surrogate model is developed to predict the average torque T, torque ripple T_{rip}, total loss $Ploss$ and current total harmonic distortion THD for an induction motor in a large range of operating points determined by supply frequency f, voltage U and motor slip s. For that, we implemented two new convergence criteria directly for the steady-state cycle-averaged torque and a scheme for variable timestep length. We aimed for high convergence robustness to have reliable results in all feasible operating points without human intervention.

Manuscript received December 1, 20XX; revised August XX, 2023. Corresponding author: Janne Keränen (email: janne.sami.keranen@vtt.fi).

The presented surrogate model can be used in, e.g., EM control to reduce losses, torque ripple, or current ripple, and to indicate faults and anomalies, if the quantities behave unexpectedly compared to the surrogate. Moreover, torque and losses together can be used to optimise a single motor operation or a large-scale industrial system that uses multiple electrical machines of different sizes and power, resulting in energy savings during the whole product lifecycle. THD of the current in drive-operated machines helps to understand the stability of the control system in the drive, and in general, indicates the stability of the machine operations and overall condition of the machine. Since the high torque ripple can be co-related with bearing health and high current ripple can be co-related with high iron losses, bearing health and operational temperature can be estimated in real-time with estimations of the torque ripple and THD of the current.

II. Induction machine cloud computing

A. *Case study motor*

Our AL workflow is based on open-source FE software Elmer. As a case study, we implement a surrogate model for a three-phase 11 kW skewed-rotor induction motor nonlinear FE multi-slice model (Figure 1). The motor has a nominal point at $f = 50$ Hz, $U = 400$ V, and $s = 0.0163$. Scalar control was implemented by keeping the U/f ratio constant under 50 Hz and using flux weakening over 50 Hz. The range for frequency was 10–100 Hz and for voltage 40–500 V so the voltage range depends on frequency. The slip had an upper limit for each frequency and voltage so that the torque did not significantly exceed the nominal torque of the motor.

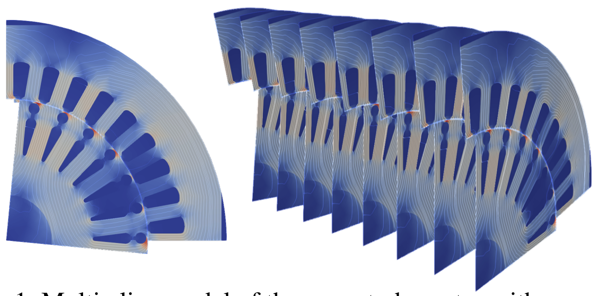
Fig. 1: Multi-slice model of the case study motor with magnetic flux density and flux line solutions. Both end and side view.

B. Robust FE computation

Since our aim is to make the FE computation a black box producing desired output from any feasible operating point without human intervention, we implement methods to make the FE solution as robust as possible. We aim at a faster FE solution time for surrogate development. For that, we utilise the parallel computation capabilities of Elmer for multi-slice modelling in the most robust way by using one MPI process for each slice, since using more processes per slice would not make the surrogate development substantially faster [7]. On the contrary, we choose to solve several Elmer runs of different operating points in parallel, and that way reach full parallel efficiency. The Elmer multi-slice FE model has been experimentally validated, e.g., in [7] with a case close to ours.

The case study model has long solution transients even when a harmonic solution is used as an initial guess. To reduce solution time, a methodology to geometrically shorten the timestep length was implemented in Elmer. The simulation was initiated with 50 timesteps per electrical period and shortened to 200 timesteps per period during four cycles in the ramp-up phase, and timestepping continued from there until the transients caused by the timestep change were attenuated. With such short timesteps also torque ripple may be studied.

In FE analysis of EMs, an excessive number of timesteps is typically solved to get sufficient accuracy for all interesting quantities. For running hundreds or thousands of FE simulations in different operation points, the convergence criteria have to be selected cleverly and the convergence needs to be reached in all operating points. In a surrogate generation, the convergence criteria should directly measure the surrogate output quantities. We ended up implementing two convergence criteria to measure the convergence of the torque. The first one measures when the torque averaged over the cycle has been converged, monitoring when the difference between the cycle-average torques of two successive cycles falls below a limit. The second one measures the variance of the torque inside each cycle and accepts the result, when the variance is less than the selected value. These criteria were selected, as we are especially interested in the steady-state average torque and the torque ripple of the motor. These criteria were observed to work as well for getting exact enough results for motor loss components and for the stator winding current harmonics.

After the solution has converged on our two criteria, one additional cycle was computed with post-processing activated, computing the total losses and stator winding current harmonic components over the work cycle. The core losses were integrated with the Bertotti model, the stator losses from electrical current, and the bar losses from eddy currents. The THD was evaluated based on the FFT of the electrical current waveform.

C. Cloud computation

The Elmer simulation was performed in CSC's Rahti Kubernetes container cloud, enabling easy and quick deployment of fast computation resources without queueing. A generic network API was built for Elmer in Rahti and a Python code was implemented to utilise the API from a local computer. For each run, new input parameters (f, U, and s) were updated into Elmer model files and sent to Rahti. After the simulation was completed, the results were available for download. Each job utilises eight parallel MPI processes and 10 jobs could be run in parallel. The computation of one job took 297.5 s on average. Figure 2 shows an example of the average torque in selected data points. To benchmark the AL method efficiently, the Rahti API was used to generate all the datasets described in the Section III before AL experiments, and samples chosen by the AL algorithm were picked from one of them called the pool. In a real setting, one directly connects the AL algorithm to the Rahti API to query new data.

III. Active learning surrogate

AL provides a data-efficient way to generate an accurate surrogate model by querying new data from an FE model. In AL, a surrogate quickly evaluates a large number of input candidates, of which the most promising, in terms of potential improvement to the surrogate model performance, are selected for querying [3], as illustrated in Figure 3. This way, the number of required data points for training an accurate ML model can be potentially reduced which is important when the FE model is computationally slow. In this study, the standard deviation of model predictions is used to select the inputs.

The initialisation of AL requires an initial dataset for training the first surrogate, which is here an ensemble of eight neural networks (NNs) or random forest (RF) models built using the Scikit-learn Python library [8]. Initial datasets of 8 and 64 samples were generated using grid-like sampling. In addition, Latin hypercube sampling (LHS) was used to generate two datasets of 100 samples for validating and testing the surrogate. To study how much the surrogate's accuracy improves with very large datasets, grid datasets of 216 and 729 samples and an LHS dataset of 1490 samples were generated. AL experiments started from 8 or 64 samples, querying one sample from a pool of 729 samples per round until the size of the training datasets was 64 and 216, respectively. In addition to the given input variables, polynomial features $s \times U$, $s \times f$, and $U \times f$ were computed and used as input to the surrogate models. All experiments were repeated five times with both model types, and each experiment was also repeated using random sampling instead of AL to compare the performance of the two.

During the sampling process, the surrogate was updated after each round and hyperparameter optimisation was performed every eight round using the Hyperopt Python library [9]. The hyperparameters options included the number of neurons (32-128), alpha (0.002-0.2), initial learning rate (0.001, 0.02),

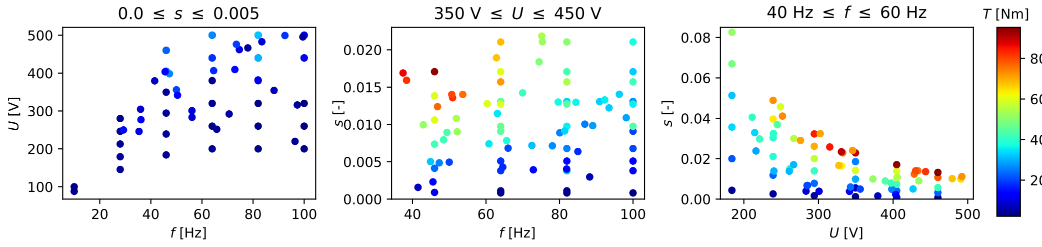

Fig. 2: FE torque in data points in three different 2-d sections of the 3-d parameter space, with the third parameter limited to a narrow range described above each subfigure. Left: s around 0.015, Middle: U around 400 V, and Right: f around 50 Hz.

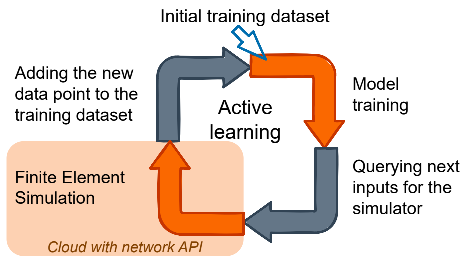
Initial training dataset
Model training
Querying next inputs for the simulator
Adding the new data point to the training dataset
Active learning
Finite Element Simulation
Cloud with network API

Fig. 3: Active learning process.

learning rate schedule (inverse scaling or adaptive), and batch size (8, 16, 32, 64, or 128) for NN, and the number of estimators (20, 40, 80, 120) for RF. In hyperparameter optimisation, each model candidate was evaluated using 6-fold cross-validation.

The model error was evaluated using root mean squared error (RMSE) in the experiments and the results in Section IV represent the average and standard deviation of RMSE of the five repetitions. The RMSEs were calculated using the test dataset to measure the generalization ability of the model.

IV. Surrogate Modelling Results

The prediction RMSEs of models trained with different datasets in Table I show that with 216 samples, active learning outperforms grid sampling (traditional DoE) except when NN is used to model T_{rip}. However, other results show that RF is better suited for modelling T_{rip}, and NN achieves lower RMSE with T, $Ploss$, and THD. The results with grid and LHS datasets of 729 and 1490 samples, respectively, show that significant improvements in RMSE can be achieved with very large datasets.

Figures 4, 5, 6, and 7 represent the evolution of the average and standard deviation of RMSE as more data is added to the training dataset. They also confirm that RF works best for T_{rip} and NN for the rest. A comparison of AL and random sampling (labelled R in the figures) shows that there is only a little difference when modelling T and $Ploss$. With T (Figure 4), the RMSE decrease slightly faster with AL but in the end, it is the same as with random sampling. However, it should be noted that the standard deviation of RMSE is much larger with random sampling than with AL.

With $Ploss$ (Figure 5), the RMSE with random sampling remains slightly lower throughout the sampling process than with AL. The predictions of the best model for T and $Ploss$ with datasets of 64 samples are visualised as coefficient of determination (R^2) plots in Figures 8 shows that the error is at a sufficient level. R^2 is the percentage of variance in the output

TABLE I: RMSEs of models trained with different datasets.

		A64[a]	G64[b]	A216	G216	G729	L1490[c]
T	NN	1.83	1.89	1.32	5.72	0.40	0.38
[Nm]	RF	10.41	9.90	6.50	8.18	3.87	2.81
$Ploss$	NN	50.67	66.46	34.95	92.38	14.05	10.47
[W]	RF	167.56	210.12	126.99	187.63	83.09	55.81
T_{rip}	NN	1.92	4.09	2.36	2.18	1.68	0.80
[Nm]	RF	1.37	1.28	0.99	1.18	0.96	0.44
THD	NN	1.44	1.26	0.67	1.51	0.43	0.33
[-]	RF	1.83	1.49	0.96	1.31	1.01	0.43

[a] Active learning. [b] Grid sampling. [c] LHS.

that is explained by the inputs. The standard deviation of $Ploss$ RMSE is approximately equal with both random sampling and AL.

Since the RMSE of T_{rip} and THD was not sufficient with datasets of 64 samples, the AL process was repeated for them starting from 64 samples and sampling until 216 samples were reached. The results for T_{rip} (Figure 6) show that the RMSE decreases from 1.45 Nm to 0.7 Nm on average. Its RMSE decreases rapidly when AL starts but then the standard deviation of RMSE increases and the average RMSE stalls before decreasing again at around 155 samples. The results for THD (Figure 7) show that the RMSE decreases from 1.28 Nm to 0.99 Nm on average but there are three sudden increases in the RMSE during the AL. The reason for such behaviour is to be studied more closely in future work.

The R^2 plots (Figure 9) show that even with 216 samples, the variance of T_{rip} predictions is still relatively high, whereas the NN achieve low error for THD. However, Table I and Figure 9 show that modelling T_{rip} accurately requires a large number of training data points. Its RMSE decreases by a factor of 2.7 when the amount of training data is increased by a factor of 6.9 (A216 vs. L1490).

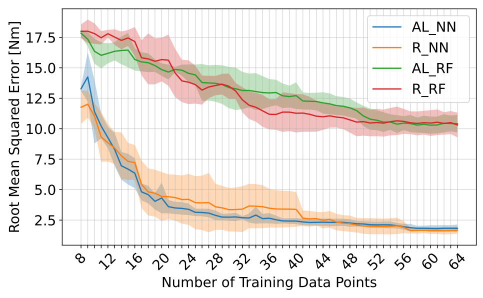

Fig. 4: Evolution of the torque prediction error (8–64 points).

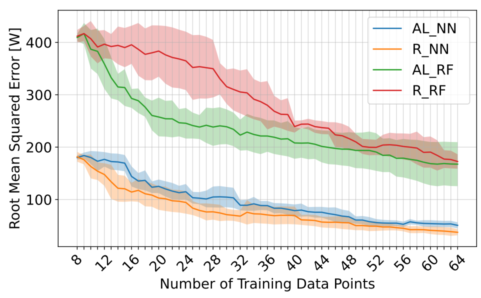

Fig. 5: Evolution of the loss prediction error (8–64 points).

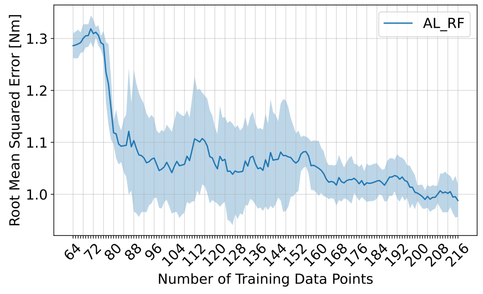

Fig. 6: Evolution of the torque ripple error (64–216 points).

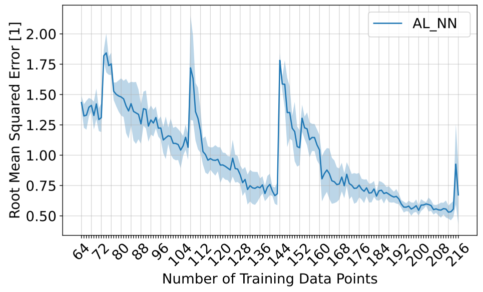

Fig. 7: Evolution of the THD error (64–216 points).

The CPU times to predict one sample with an NN and an RF model were 0.54 ms and 0.32 ms, which correspond to predicting 1866 and 3110 samples per second, respectively. Therefore, the speed was approximately 0.5–1 million times greater than the FE simulation. In the surrogate model generation, FE simulations dominate the time consumption, which is determined by the number of required training data points.

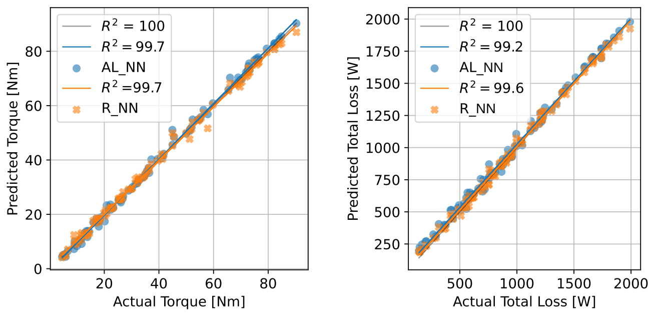

Fig. 8: Predicted torque and total loss.

V. Conclusion

We presented a novel active learning workflow for generating FE data for surrogate model development in a robust and rapid way without human intervention. Induction motor torque, torque ripple, loss, and stator winding current THD estimation results were presented, where active learning was recognised to improve the surrogate accuracy, at least with a low amount of data. However, the difference to the random sampling was not substantial, which might be due to the test case with only three-dimensional parameter space, but the reason should be further investigated. Modelling torque ripple accurately required a substantially larger training dataset than the rest of the outputs, and it was also the only studied output variable that could be predicted more accurately with an RF than with an NN model. However, it was noticed that the use case needs to be quite complex in order to active learning to outperform traditional DoE in the required FE simulation time.

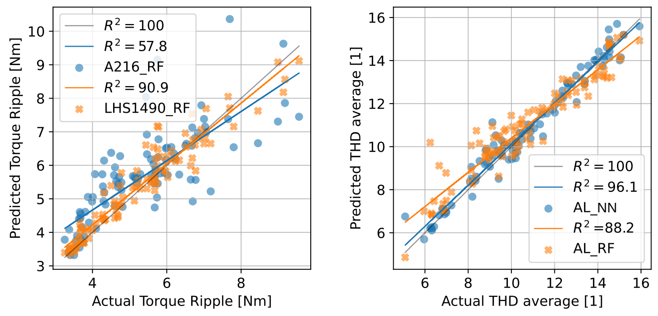

Fig. 9: Predicted torque ripple and current THD.

Acknowledgments

This work was supported by the Centre of Excellence in High-Speed Electromechanical Energy Conversion Systems (HiECSs), funded by Academy of Finland (GN 346441), and Arrowhead Tools project, funded by ECSEL-JU, European Commission, and Business Finland (GA 826452).

References

[1] M. Tahkola, J. Keranen, D. Sedov, M. F. Far, and J. Kortelainen, "Surrogate Modeling of Electrical Machine Torque Using Artificial Neural Networks," *IEEE Access*, 2020.
[2] M. Farzam Far, F. Martin, A. Belahcen, L. Montier, and T. Henneron, "Orthogonal Interpolation Method for Order Reduction of a Synchronous Machine Model," *IEEE Transactions on Magnetics*, vol. 54, no. 2, 2018.
[3] S. Chabanet, H. Bril El-Haouzi, and P. Thomas, "Coupling digital simulation and machine learning metamodel through an active learning approach in Industry 4.0 context," *Computers in Industry*, vol. 133, 2021.
[4] R. Pestourie, Y. Mroueh, T. V. Nguyen, P. Das, and S. G. Johnson, "Active learning of deep surrogates for PDEs: application to metasurface design," *npj Computational Materials*, vol. 6, no. 1, 2020.
[5] H. Wang, Y. Jin, and J. Doherty, "Committee-Based Active Learning for Surrogate-Assisted Particle Swarm Optimization of Expensive Problems," *IEEE Trans. on Cybernetics*, vol. 47, no. 9, pp. 2664–2677, 2017.
[6] H. Vardhan, U. Timalsina, P. Volgyesi, and J. Sztipanovits, "Data efficient surrogate modeling for engineering design: Ensemble-free batch mode deep active learning for regression," 2022. [Online]. Available: https://arxiv.org/abs/2211.10360v1
[7] J. Keränen, P. Ponomarev, J. Pippuri, P. Råback, M. Lyly, and J. Westerlund, "Parallel Performance of Multi-Slice Finite-Element Modeling of Skewed Electrical Machines," *IEEE Trans. on Magn.*, vol. 53, no. 6, 2017.
[8] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
[9] J. Bergstra, D. Yamins, and D. Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," in *Proceedings of Machine Learning Research*, vol. 28. Atlanta, USA: PMLR, 17–19 Jun 2013, pp. 115–123.