BULLETIN OF THE POLISH ACADEMY OF SCIENCES
TECHNICAL SCIENCES, Vol. 71(6), 2023, Article number: e147061
DOI: 10.24425/bpasts.2023.147061

SPECIAL SECTION

Data-driven virtual sensor for powertrains based on transfer learning

Aku KARHINEN[1]*, Aleksanteri HÄMÄLÄINEN[1], Mikael MANNGÅRD[2], Jesse MIETTINEN[1], and Raine VIITALA[1]

[1] Department of Mechanical Engineering, Aalto University, 02150, Espoo, Finland
[2] Novia University of Applied Sciences, Juhana Herttuan puistokatu 21, 20100 Turku, Finland

Abstract. Variation in powertrain parameters caused by dimensioning, manufacturing and assembly inaccuracies may prevent model-based virtual sensors from representing physical powertrains accurately. Data-driven virtual sensors employing machine learning models offer a solution for including variations in the powertrain parameters. These variations can be efficiently included in the training of the virtual sensor through simulation. The trained model can then be theoretically applied to real systems via transfer learning, allowing a data-driven virtual sensor to be trained without the notoriously labour-intensive step of gathering data from a real powertrain. This research presents a training procedure for a data-driven virtual sensor. The virtual sensor was made for a powertrain consisting of multiple shafts, couplings and gears. The training procedure generalizes the virtual sensor for a single powertrain with variations corresponding to the aforementioned inaccuracies. The training procedure includes parameter randomization and random excitation. That is, the data-driven virtual sensor was trained using data from multiple different powertrain instances, representing roughly the same powertrain. The virtual sensor trained using multiple instances of a simulated powertrain was accurate at estimating rotating speeds and torque of the loaded shaft of multiple simulated test powertrains. The estimates were computed from the rotating speeds and torque at the motor shaft of the powertrain. This research gives excellent grounds for further studies towards simulation-to-reality transfer learning, in which a virtual sensor is trained with simulated data and then applied to a real system.

Key words: powertrain; torsional vibration; long short-term memory;virtual sensor; transfer learning.

1. INTRODUCTION

Accurate monitoring and control are essential for lifetime optimization of different powertrains across multiple industries. To monitor and control powertrains accurately, many measurements are needed from all over the powertrain to have as much information as possible. Typical physical sensors employed to monitor powertrains include accelerometers, rotary encoders and torque transducers. With such sensors, torques and rotating speeds of different parts of the powertrain can be employed, for example to perform an analysis of powertrain loading dynamic characteristics and fatigue damage [1]. Applying such analysis results, timely maintenance actions can be taken. However, measurements of this scope on powertrains require large amounts of physical sensors, naturally reducing the cost-effectiveness of powertrain engineering. Large simultaneous use of sensors also increases the risk of sensor malfunction. Furthermore, physical measurements of these wanted quantities may be prevented for many reasons. For example, the wanted quantity can be difficult to measure during operation because of geometrical limitations of the powertrain or the operating environment.

Virtual sensors, also known as soft sensors, appear as a promising branch of solutions well suited for machine monitoring. Virtual sensors are computational models that estimate physical quantities from other available information about the system, for example secondary measurements or known physical quantities. Virtual sensors have attracted much attention across multiple research fields, including process monitoring and process fault diagnosis. Additionally, virtual sensors have recently gained some attention in mechanical application fields, for example in machine condition monitoring [2]. Virtual sensors are often classified either as model-based virtual sensors, or as data-driven virtual sensors [3]. Model-based virtual sensors apply given system dynamics and domain knowledge, for example in the form of a physics model that simulates the system dynamics, to determine the wanted output quantity. Data-driven virtual sensors on the other hand apply machine learning (ML) techniques to learn patterns directly from historical data without any domain knowledge.

Commonly known model-based virtual sensor implementations include methods such as state estimators [4], Kalman filters [5], extended or augmented Kalman filters [6–8] and inverse models [9,10]. However, the functionality of model-based virtual sensors is limited to the system they are built around. For example, slight differences in the powertrain due to manufacturing or assembly inaccuracies may hinder the accuracy of model based virtual sensors. That is, a virtual sensor might estimate the desired measures of a powertrain inaccurately if properties such as stiffness, damping, loads, excitations or contacts have been modelled inaccurately. Furthermore, classical model based implementations may struggle with non-linear dynamics

*e-mail: aku.karhinen@aalto.fi

Manuscript submitted 2023-04-28, revised 2023-08-11, initially accepted for publication 2023-08-15, published in December 2023.

in these conditions. For example, extended Kalman filters are known to be suboptimal. Therefore, creating a general model-based virtual sensor for a certain type of powertrain can prove difficult and inaccurate.

Traditionally, data-driven virtual sensors are optimized with historical data from the system [3], thus requiring measured data for training. They however require little prior knowledge of the system dynamics. Traditional ML-based implementations of data-driven virtual sensors include models such as multilayer perceptrons (MLP) [11] and support vector machines (SVM) [12]. A disadvantage of traditional ML-based virtual sensors is the necessity for complex feature extraction and manual feature selection from the training data. Deep learning has been outperforming traditional ML methods in applications such as natural language processing [13] and image classification [14] and it has become popular in mechanical applications such as rotating machinery diagnosis [15]. Unsurprisingly, many current data-driven virtual sensors are based on better-performing deep learning architectures [16].

While there has been an increase in popularity of data-driven virtual sensors [17] and some proposed applications in the field of rotating machinery [18], data-driven virtual sensors for powertrains are few. Furthermore, slight differences between different powertrain instances of the same powertrain design give rise to a demand for a generalized virtual sensor, that can take these differences into account and function within the required accuracy range without the need for further training. Potential techniques useful for generalization can be found in transfer learning, where patterns from a source domain are learned and then applied in the target domain. Depending on the use case, the different domains can vary largely, or be almost identical. For example, transfer learning techniques can be used to train a deep reinforcement learning-based autonomous robot entirely with simulated data, so that it can function in a real environment.

This research presents a training procedure for a data-driven virtual sensor, which generalizes the virtual sensor to work with powertrains with variations. The variations introduced in this research were drawn from a continuous uniform distribution with minimum and maximum values 0.5 and 1.5 times the nominal value. The proposed training procedure includes a technique commonly known as domain randomization. The randomized data domains are the parameters for the powertrain model and the excitations applied to the rotating powertrain. In contrast, model-based estimation with unknown input-excitations requires models of the input excitations [7]. Generating the training data with a simulated model simply allows us to simulate a variety of excitations. Additionally, parameter uncertainty has classically been handled by robust-estimation methods. Such approaches, although robust to uncertainty, may often lead to conservative estimates. The data-driven approach provides alternative methods to these. The presented data-driven virtual sensor is based on long short-term memory (LSTM), which is a type of recurrent neural network (RNN), capable of learning patterns from sequential data over a period of time.

The contributions of this paper are the following:

- We propose an LSTM-based data-driven virtual sensor estimating rotating speeds and torque from the lower parts of the powertrain, accurately given rotating speeds and torque from the upper parts of the powertrain, which has been generalized to work with different powertrain parameter configurations.
- We showcase the use of the transfer learning technique domain randomization in the form of parameter randomization and random excitation and their separate and combined effect on the virtual sensor performance.

2. THEORY AND WORKING PRINCIPLES

This section presents theoretical background for understanding the later presented virtual sensor architecture and the two transfer learning techniques applied.

2.1. Long-short-term memory

RNNs have become popular for working with time-series data due to their ability to learn temporal patterns [19]. RNNs learn temporal patterns by processing each sampled time-instant separately and taking into account information from the previous time instants sequentially. LSTM [20] is an RNN architecture popularly applied to long time-series data samples. LSTM structure and functioning principles are based on two signals called the cell state and the hidden state. These signals allow the LSTM to store historical data and make further predictions based on the data. This is controlled by a gating mechanism consisting of four gates that control the information flow to and from the cell and hidden state. These four gates are commonly known as the Input, Output, Forget and Update gates. At every time step t, the hidden state is updated by adding data from the same time step, from the aforementioned gates, the cell state and the hidden state of the last step before the current one. Equations that describe the gates of a LSTM cell over a single time step are shown in the following equations (1)–(6),

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f), \tag{1}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i), \tag{2}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o), \tag{3}$$

$$\tilde{c}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c), \tag{4}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t\,, \tag{5}$$

$$h_t = o_t \odot \tanh(c_t), \tag{6}$$

where f_t is the forget gate, i_t is the input gate, o_t is the output gate, c_t is the cell state h_t is the current hidden state and h_{t-1} is the last time step. Possibly, bias terms marked with a b may also be added to each gate value. W corresponds to the weight of each gate and $[h_{t-1}, x_t]$ corresponds to the concatenation of the input x_t and the previous hidden state h_{t-1}. *Tanh* represents a hyperbolic tangent, $\odot$ the element-wise product and σ the activation function.

Figure 1 shows these four gates in green with the corresponding activation functions labelled on top of each block. The circles and ellipses highlighted in orange correspond to point-wise operations shown in equations (5) and (6), where X means multiplication and + summation.

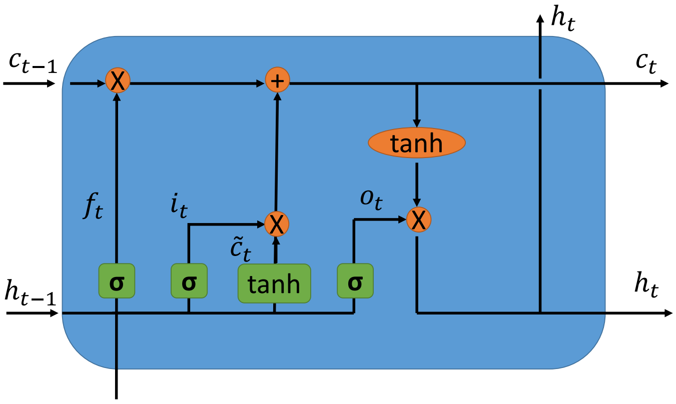


Fig. 1. Computation graph of an unfolded LSTM cell over a single time step. *Tanh* represents the hyperbolic tangent, X the element-wise product, σ the sigmoid function and $+$ the element-wise sum. X_t represents the input features of this time step, f_t is the forget gate, i_t is the input gate, o_t is the output gate, c_t is the cell state, c_{t-1} is the cell state from the last time step, h_t is the current hidden state and h_{t-1} is the hidden state brought from last time step

2.2. LSTM as a virtual sensor

LSTM architectures have seen a rise in popularity in data-driven virtual sensor applications [21]. This may be due to their ability to process large sequences of data at a time, process temporal data and find temporal patterns from the sequences of data. Many industrial applications for virtual sensors take in measured sensory data, either raw or modified, as input for the sensor. Raw sensory data is typically a time-series signal, suggesting the use of an RNN architecture such as the LSTM. For example, the LSTM architecture has been used in virtual sensors conducting quality prediction of industrial processes [21] and helicopter gearbox monitoring [22]. Furthermore, even simpler RNN architectures have been popularly used as a virtual sensor. Some use cases include estimating the contact area that tires of a car are making with the ground [23] and estimating the melt-flow-length in an injection moulding process [24].

2.3. Domain randomization

Domain randomization is a set of methods commonly used in simulation-to-reality transfer learning for deep reinforcement learning. Reinforcement learning is a subsection of machine learning, where the goal is to learn an optimal policy, or a set of actions, that will lead to the highest reward [25]. Deep reinforcement learning applies deep neural networks to accomplish this task.

The goal of domain randomization is to expose the ML model in training to randomized variation in its properties, so that it is able to generalize to the target domain despite the bias between the source and target domains [26, 27]. In sim-to-real in the field of robotics, it can be seen as a form of regularization that prevents the agent from basing its policy solely on the parameters of an individual simulated instance [28]. For example, one application in robotics is to add visual randomization to the simulation, so that the neural network learns to accomplish its task in different lighting conditions [29]. Furthermore, domain randomization can be applied in principle to any property [30]. For example, it can be applied directly to physical quantities, possibly making the trained virtual sensor more generalized to some variation in the system physical properties.

3. METHODS

This section presents the methods used in this study. First, the data acquisition is explained. Second, the presented virtual sensor architecture and the training algorithm are introduced. Lastly, the testing procedure is explained.

3.1. Data acquisition

Data-driven virtual sensors require data from the system to be trained. The variation required for this study was introduced in a simulated environment, but the underlying function of the introduced virtual sensor was not model-based. That is, all the used data for this study was gathered from a simulated environment, but the virtual sensor had no domain knowledge in itself, and only learned patterns directly from the training data. Data for the training and testing were generated with a simulation model applying a lumped mass model of a powertrain introduced in an earlier study, which had been parameterised to resemble a test bench of a miniature-sized azimuth thruster as its nominal values preceding variation [31]. The underlying topology of the simulation model was not changed in this study: only the parameters experienced variation. The test bench, and thus the simulation model, were built to emulate and simulate a thruster operated in ice-load conditions [32]. While the simulation modelled the physical test bench, actual measurements from it were not used. By using a simulation model modelled after a real test bench, this work is more likely to function with future work on sim-to-real applications. The lumped-mass model is showcased in Fig. 2 and nominal values of the lumped-mass model are shown in Table 1. A graphical representation of the miniature sized thruster test bench is shown in Fig. 3. The simulated model had been used in Kalman filter-based torque estimation of the test bench, which yielded promising results, motivating its use in data-driven applications as the source for the training data as well [31].

The dataset consisted of continuous time domain simulation data from 61 powertrain instances with domain-randomized physical parameters for the modelled powertrain. For each pow-

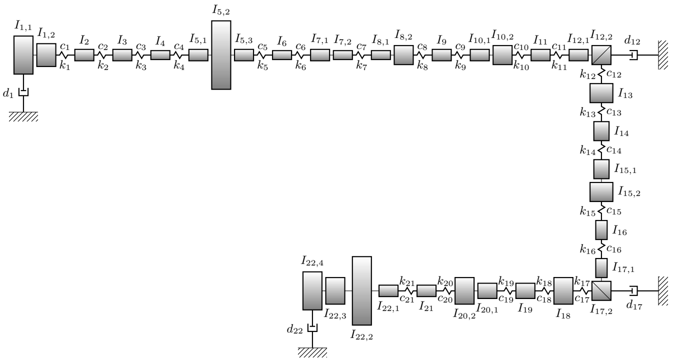


Fig. 2. Lumped-mass model [31] used in the simulation of the test bench

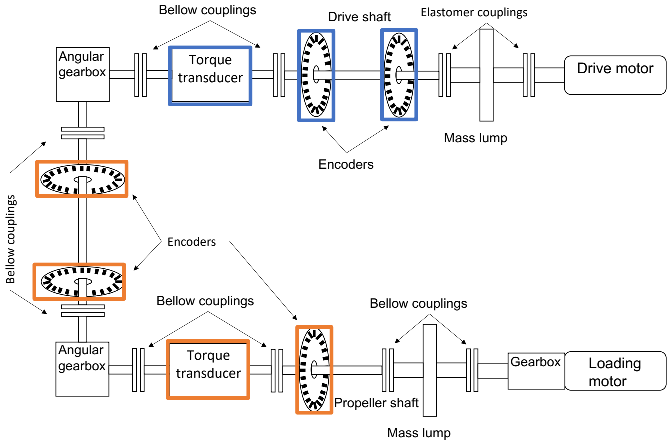


Fig. 3. Block diagram of the topology used in a miniature sized test bench of a full-scale azimuth thruster [32]. The simulated model of the powertrain matches the test bench topology, including the placement of sensors. That is, encoder and torque data were generated in corresponding places from the simulation compared to the physical test bench sensor placement seen in the topology: places marked blue correspond to the input values of the virtual sensor and places marked orange to the estimated quantities

ertrain instance, the simulation randomized physical properties of the components of the powertrain and added predetermined and randomized excitation during data acquisition, making acquired data from each powertrain instance unique. Randomized physical properties included the mass moment of inertias, damping parameters, and stiffnesses present in the powertrain model. Each randomized property was sampled from a continuous uniform distribution with minimum and maximum values 0.5 and 1.5 times the nominal value. The nominal values are shown in Table 1.

During data generation, excitation was presented in the generated measurement data. There were three types of excitation: acceleration, ice excitation and random excitation. Acceleration was introduced by accelerating the simulated powertrain instances with a constant acceleration to different rotating speeds. The rotating speeds of the simulated instances started at 500 rpm, rising in 500 rpm increments, until reaching 3500 rpm. Ice excitation was introduced according to propeller loads typical in ice conditions [33]. Ice excitations were introduced in each rpm range. Random excitation was generated with a pseudorandom binary sequence, and added for each rpm range introduced. Examples of how excitation was visible

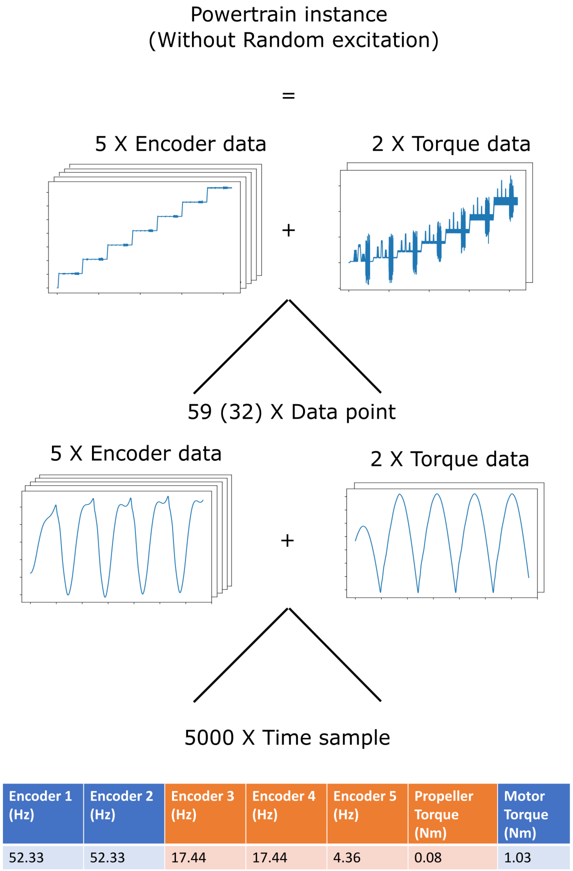
Table 1

Nominal powertrain parameters used in the simulated powertrain instances. Parameters belonging to the mass moment of inertia (I_i), damping (c_i) and stiffness (k_i) for each powertrain instance were drawn from a uniform distribution between 0.5 and 1.5 times each nominal value separately

Component	Index i	I_i (kgm^2)	c_i (Nm/(rad/s))	k_i (Nm/rad)	d_i (Nm/(rad/s)
Driving motor, coupling	1	7.94×10^{-4}	8.08	1.90×10^5	0.0030
Shaft	2	3.79×10^{-6}	0.29	6.95×10^3	0
Elastomer coupling hub	3	3.00×10^{-6}	0.24	90.0	0
Elastomer coupling middle piece	4	2.00×10^{-6}	0.24	90.0	0
Elastomer coupling hubs & shaft	5	7.81×10^{-3}	0.24	90.0	0
Elastomer coupling middle piece	6	2.00×10^{-6}	0.24	90.0	0
Elastomer, coupling hub, encoder & shaft	7	3.17×10^{-6}	0.00	30.13	0
Shaft, encoder & coupling	8	5.01×10^{-5}	1.78	4.19×10^4	0
Torque transducer	9	6.50×10^{-6}	0.23	5.40×10^3	0
Torque transducer & coupling	10	5.65×10^{-5}	1.78	4.19×10^4	0
Shaft	11	4.27×10^{-6}	0.52	1.22×10^3	0
Shaft & gear	12	3.25×10^{-4}	1.84	4.33×10^4	0.0031
Coupling	13	1.20×10^{-4}	1.32	3.10×10^4	0
Shaft	14	1.15×10^{-5}	0.05	1.14×10^3	0
Shaft & coupling	15	1.32×10^{-4}	1.32	3.10×10^4	0
Shaft	16	4.27×10^{-6}	0.52	1.22×10^4	0
Shaft & gear	17	2.69×10^{-4}	1.88	4.43×10^4	0.0031
Coupling	18	1.80×10^{-4}	5.86	1.38×10^5	0
Torque transducer	19	2.00×10^{-5}	0.85	2.00×10^4	0
Torque transducer & coupling	20	2.00×10^{-4}	5.86	1.38×10^5	0
Shaft	21	4.27×10^{-6}	0.52	1.22×10^4	0
Shaft, mass & loading motor	22	4.95×10^{-2}	–	–	0.2400

in the generated data can be seen in Fig. 5. Training and testing data consisted only of parts of the data that were acquired while the system experienced one of the aforementioned excitations: time spent running the system at constant rotating speeds was cut out.

From each randomized powertrain instance, 210 seconds of continuously generated measurement data was acquired at a sampling rate of 3 kHz. This cycle was similar to the predetermined test cycle for the real test bench, allowing for future use of this data in sim-to-real scenarios. Each time sample consisted of generated measurement for five rotary encoders measuring angular speeds and two torque transducers measuring torque at the given time sample from the propeller and the motor. The data from two simulated rotary encoders measuring angular speed near the motor, and a simulated torque transducer measuring the motor torque were used as input, in other words as features for the virtual sensor. The virtual sensors goal was to estimate three angular speeds measured closer to the propeller and the propeller torque. Three simulated rotary encoders and one simulated torque transducer were used as the target output for the virtual sensor being trained as seen in Fig. 3. Generated measurements were cut into 5000 time sample sized data points, or approximately 1.67 seconds of generated measurement. Each time sample included the aforementioned seven generated measurement data samples. The dataset has been visualized in Fig. 4. To increase the size of the training set, a stride of 250 measured samples was used for the powertrain instances used in training. This increased the amount of data points obtained from a single powertrain instance from 59 to 1008 if random excitation was present and from 32 to 552 if random excitation was not present.



Encoder 1 (Hz)	Encoder 2 (Hz)	Encoder 3 (Hz)	Encoder 4 (Hz)	Encoder 5 (Hz)	Propeller Torque (Nm)	Motor Torque (Nm)
52.33	52.33	17.44	17.44	4.36	0.08	1.03

Fig. 4. Visualized data acquisition and the form of data. A single powertrain instance consists of 210 seconds of continuously generated data. A data point consists of 1.67 seconds of continuously generated data, or 5000 measured time samples, which each include a single measurement from five simulated encoders and two torque transducers. The amount of non-overlapping data points depends on whether random excitation parts of the simulated data are included in the instance (59 if yes, 32 if not). Blue values are given as input and orange as output

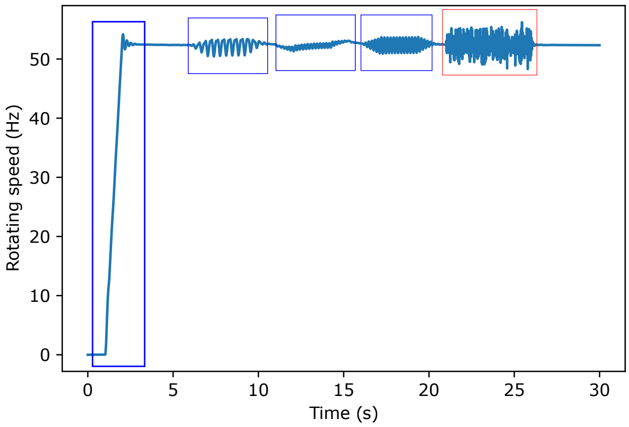


Fig. 5. Parts of generated data, where the simulation models an excitation. Highlighted blue areas correspond to realistic excitations: acceleration and ice excitations. Excitation highlighted in red corresponds to random excitation only included in parts of training and testing

3.2. Training algorithm and presented virtual sensor architecture

A virtual sensor architecture consisting of four layers of LSTM followed by a fully connected layer was employed. The suggested virtual sensor takes generated data points as input. Model parameters are listed in Table 2.

The training procedure for the data-driven virtual sensor was conducted with supervised learning. During training, rotating speeds and torque near the propeller end of the powertrain were estimated from rotating speeds and torque near the motor end of the powertrain, and an error term was established using a loss function. The loss was calculated using mean squared error (MSE) defined in equation (7), where n corresponds to the number of samples, Y_i to the target value to be estimated and $\hat{Y}_i$ to the estimated value for each estimation i. The loss terms shown in Results have been expressed in MSE. The loss is applied via Stochastic Gradient Decent (SGD) [34] to improve the weights of the model after every training epoch. More precisely, the model was optimized using Adam [35], which is a widely used optimizer for neural networks trained with SGD. Specific training parameters are shown in Table 2.

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2, \tag{7}$$

In addition, transfer learning technique domain randomization in the form of parameter randomization and random exci-

Table 2

Model and training parameters used in training and testing. Window length corresponds to the time sample size, stride corresponds to the amount of samples between the start of a data point and the start of the next data point, dimensions correspond to the used array dimensions as input and output of the virtual sensor, batch size to the amount of data points given to the model at a time, layers to the amount of LSTM layers in the model, learning rate to a parameter used in the back propagation algorithm and epochs to the amount of iterations of training through the training set

Parameter	Value
Window length	5000
Training stride	250
Test stride	5000
Model input dimension	5000 × 3
Model output dimension	5000 × 4
Batch size	2
Cell size	81
Number of layers	4
Dropout	0
Learning rate	0.0001
Epochs	15

tation were introduced to the training set as described in Data acquisition. Since domain randomization can be applied in principle to any property, applying it directly to physical quantities of the powertrain seemed promising. In practice, parameter randomization implies that the training set should contain data from multiple powertrain instances with differences in physical parameters. Random excitation hopes to broaden the feature space of the data by introducing quantities that are not inspired by realistic excitation that the system would experience in normal use. Otherwise, the domains remained the same: the sensor placements and types remained identical, the sampling frequency was constant and so on as stated in Data gathering.

3.3. Testing procedure

The test set included data from 30 separate powertrain instances. For further validation, two test sets were created: one without random excitations mentioned above, and one with random excitation present, labelled respectively as Test set 1 and Test set 2. While splitting the test sets into 5000 time sample sized data points, a stride of 5000 time samples was used to ensure that the test samples did not possess identical information, which could have affected the results. Test data only included data points where the system experienced an excitation, similarly to the training data.

To analyse the performance, a baseline model was created. The baseline model was trained using a singular powertrain without random excitation. This was done to give a baseline for the two domain randomization techniques (parameters and excitations) and to showcase the differences in results more clearly.

To even further demonstrate the influence of each transfer learning technique, three training scenarios were conducted. The first training scenario focused on finding out if the random excitation included in the simulated powertrain data would help the virtual sensor generalize to many powertrains with slight variations. A virtual sensor was trained using the same powertrain as the baseline was trained with but with the random excitation parts of the acquired data present. Notably, this heavily increased the amount of training data for the virtual sensor compared to the baseline.

The second training scenario focused on slight differences in the powertrain configurations. A virtual sensor was trained using 30 separate powertrain instances generated with the simulated model, not including the powertrain involved in the training of the baseline virtual sensor. Randomized excitation was not present in the training set.

Lastly, the third training scenario combined both suggested generalization methods from previous cases. The virtual sensor was trained with the same 30 powertrain instances that were used to train the virtual sensor of the second case study, but with random excitation present similarly to the first case study. This once again significantly increased the amount of training data in comparison to the previous scenario. For each presented virtual sensor, tests with both test sets were conducted and an MSE score was calculated with the combined MSE of the output, and for each output sensor separately.

4. RESULTS AND DISCUSSION

Results for each training scenario and the baseline virtual sensor have been shown in Tables 3 and 4 and visualized in Fig. 6. Based on the results, parameter randomization seems to be an excellent way to train data-driven virtual sensors that generalize for many similar but slightly different powertrains, resulting in successful transfer learning between these slight differences. It seems that the use of multiple parameter randomized powertrains allowed the neural network to generalize to new parameter randomized powertrains. Furthermore, adding randomized excitation seems to have a similar improving effect, though smaller. With these methods combined, a significantly better generalized virtual sensor for powertrains was obtained.

The baseline virtual sensor had really poor MSE scores due to inability to see torque levels beyond the singular powertrain

Table 3

Testing scenario results for Test set 1. Scores are represented in MSE. Titles match titles used in Fig. 6. Enc 3–5 correspond to the encoder 3–5 outputs described in Fig. 4. Combined MSE is the combined MSE value of all output values

Model	Enc3	Enc4	Enc5	Torque	Combined
Baseline	1813.53	1835.86	35.72	126.01	952.78
Random exc	791.79	829.17	19.83	93.11	433.48
Random par	0.33	0.30	0.02	0.17	0.21
Combined	0.28	0.28	0.02	0.05	0.16

Table 4

Testing scenario results for Test set 2. Scores are represented in MSE. Titles match titles used in Fig. 6. Enc 3–5 correspond to the encoder 3–5 outputs described in Fig. 4. Combined MSE is the combined MSE value of all output values

Model	Enc3	Enc4	Enc5	Torque	Combined
Baseline	2207.46	2233.65	40.66	161.98	1160.94
Random exc	989.95	1035.23	22.68	121.56	542.36
Random par	0.61	0.68	0.07	3.27	1.16
Combined	0.25	0.25	0.02	0.15	0.17

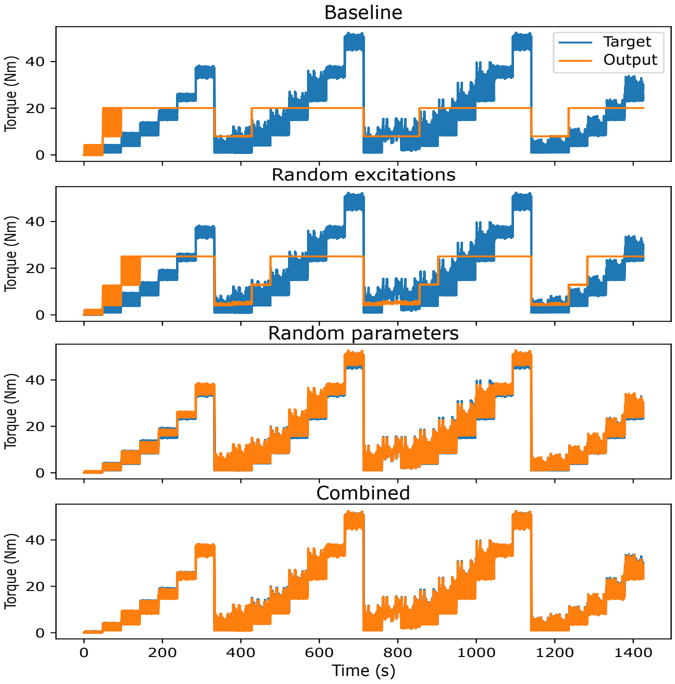


Fig. 6. A collection of differences between all the training scenario virtual sensors when estimating the propeller torque from powertrain instances in Test set 1. The output of the virtual sensor is in orange and target data is in blue. The virtual sensor for the first training scenario is titled “Random excitations”, the second is titled “Random parameters” and the third is titled “Combined”. The baseline virtual sensor is titled “Baseline”

levels. By introducing random excitation to the training data, the combined MSE score went down over 40% for both test sets. This could be due to reduced overfitting of the model to the data in a singular powertrain instance, since the new excitation type increases the amount of data. By introducing 30 different powertrain instances (Random parameters), the training set was multiplied by 30, further lowering the chance of overfitting. Combining these transfer learning techniques amounted with the lowest combined MSE score for both test sets. Interestingly, including randomized excitation in the training set where randomized parameters were already present still increased the accuracy notably. Adding data from random excitation and random parameters to the training data seems to be a significant way of helping the virtual sensor generalize more to physical differences of the system. Since the variation introduced in this study was severe (0.5–1.5 times the nominal values), it is sufficient to assume that slighter variations, for example stemming from manufacturing inaccuracies or differences in installations and operation environments, would produce good results as well.

4.1. Future work

The results obtained from this study suggest that the random parameter and random excitation approach is valid on training more general data-driven virtual sensors for powertrains. That is, by adding randomization into the training data obtained from a simulated powertrain, the virtual sensor could generalize to other simulated powertrains with different physical parameters. Next steps include combining the method used in this study with other techniques, such as an improved data generation model, introducing domain randomization to other properties, and fine-tuning the model with real data during training, to implement a simulation trained virtual sensor that could function on real measured data from the physical test bench. In the future, this could potentially reduce the amount of real measurement data required to train data-driven virtual sensors.

5. CONCLUSIONS

In the present study, a training procedure for a data-driven LSTM based virtual sensor for a single powertrain with realistic variations was introduced. The trained virtual sensor was proven to work accurately. In addition, the effect of two domain randomization methods, parameter randomization and adding randomization, to the excitation were tested.

The results stated that a virtual sensor with the proposed architecture, trained with the combined training procedure, accomplished accurate results at estimating torques and rotating speeds from the drive end of the powertrain. The model accomplished fair generalization, despite major variations between different powertrain configurations.

The results of this study show that the random parameter and random excitation approach is valid on training more general virtual sensors for powertrains. However, this study demonstrated the functionality in a simulation-to-simulation approach - the test set was naturally different from the training data but still simulated. The next steps obviously include the replacement of the test set with a measurement from a physical test bench.

REFERENCES

[1] J. Bai, X. Wu, F. Gao, and H. Li, “Analysis of powertrain loading dynamic characteristics and the effects on fatigue damage,” *Appl. Sci.*, vol. 7, no. 10, p. 1027, 2017.

[2] Q. Sun and Z. Ge, “A survey on deep learning for data-driven soft sensors,” *IEEE Trans. Ind. Inform.*, vol. 17, no. 9, pp. 5853–5866, 2021, doi: 10.1109/TII.2021.3053128.

[3] P. Kadlec, B. Gabrys, and S. Strandt, “Data-driven soft sensors in the process industry,” *Comput. Chem. Eng.*, vol. 33, no. 4, pp. 795–814, 2009, doi: 10.1016/j.compchemeng.2008.12.012.

[4] S. Gillijns and B. De Moor, “Unbiased minimum-variance input and state estimation for linear discrete-time systems,” *Automatica*, vol. 43, no. 1, pp. 111–116, 2007.

[5] K. Tiwari, A. Shaik, and A. N, “Tool wear prediction in end milling of ti-6al-4v through kalman filter based fusion of texture features and cutting forces,” *Procedia Manuf.*, vol. 26, pp. 1459–1470, 2018, 46th SME North American Manufacturing Research Conference, NAMRC 46, Texas, USA.

[6] J. Škach and I. Punčochář, “Input design for fault detection using extended kalman filter and reinforcement learning,” *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 7302–7307, 2017, 20th IFAC World Congress.

[7] M. Manngård *et al.*, “Torque estimation in marine propulsion systems,” Jun. 2022.

[8] U. Lagerblad, H. Wentzel, and A. Kulachenko, “Study of a fixed-lag kalman smoother for input and state estimation in vibrating structures,” *Inverse Probl. Sci. Eng.*, vol. 29, no. 9, pp. 1260–1281, 2021.

[9] R. De Waal, A. Bekker, and P.S. Heyns, “Indirect load case estimation for propeller-ice moments from shaft line torque measurements,” *Cold Reg. Sci. Tech.*, vol. 151, pp. 237–248, 2018.

[10] T. Ikonen, O. Peltokorpi, and J. Karhunen, “Inverse ice-induced moment determination on the propeller of an ice-going vessel,” *Cold Reg. Sci. Tech.*, vol. 112, pp. 1–13, 2015.

[11] J.C.B. Gonzaga, L.A.C. Meleiro, C. Kiang, and R. Maciel Filho, “Ann-based soft-sensor for real-time process monitoring and control of an industrial polymerization process,” *Comput. Chem. Eng.*, vol. 33, no. 1, pp. 43–49, 2009, doi: 10.1016/j.compchemeng.2008.05.019.

[12] W. Yan, H. Shao, and X. Wang, “Soft sensing modeling based on support vector machine and bayesian model selection,” *Comput. Chem. Eng.*, vol. 28, no. 8, pp. 1489–1498, 2004, doi: 10.1016/j.compchemeng.2003.11.004.

[13] S. Sharma, M. Diwakar, P. Singh, A. Tripathi, C. Arya, and S. Singh, “A review of neural machine translation based on deep learning techniques,” in *2021 IEEE 8th Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON)*, 2021, pp. 1–5, doi: 10.1109/UPCON52273.2021.9667560.

[14] J. Su, B. Xu, and H. Yin, “A survey of deep learning approaches to image restoration,” *Neurocomputing*, vol. 487, pp. 46–65, 2022, doi: 10.1016/j.neucom.2022.02.046.

[15] S. Tang, S. Yuan, and Y. Zhu, “Deep learning-based intelligent fault diagnosis methods toward rotating machinery,” *IEEE Access*, vol. 8, pp. 9335–9346, 2020, doi: 10.1109/ACCESS.2019.2963092.

[16] C. Shang, F. Yang, D. Huang, and W. Lyu, “Data-driven soft sensor development based on deep learning technique,” *J. Process Control*, vol. 24, no. 3, pp. 223–233, 2014, doi: 10.1016/j.jprocont.2014.01.012.

[17] R. Zhao, R. Yan, Z. Chen, K. Mao, P. Wang, and R. X. Gao, “Deep learning and its applications to machine health monitoring,” *Mech. Syst. Signal Proc.*, vol. 115, pp. 213–237, 2019, doi: 10.1016/j.ymssp.2018.05.050.

[18] J. Miettinen, T. Tiainen, R. Viitala, K. Hiekkanen, and R. Viitala, “Bidirectional lstm-based soft sensor for rotor displacement trajectory estimation,” *IEEE Access*, vol. 9, pp. 167 556–167 569, 2021, doi: 10.1109/ACCESS.2021.3136155.

[19] S. Vijaya Raghavan, T. Radhakrishnan, and K. Srinivasan, “Soft sensor based composition estimation and controller design for an ideal reactive distillation column,” *ISA Trans.*, vol. 50, no. 1, pp. 61–70, 2011, doi: 10.1016/j.isatra.2010.09.001.

[20] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Comput.*, vol. 9, pp. 1735–1780, 12 1997, doi: 10.1162/neco.1997.9.8.1735.

[21] Q. Sun and Z. Ge, “Probabilistic sequential network for deep learning of complex process data and soft sensor application,” *IEEE Trans. Ind. Inform.*, vol. 15, no. 5, pp. 2700–2709, 2019, doi: 10.1109/TII.2018.2869899.

[22] F. Galati, W. Wang, B. Bielenberg *et al.*, “Gear-bearing fault detection based on deep learning,” in *AIAC18: 18th Australian International Aerospace Congress (2019): HUMS-11th Defence Science and Technology (DST) International Conference on Health and Usage Monitoring (HUMS 2019): ISSFD-27th International Symposium on Space Flight Dynamics (ISSFD)*. Engineers Australia, Royal Aeronautical Society., 2019, p. 916.

[23] C.A. Duchanoy, M.A. Moreno-Armendáriz, L. Urbina, C.A. Cruz-Villar, H. Calvo, and J. de J.Rubio, “A novel recurrent neural network soft sensor via a differential evolution training algorithm for the tire contact patch,” *Neurocomputing*, vol. 235, pp. 71–82, 2017, doi: 10.1016/j.neucom.2016.12.060.

[24] X. Chen, F. Gao, and G. Chen, “A soft-sensor development for melt-flow-length measurement during injection mold filling,” *Mater. Sci. Eng. A*, vol. 384, no. 1, pp. 245–254, 2004, doi: 10.1016/j.msea.2004.06.039.

[25] T.T. Nguyen, N. D. Nguyen, and S. Nahavandi, “Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications,” *IEEE Trans. Cybern.*, vol. 50, no. 9, pp. 3826–3839, 2020, doi: 10.1109/TCYB.2020.2977374.

[26] W. Zhao, J.P. Queralta, and T. Westerlund, “Sim-to-real transfer in deep reinforcement learning for robotics: a survey,” in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2020, pp. 737–744, doi: 10.1109/SSCI47803.2020.9308468.

[27] F. Muratore, C. Eilers, M. Gienger, and J. Peters, “Data-efficient domain randomization with bayesian optimization,” *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 911–918, 2021.

[28] F. Muratore, F. Ramos, G. Turk, W. Yu, M. Gienger, and J. Peters, “Robot learning from randomized simulations: A review,” *Front. Robot. AI*, p. 31, 2022.

[29] O.M. Andrychowicz *et al.*, “Learning dexterous in-hand manipulation,” *Int. J. Robot. Res.*, vol. 39, no. 1, pp. 3–20, 2020, doi: 10.1177/0278364919887447.

[30] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, “Domain randomization for transferring deep neural networks from simulation to the real world,” in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 23–30, doi: 10.1109/IROS.2017.8202133.

[31] M. Manngård *et al.*, “Torque estimation in marine propulsion systems,” *Mech. Syst. Signal Proc.*, vol. 172, p. 108969, 2022.

[32] S. Haikonen, I. Koene, J. Keski-Rahkonen, and R. Viitala, “Small-scale test bench of maritime thruster for digital twin research,” in *2022 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, 2022, pp. 1–6, doi: 10.1109/I2MTC48687.2022.9806563.

[33] T. TSA, “Maritime safety regulation. ice class regulations and the application thereof,” 2010.

[34] S. Ruder, “An overview of gradient descent optimization algorithms,” *CoRR*, vol. abs/1609.04747, 2016. [Online]. Available: http://arxiv.org/abs/1609.04747

[35] D.P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.