ESTIMATING BALL-BEARING FRICTION

To date, Machine Learning (ML) has found only relatively limited implementation in high-end mechatronic systems. To test its capabilities in a real industrial application, the challenging case of friction estimation was investigated. A model was developed for predicting frictional properties of linear ball bearings, for very small displacements. The resulting ML model performed well during training and validation, but rather less so in stand-alone operation. ML is a promising tool for friction estimation, but clearly there is room for improvement in algorithm development.

JORN VEENENDAAL

Introduction

The field of Machine Learning (ML), a subset of Artificial Intelligence (AI), has seen impressive development and growth over the last decade. ML's astonishing rise to prominence within many disciplines is no doubt aided by its numerous impressive accomplishments, building on the ability of ML algorithms to improve themselves automatically through acquiring and processing experience [1]. Examples are DeepMind's AlphaGo algorithm that defeated the world champion in the game of go in 2016, OpenAI's ChatGPT's natural language processing capability, and DeepMind's AlphaFold that solved the long-standing protein folding problem in 2020.

Despite its many achievements, ML has seen only relatively limited, cautious implementation in the field of mechatronics. For this reason, MI-Partners, which specialises in the development of high-end mechatronic systems, decided to start a project to test the current capabilities of ML in a real industrial application. The research question prompted by this ambition was directed towards a practical use case: can an ML model reliably predict the friction of a linear ball bearing?

This article describes the development of an ML algorithm for this use case. First, the friction model is elaborated, followed by a discussion of the learning strategy and the algorithm selection, after which the detailed algorithm design is presented. To conclude, the algorithm is tested, with a direct comparison with the friction model concerned, the LuGre model.

Modelling friction

Currently, a Lund-Grenoble (LuGre) [2] friction model is a good method of modelling the friction behaviour of linear ball bearings. Although the model is accurate when well tuned, the reason for undertaking this investigation was to discover whether it is possible for an ML algorithm to approach the performance of the LuGre model predictions, and possibly overcome the LuGre shortcomings in parameter sensitivity and inability to model advanced friction properties, such as a change in viscosity or the creation of oil bumps.

The LuGre model, as described by Equations 1 to 3, builds upon the principles of the Dahl friction model [3]. It is governed predominantly by the relative velocity $v(t)$ and the hidden state variable $z(t)$. This $z(t)$ cannot be measured practically, but it can be interpreted as the averaged bristle deformation (see Figure 1), where a bristle describes a small connection between two sliding objects.

The bristle deformation, partially described by $g(v)$ in Equation 3, is based on constants such as the Coulomb and Stribeck force, F_c and F_{st}, respectively. Combining the scaled contribution of the bristle deflection and its derivative with respect to time, $\dot{z}$, with the viscous damping term $f(v) = \sigma_2 v$ defines the total predicted friction F_f and with it the benchmark for the ML friction model.

$$\dot{z} = v - \sigma_0 \frac{|v|}{g(v)} z \tag{1}$$

$$F_f = \sigma_0 z + \sigma_1 \dot{z} + f(v) \tag{2}$$

with

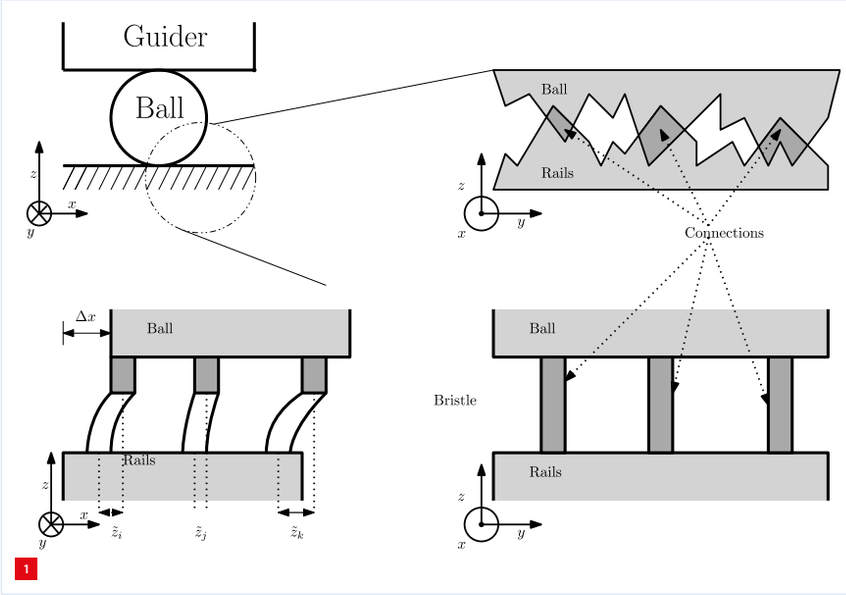
$$g(v) = F_c + (F_{st} - F_c) e^{-\left(\frac{v}{v_{st}}\right)^2} \tag{3}$$

Here σ_0, σ_1 are the material property coefficients for stiffness and damping, respectively. The constant v_{st} represents the characteristic velocity of the velocity-friction force.

AUTHOR'S NOTE

Jorn Veenendaal is a graduate student in Systems and Control at Eindhoven University of Technology, Eindhoven (NL). He studied Mechatronics Engineering at Fontys Engineering University of Applied Sciences in Eindhoven and performed his graduation work at MI-Partners in Veldhoven. For his thesis he received a nomination for the Wim van der Hoek Award 2021. He acknowledges the support and feedback of Wilco Pancras, software architect at MI-Partners.

jt.veenendaal@hotmail.com
www.mi-partners.nl



Graphical representation of bristle deformation [4].

Machine Learning strategy

ML algorithms are often characterised by their learning strategy. There are three main learning strategies and the differences between them derive from the way they allow an ML algorithm to extract patterns from a particular data set.

The first strategy concerns supervised learning, which encompasses algorithms that learn to improve their output predictions based on some input by comparing the output to a known true answer in a given training set [5] and using the resulting error to improve the ML model.
Another strategy involves unsupervised learning, its aim being to unravel the structure that underlies the given set of data [6]. Finally there is reinforcement learning, which distinguishes itself from the other strategies by its emphasis on an agent learning by direct interaction with its environment, without relying on exemplary supervision or complete models of the environment [7].

In the current use case, the nature of the generally well-understood phenomenon of friction did not favour the use of unsupervised strategies. That is because unsupervised learning specialises mainly in finding the main features that govern the observed behaviour in a data set. As centuries of research into the subject of friction have already provided most of the relevant information, it would be unwise (and inefficient) to neglect this knowledge. Furthermore, the required input and output data for the friction use case could be generated on demand via the LuGre model or physical experiments. This would create an abundance of representative training data sets.

Likewise, reinforcement learning was deemed a plausible but more complex and demanding solution compared to supervised learning. Altogether, supervised learning was selected as the most promising strategy for the friction use case, which paved the way for the next step in the project: algorithm selection.

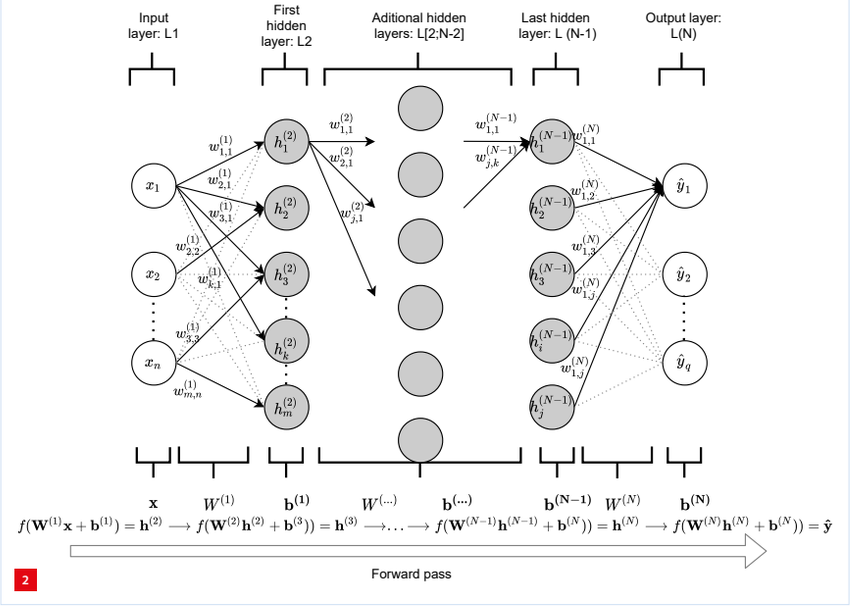
Algorithm selection

Within the strategy of supervised learning [8], three different algorithms were considered as serious candidates for this use case. The algorithms considered – K-Nearest Neighbour (KNN), Support Vector Regression (SVR) and Artificial Neural Networks (ANN) – are all well suited for the inherent regression problem type of this use case.

An ANN-based algorithm (for a schematic representation see Figure 2) was deemed best suited. ANNs are proven universal function approximators [9]. It is this property, combined with various examples that show they can solve differential equations [10] [11] and systems with hysteresis [12], that made ANNs a good candidate. Although the KNN and SVR were also considered, neither of these options was commonly associated with this type of problem, which made them a less promising solution compared to the ANN algorithm. With the algorithm selected, the next logical phase in the project was the detailed design.

Detailed design

Four sequential steps were taken for the design of the final ML algorithm: model architecture selection, data creation, model regularisation and hyperparameter tuning.
For the ML-model architecture, it was concluded that the ML model should have two inputs: the current velocity of the reference profile imposed on the system with friction,



Schematic representation of the main operating principle of an ANN: an interconnected group of perceptrons (artificial neurons) that are typically aggregated into layers. The mathematical representation on the bottom includes the nonlinear activation function f(·), which produces the output per layer.

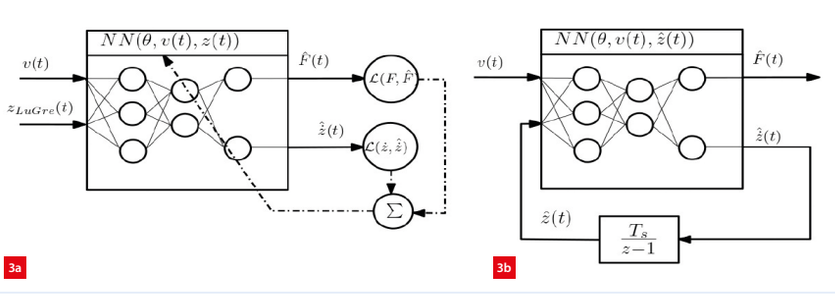
and the current average bristle deflection. These choices were inspired by the LuGre model, in which these input parameters are needed to generate the output friction force.

In addition to this output force, it was concluded that the ML model needs to predict the derivative of the average bristle deflection. This additional output would enable the ML model to become self-sufficient by providing its own next bristle deflection input through the discrete integration of the $\dot{z}$ variable. This eliminates the need to learn a solution for a nonlinear differential equation, enabling the ANN to focus on finding the already challenging nonlinear mapping $F_f : (v, z) \rightarrow (\hat{F}, \hat{z})$, with the ^ symbol indicating the predicted values of the ANN.

In practice, this means that the ML model needs two configurations: training and operational (see Figure 3). In the training configuration, the model will use average bristle deflection calculated by the LuGre model as its input, combined with the velocity of the reference profile, while its output parameters will be compared to the friction force and deflection derivative of the LuGre model. When it is put in general use, however, it only requires input velocity and a feedback loop with a discrete integration of its hidden state variable $\hat{z}(t)$.

This learning process requires data; ML models can only learn the properties of a system through the data they are presented with, making their performance very sensitive to the quality of the data set. This use case required three specific data sets: training, validation and falsification.

The training data set, derived from the LuGre model, was a specific cloud of random points, selected in order to teach the ML model as many different input-output relationships as realistically feasible. For its validation data set, an arbitrarily selected reference profile was chosen to test the model's performance in normal operation. This provided a good indication of the ML model's performance on a realistic reference profile during the training and validation process (Figure 3a).



Friction use case architecture.
(a) The training configuration, which uses the $z_{LuGre}(t)$ as its external input.
(b) The operational configuration, which uses the integrated $\hat{z}(t)$ as its next (internal) input.

The last data set would involve testing the ML model in an operational configuration (Figure 3b). The performance of the model would be assessed by comparing its results to the LuGre model, allowing for a reproducible and isolated testing environment. The final test of the ML model, involving a new realistic data set generated by the LuGre model, can however only falsify the model performance on the tested situation, as complete performance guarantees are not plausible due to the black-box nature of the ANN.

With an ANN, model optimisation involves tuning the ANN's hyperparameters; this refers to the configuration of the model as defined by its shape, size and other parameters that control the algorithm's behaviour. Manually tuning them is usually based on previous work and/or (extensive) experience in the field [5]. For this reason, a more systematic approach was employed to find a good ANN model configuration.

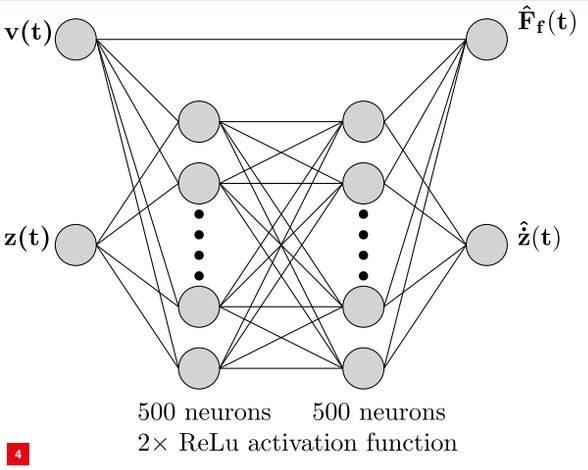
The core of the approach is based on an automatic Bayesian hyperparameter optimisation algorithm. The working principle is based on sampling the performance of different ANN models with different hyperparameters and determining the next sampled model configuration based on a Bayesian optimisation [13].

The hyperparameters chosen for tuned optimisation in this use case are the number of ANN layers, the size of each layer, the activation functions, and the learning rate (the 'step size' in learning, i.e. the extent to which new information overrides old information). As additional hyperparameters, different regularisation techniques and configurations are also included in the optimisation of the ANN configuration. Regularisation techniques are included because they aid in preventing the ANN from overfitting its predictions to the training data set, in order to make its predictions more robust during operation. All in all, this hyperparameter optimisation ultimately determined the final form of the model, as graphically represented in Figure 4.

Results

The ML model's performance as compared to the LuGre model was assessed by putting the final model to the test using the data sets described above. The results are presented as error histograms in Figure 5 and are summarised in Table 1, with the error mean μ and the minimum and maximum prediction errors. Here, the error is the difference between the ML model output and the LuGre output value.

It is evident that the training and validation results (Figures 5a and 5b) remain well within the desired force prediction bounds of ±1 N, averaging to a prediction precision that is nearly 100 times better, while the derivative of the bristle



4

Simplified graphical overview of the complete ANN model. ReLu stands for rectified linear unit.

deflection was on average 200 times more accurate than the desired ±0.033 m/s. Here the force and average bristle deflection derivative requirements have been derived from acceptable errors during normal use in a practical project.

These favourable results were nevertheless not maintained when testing the algorithm in the operational configuration during the falsification test (Figure 5c). The prediction precision compared to the LuGre model showed that only 26.7% of the prediction errors were between −3.37 N and 2.63 N, which does not meet the requirement. In addition, the distribution of the error cannot be regarded as a normal distribution. This would suggest that there are still a few important properties not yet learned by the algorithm. For the derivative of the bristle deflection, the falsification error more closely approximates a normal distribution with approximately all errors between ±220 µm/s, meeting the desired requirements.

A possible explanation for the difference in accuracy between the training/validation and falsification test could be found in the integration of small errors. These errors compound over time into a larger deviation in the $\dot{z}$ predictions, degrading their performance. Higher accuracies in $\dot{z}$ predictions or reset possibilities could mitigate this phenomenon.

Table 1
ANN results: error mean µ and minimum and maximum prediction error for F and $\dot{z}$.

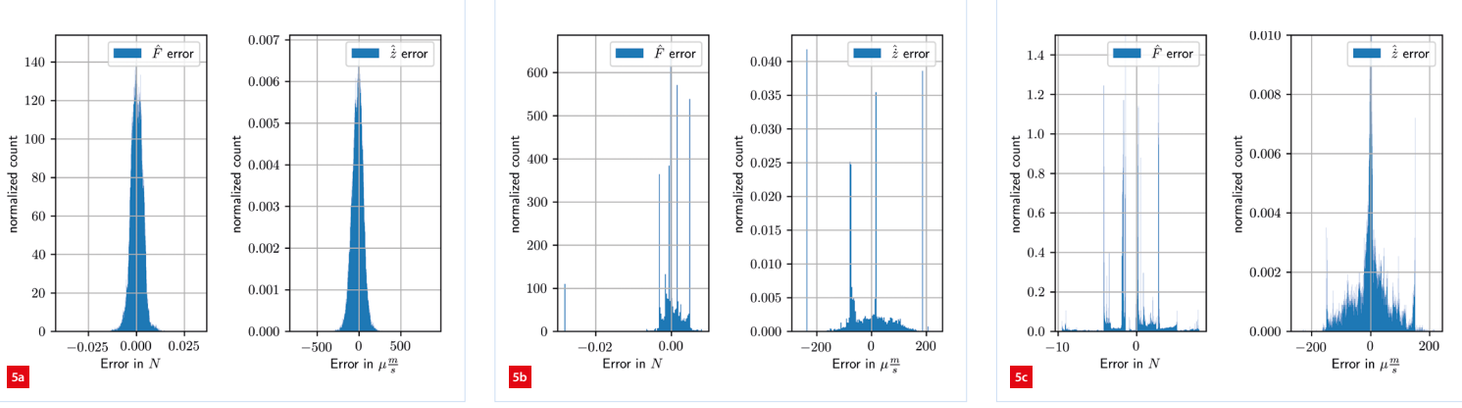
Data set	Error in F (N)		Error in $\dot{z}$ (µm/s)	
	μ	[min; max]	μ	[min; max]
Training	$+2.90 \cdot 10^{-4}$	$[-3.85; 3.30] \cdot 10^{-2}$	−17.42	[−748.05; 896.25]
Validation	$-1.33 \cdot 10^{-4}$	$[-2.83; 0.81] \cdot 10^{-2}$	−18.41	[−266.52; 233.81]
Falsification	$-3.95 \cdot 10^{-1}$	[−9.51; 7.99]	−0.183	[−246.68; 219.66]

Conclusion

This project has introduced an industrial use case for Machine Learning (ML) in the context of high-end mechatronics. The chosen LuGre friction model proved to be a suitable problem for ML with room for real improvement when striving to address the unmodelled friction phenomena in the LuGre model. The subsequent structured ML algorithm design process – featuring ML strategy selection, algorithm selection and algorithm design – resulted in a final ML model.

This model, based on an Artificial Neural Network (ANN) algorithm combined with a discrete integrator, had its trainable parameters optimised – based on a supervised learning strategy – on a data set created by a LuGre model. The model's hyperparameters were determined through Bayesian optimisation in pursuit of a good model configuration.

The resulting model showed favourable performance during training and validation, but exhibited a considerable



ANN results: error histograms for F (left) and $\dot{z}$.
(a) Training data set.
(b) Validation data set.
(c) Falsification data set.

decrease in precision when operated in its intended stand-alone operating configuration. Ultimately, the model failed to achieve the desired force prediction precision of ±1 N and thus failed to provide clear performance guarantees. As a result, the algorithm offered only an incomplete answer to the project's research question: can an ML model reliably predict the friction of a linear ball bearing?

All in all, this project has demonstrated that ML requires a significant design process in order to address certain challenges within the field of high-end mechatronics. Per contra, it did show a practical way of working for ML-model development and strong prediction capability in certain situations. In addition, the algorithm itself still shows room for improvement, with the algorithm's interaction with discrete integration of noisy data or the integration of physics-informed learning [14] being two options that could enable an implementation with experimental data.

REFERENCES

[1] T.M. Mitchell, *Machine Learning*, McGraw-Hill, Inc., USA, 1st edition, 1997.
[2] C. Canudas De Wit, H. Olsson, K.J. Astrom, and P. Lischinsky, "A new model for control of systems with friction", *IEEE Transactions on automatic control*, 40 (3), pp. 419-425, 1995.
[3] P. Dahl, *A solid friction model*, Aerospace Corp., El Segundo, CA, Tech. Rep. TOR-0158(3107-18)-1, 1968.
[4] D.A. Haessig Jr., and B. Friedland, "On the modeling and simulation of friction", *ASME J. Dyn. Sys., Meas., Control*, 113 (3), pp. 354-362, 1991.
[5] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016, *www.deeplearningbook.org*
[6] S. Theodoridis, *Machine learning: a Bayesian and optimization perspective*, 2nd edition, Academic Press, 2020.
[7] R.S. Sutton, and A.G. Barto, *Reinforcement learning: An introduction*, MIT Press, 2018.
[8] *en.wikipedia.org/wiki/Supervised_learning*
[9] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators", *Neural networks*, 2 (5), pp. 359-366, 1989.
[10] S. Mall, and S. Chakraverty, "Hermite functional link neural network for solving the Van der Pol-Duffing oscillator equation", *Neural Computation*, 28 (8), pp. 1574-1598, 2016.
[11] J. Berg, and K. Nyström, "A unified deep artificial neural network approach to partial differential equations in complex geometries", *Neurocomputing*, 317, pp. 28-41, 2018.
[12] C. Serpico, and C. Visone, "Magnetic hysteresis modeling via feedforward neural networks", *IEEE Transactions on Magnetics*, 34 (3), pp. 23-628, 1998.
[13] J. Snoek, et al., "Scalable bayesian optimization using deep neural networks", *Proc. 32nd Int'l Conf. Machine Learning*, PMLR 37, pp. 2171-2180, 2015.
[14] M. Raissi, P. Perdikaris, and G. Em Karniadakis, "Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations", *arxiv.org/pdf/1711.10561v1.pdf*, 2017.