# Machine Learning for Digital Twins to Predict Responsiveness of Cyber-Physical Energy Systems

Snijders, Ron; Pileggi, Paolo; Broekhuijsen, Jeroen; Verriet, Jacques; Wiering, Marco; Kok, Koen

*Published in:*
2020 8th Workshop on Modeling and Simulation of Cyber-Physical Energy Systems

*DOI:*
10.1109/mscpes49613.2020.9133695

# Machine Learning for Digital Twins to Predict Responsiveness of Cyber-Physical Energy Systems

Ron Snijders*, Paolo Pileggi†, Jeroen Broekhuijsen‡, Jacques Verriet§, Marco Wiering¶ and Koen Kok‖
*Dept. Monitoring & Control Services, TNO, Groningen, ron.snijders@tno.nl
†Dept. Monitoring & Control Services, TNO, The Hague, paolo.pileggi@tno.nl
‡Dept. Monitoring & Control Services, TNO, Groningen, jeroen.broekhuijsen@tno.nl
§ESI (TNO), Eindhoven, jacques.verriet@tno.nl
¶Dept. Artificial Intelligence and Cognitive Engineering, University of Groningen, Groningen, m.a.wiering@rug.nl
‖Dept. Electrical Energy Systems, TU Eindhoven, Eindhoven &
Dept. Monitoring & Control Services, TNO, Groningen, koen.kok@tno.nl

***Abstract*—Cyber-Physical Systems are becoming more autonomous, interconnected, complex and adaptive, and are expected to operate in highly dynamic environments. This is especially challenging for energy ecosystems that are increasingly difficult to control and maintain as the number of participating manufacturers and users grows. Digital Twins help analyze and predict these systems in the form of digital reflections that operate in parallel with the physical system. In this paper, we use Machine Learning to improve the predictive power of Digital Twins for Cyber-Physical Energy Systems. Specifically, we use a Temporal Convolutional Neural Network model to learn the temporal patterns in the system and predict its responsiveness to specific power setpoint instructions. Real-life data from ten batteries were used to predict the behavior over time. Compared to the baseline model that uses the prior probability of response and the average response rate within the configured time window, the model predicts the batteries' responsiveness more accurately. The more temporal information is used as input for prediction, the better the model performs in both precision and recall. The results show that this compensates for the lack of information when fewer metrics are used. The use of Machine Learning for Digital Twins can help maintain a heterogeneous energy ecosystem, while minimizing the need to acquire or disclose detailed information.**

***Index Terms*—Machine Learning, Digital Twin, Cyber-Physical Energy System, Temporal Convolution Neural Network**

## I. Introduction

Digitalization and its pervasiveness have brought new opportunities to monitor and control complex physical systems effectively. Industrial IoT, Industry4.0, Cloud Native, Cloud Computing, and Big Data paradigms are among the key drivers of the effective Digital Twin for industrial Cyber-Physical Systems of Systems (CPSoS).

In the energy domain, sensor and context data from a Cyber-Physical Energy System (CPES) can be used to synchronize fit-for-purpose models with reasoning logic during the operational phase of the system, i.e., Digital Twins coordinated to support specific system goals. Previously, Pileggi et al. used an actual flexible energy system deployment to demonstrate the applicability of the Digital Twin [1].

The energy system pilot [2] was deployed in Heerhugowaard, a municipality in the Netherlands. Ninety households participated over a period of two years. Household controllers and a central controller managed energy production and consumption using PowerMatcher [3], [4], which balances smart devices in a virtual market in an auctioneering process.

Sensor data were collected from various energy system assets, like solar panels and heat pumps. In their investigation, Pileggi et al. mainly focus on the battery type of assets, as shown in Fig. 1. These batteries provide flexibility to compensate for the energy imbalance in the ecosystem. They demonstrated an application of Digital Twin simulation to explore system battery behavior using an expert model [1]. The battery receives instructions from the controller and provides the controller with its flexibility information. Data about the battery is provided to the Digital Twin, which provides anomaly insights about the unknown battery behavior. In this process, Pileggi et al. observed that expert knowledge and insight can be incomplete, inconsistent and incorrect.

In this paper, we extend their research by investigating the use of Machine Learning for Digital Twins to predict a battery's responsiveness. We set out to predict whether the battery will respond to a desired negative power setpoint, i.e., whether it will produce energy as requested by the setpoint instruction received. This is essential to balance the power usage within a CPES. There are several reasons why a battery does not respond to a desired negative setpoint. For example, the state-of-charge may be too low, or the battery may be operating in an emergency state to avoid damage. The exact underlying root causes are often poorly understood or obfuscated by battery firmware.

State-of-the-art Machine Learning methods, like Deep Learning [5], are able to efficiently approximate a large state space in an implicit abstract way. They outperform expert models in that they do not have to evaluate all possible states, resulting in higher performance [6]. Specifically, our method applies a Temporal Convolutional Neural Networks (TCN) [7],

978-1-7281-8721-1/20/$31.00 ©2020 IEEE

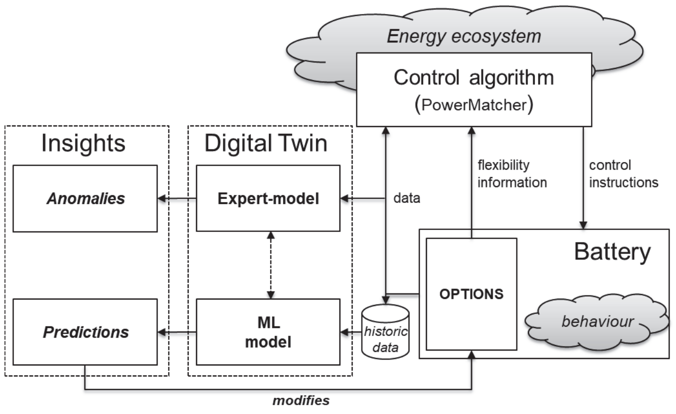

Fig. 1. Digital Twin using an expert model to investigate battery behavior as used in [1] extended with the Machine Learning model. Ideally, the battery provides current flexibility information, e.g., the amount of power it may consume or produce, which is used by the central energy controller to balance the energy ecosystem. Using Machine Learning, the responsiveness of the CPES becomes more predictive and allows the modification of the flexibility information provided to the energy control algorithm.

a form of deep learning, to learn temporal patterns in the system. These patterns are used to predict whether a battery will produce energy when it receives a negative power setpoint instruction.

In the sequel, we discuss state-of-the-art Machine Learning methods. In Sect. III, we describe the Digital Twin in the context of CPSoS and outline related work. Our method is presented in Sect. IV and, in Sect. V, we present our experiments and results. We conclude with future work in Sect. VI.

## II. State of the Art

The method we use to predict whether a battery in a CPES will respond to a negative power setpoint has similarities with condition and system health monitoring. Condition monitoring observes a system for certain conditions for early fault detection in order to avoid catastrophic failures. Hameed et al. [8] consider condition monitoring of wind turbines, they present a review of available techniques for condition monitoring and fault detection.

Similarly, system health monitoring assesses whether a system is capable of performing its intended task. Borth and Barbini [9] present a method based on Bayesian networks to assess whether a ship is ready for its mission. They try to determine whether a system needs maintenance. Also, Valant et al. [10] use 1D CNN autoencoders (a form of deep learning) trained on synthetic data to assess the condition of batteries.

Despite the similarities, there are also differences. In our research we use real-life data and our method does not focus on system maintenance, but on system operation: it predicts the readiness of a system for a single action instead of a long mission. Because of the shorter scope, the context of our methods is more dynamic and transient than that of system health monitoring. In other words and more specifically, a currently unresponsive battery may become responsive later.

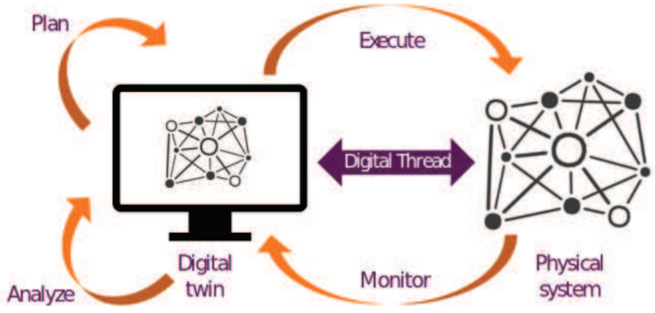

Fig. 2. Digital Twin in the physical system with the MAPE-K model.

The combination of expert and black-box models, where they complement or update each other, with the aim to improve explainability and to predict performance, is a sought-after solution. Existing research already offers some solutions. For example, Disentangled Variational Autoencoders, another form of Deep Learning, have the ability to encode abstract representations into discrete symbols (latent variables) in the bottleneck [11]. These symbols could, for example, represent color and size, in case of object recognition, but also an abstract combination of global states in case of running processes, i.e., a charging battery, operating in an emergency state, etc.

Process Mining techniques seem an appropriate passive learning technique [12], [13], but do not easily address the state-space problem that many algorithms face in the way deep learning implicitly does. Instead, process discovery from logs and conformance checks need specialized algorithms.

## III. Digital Twin in the CPSoS Context

The *Digital Twin*, despite there being no clear consensus on a definition in industry or academia [14], is a fit-for-purpose model with reasoning logic synchronized with the operating physical system to help the system be effective. In our example case, the designated system is a CPES.

CPSoSs are complex and there is a never-ending drive towards automating adaptivity of control processes, given the dynamic nature of the operating environment. Dynamicity is characterized by both frequency and degree to which the system environment changes. Our vision of the Digital Twin in this context aligns with the MAPE-K reference model for autonomous systems [15], as shown in Fig. 2.

The interactions shown in the figure represent an operating monitoring and control cycle consisting of the following actions:

**Monitor:** The twin continuously observes the system, obtaining sensor and context data by what is called the *Digital Thread*. The Digital Thread represents the sensor data and context information, as well as the control and flow information communicated between the physical system and its twin.

**Analyze and Plan:** Reasoning about how the system should be configured or which action(s) it should take to reach the specific goal.

**Execute:** Change the system by interacting with the controller of the system, which has a co-ordination mechanism to consider trade-offs between the objectives of different optimization processes in the system, that, in turn, passes control to some actuation mechanism.

The use and availability of shared Knowledge is supported by the Digital Thread. The Twin may receive context information about the system and its environment, and it is not excluded from generating insights about the system that can be shared with the environment as well. This makes it possible to foster a co-operative collaborative ecosystem.

## IV. Response Prediction Methods

The aim is to identify what can be learned and predicted solely based on the data collected from the CPES. Hence, we do not model the CPES in precise detail. During the development of our method, we were deliberately uninformed about the inner workings of the battery.

### *A. Data Input and Pre-processing*

We collected the following metric data from the battery and used them as input for the model:

**Power setpoint:** Amount of power requested by the energy control algorithm from the battery indicated in the instruction of the controller. A positive setpoint indicates that the battery should consume energy, i.e., charge, while a negative setpoint indicates that the battery should produce energy, i.e., discharge.

**Power:** Actual amount of power produced or consumed by the battery. A positive or negative difference between the power setpoint and power indicates that the battery does not respond to the power setpoint request.

**Current:** Actual current of the battery.

**Voltage:** Actual voltage of the battery.

**State-of-charge:** Reported state-of-charge as calculated by the internal firmware of the battery.

The power setpoint, power, and state-of-charge are regarded as the most important metrics, since the current and voltage metrics directly correlate with the power and state-of-charge. Additional information, such as the internal reported state, mode and control information of the battery, were deliberately ignored for the sake of generality of the method.

The original metric data are resampled from an individual measurement every 16 seconds to an aggregated mean of one measurement every hour. This reduces noise and the total size of the data, allowing for faster learning by the model. Refer to Sect. V-C for more detail. All negative metric values are normalized to the range $[-1, 0]$, and all positive metric values are normalized to the range $[0, 1]$.

### *B. Temporal Convolutional Network*

The method used in this paper is based on the TCN [7]. It is a Deep Artificial Neural Network, also known as Deep Learning [5], with causal dilated convolutions [16]. The output of time $t$ is convolved, and only depends on the information from $t$ and earlier in the previous layers. The TCN uses

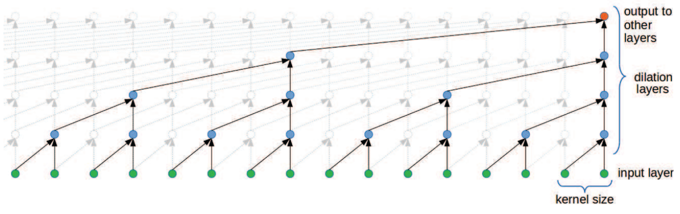

Fig. 3. Example structure of a TCN shown for a single input metric (similar as shown in [18]). Only 4 dilation layers are shown here (with dilation 1, 2, 4 and 8), the actual TCN model used in our research uses up to 7 dilation layers (with dilation 1, 2, 4, 8, 16, 32 and 64). An extra dense layer of 20 neurons is added to the end of the TCN which is connected to the final output neuron to provide the classification result. The size of the input layer depends on the window size (16 in the figure above) and the number of metrics being used (1 in the figure above).

residual connections [17] to ease training for deep neural networks with many layers. A detailed example of the network is shown in Fig. 3. In our version the network is extended with an extra fully connected layer of 20 neurons which feeds the final results to the output neuron.

Each input neuron represents the hourly mean of a single metric, e.g., power and state-of-charge, in the interval $[(t-w), t]$, where $w$ is the configurable window size in hours. The Rectified Linear Unit, known for its fast learning in networks with many layers [19], is used as the activation function in each neuron except for the output neuron which has a sigmoid activation function. Binary cross-entropy is used as the loss function. To train the network, the Adam algorithm [20] is used, which is an extension to stochastic gradient descent. We used the Python Deep Learning library *Keras* [21] and an existing TCN implementation[1] as the basis of our implementation.

We have considered using Recurrent Neural Networks (RNN) such as those based on Long Short-Term Memory (LSTM) [22] for our application. However, TCNs have shown to outperform RNNs in many tasks [7] while its design allows better parallelism. Although we do admit that a proper validation on the same dataset is required to justify this claim.

### *C. Response Prediction Model*

A single power request event used for the response prediction model is defined as the moment the power setpoint goes from 0 to a negative value (see Fig. 4). The aim of the model is to predict the expected behavior given historical data with window size $w$ in hours.

The model is a binary classifier and predicts whether the battery is likely (1) or unlikely (0) to respond to the negative power setpoint. For this purpose, an extra dense layer of 20 neurons is added to the output of the TCN in addition to a final layer of one neuron to form the final classification output, which is then rounded to a fixed classification of either 0 or 1. Fig. 3 shows an example of the structure of the network.

Only historical metric data with window size $w$ are given to the model. A *single* model is trained using data from multiple

[1]https://github.com/philipperemy/keras-tcn

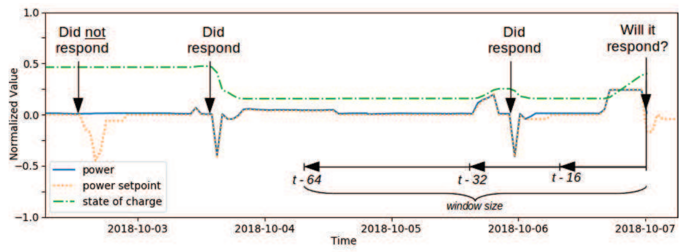

Fig. 4. Example of different negative power setpoint requests in which the battery should respond by producing power. Sometimes the battery responds and sometimes it does not despite having sufficient state-of-charge. Other metrics are omitted for sake of clarity. The key challenge of the learned model is to predict whether or not the battery will respond at time $t$ given the input in the window $t - w$.

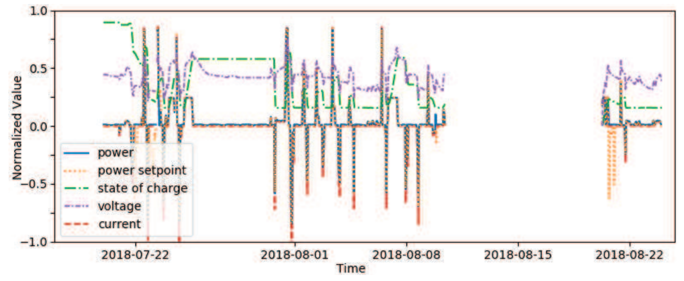

Fig. 5. A Snapshot of raw data used in our experiments for a single battery period resampled to hourly data. The data is imperfect due to missing data points and continuously changing usage patterns of the battery.

batteries collected over a period of several months. However, no battery-specific information is provided, making the model suitable for similar but unseen batteries.

## V. Experiments

We consider a period of about 8 months where data were collected for 10 different batteries. From each battery, its power level, current, voltage, state-of-charge, mode, state and power setpoint were collected. However, in many cases, data were missing for long periods at a time, resulting in incomplete data over time. Fig. 5 shows one month of raw data collected from a single battery. Noisy data and small interruptions were compensated by resampling the data to hourly intervals. Refer to Sect. V-C for an in depth discussion about resampling effects. Incomplete periods due to interruptions larger than one hour were ignored.

A total of 972 power request events were collected to train, evaluate and test the Response Prediction Model. A single power request event was defined at the moment the power setpoint went from 0 to a negative value. The duration and amplitude of the negative power setpoint were not considered. The first 85% events from each battery were used for training and evaluation during model development. The reported performance in this article is tested on the remaining 15% of all events, constituting the holdout set. The model was not optimized thereafter. The events were ordered in time such that it would reflect a realistic situation as to how the model would be used in practice, i.e., we trained using historical data and tested on new unseen data.

To evaluate the performance of the response prediction model, the classification accuracy, precision, recall and F1 score are calculated for the test set, i.e., the remaining 15% of all power request events. In addition, we calculate the Matthews Correlation Coefficient (MCC) as our primary performance metric. The MCC metric takes both the true positives and true negatives into account. It has been shown to be robust against unbalanced classes [23]. The latter is important since many batteries do not respond in most cases. Besides the accuracy and MCC score of the model, we are also interested in precision and recall. We want to identify batteries that are most likely to respond to the setpoint instruction.

### A. Baseline Performance of the Setpoint Response

The baseline method illustrates the naive way of approaching the problem without using machine learning and is used as a performance reference for the evaluation of the TCN model. It is based on the intuition that, if a battery did not respond recently in the past, it is less likely do so in the future. Its predicted response $r(b, w)$ for battery number $b$ and window size $w$ (in hours, $w \in \{1, 4, 8, 16, 32, 64\}$) is formulated by Equation 1.

$$r(b,w) = \begin{cases} \begin{cases} 1, & \text{if } E[R|b] \geq 0.5 \\ 0, & \text{otherwise} \end{cases}, & \text{if } N = 0 \\ \begin{cases} 1, & \text{if } E[R|b,w] \geq 0.5 \\ 0, & \text{otherwise} \end{cases}, & \text{otherwise} \end{cases} \tag{1}$$

$E[R|b]$ is the conditional expected response, i.e., the arithmetic mean, with respect to battery number $b$, and $E[R|b, w]$, with respect to both $b$ and window size $w$. The baseline method uses the expected response $E[R|b]$ only when there has been no power setpoint requests in the past time window $w$ ($N = 0$). Otherwise, if the number of power setpoint requests $N$ within the window is greater than 0, it uses the expected response $E[R|b, w]$ *within* that window. If $w = 1$, no past information is used, in which case $N = 0$.

### B. Response Prediction Results

The classification performance, i.e., MCC, for different trained response prediction models is shown in Fig. 6. A more detailed overview of all used performance metrics is shown in Fig. 7 for the best performing variant of each method. Six different window sizes, namely, 1, 4, 8, 16, 32 and 64, were evaluated. The use of two different sets of input metrics, namely

- only power and power setpoint (TCN-pwr), and
- power, current, state-of-charge, voltage, and power setpoint (TCN-all),

was evaluated. This resulted in a total of 12 different model configurations that were trained. In all cases, the model was trained for 300 epochs on the training set, i.e., the first 85% of the total dataset. During each epoch, a random set of 20%

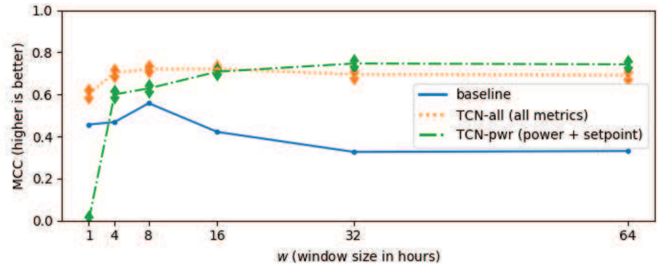

Fig. 6. Performance results showing the Matthews Correlation Coefficient (MCC) for the baseline method and two different versions of the trained TCN model. The results shown are the mean scores over 100 different training runs. The error bars indicate the 95% confidence interval.

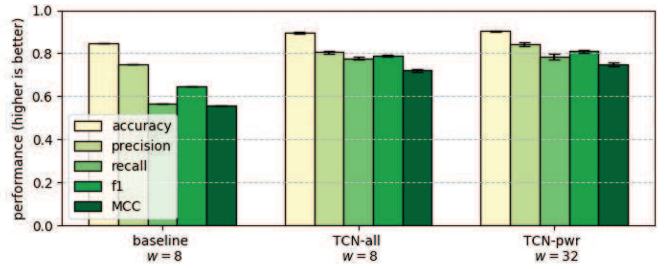

Fig. 7. Accuracy, precision, recall, F1 score and MCC performance results for the best performing variant of each method. The results shown are the mean scores over 100 different training runs. The error bars indicate the 95% confidence interval.

within the training set was used for validation. Only the trained weights for which the network scored best on this validation set were stored and later used for testing.

The reported performance measures are calculated on the classification results of the remaining 15% of the total dataset. These data were not used for training or validation during the model development. Since the model performs differently after each randomization of the order in the training set, the final mean performance was calculated over a total of 100 runs per model configuration. This resulted in a total of 1200 different models being trained.

At window size 8, the baseline model performs slightly worse than the TCN-based models in terms of its accuracy and precision (Fig. 7). However, recall and MCC is much lower than the TCN-based models for this window size. This suggests that even if an optimal window size is chosen for the baseline model, it still utilizes responsive batteries more poorly than the TCN-based models.

Using all metrics (TCN-all), the model outperforms the baseline model in many cases with minimal temporal information ($w = 1$), as shown in Fig. 6. Unlike the baseline model, the TCN-based model is unaware of the origin of the data, i.e., the battery number is not provided to the input. Performance increases with more temporal information. Precision improves as well. The performance of TCN-all decreases for window sizes larger than 8 hours. However, this drop in performance can be explained by the larger input size of the network which requires longer training times. This large input size also increases the chance of overfitting.

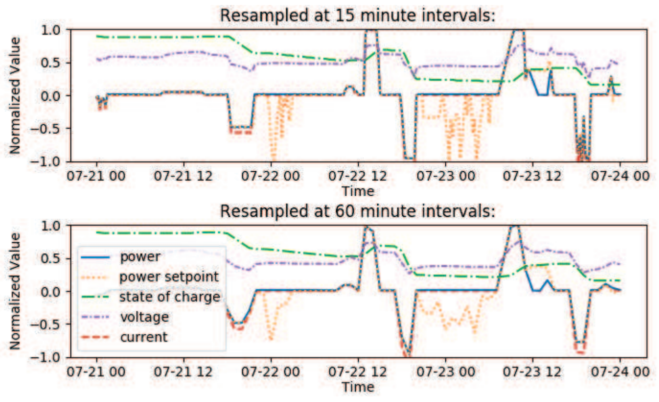

Fig. 8. An example of the raw data resampled at different intervals.

Using only the power and power setpoint metrics (TCN-pwr), the model performs poorly compared to the baseline model (Fig. 6) when minimal temporal information is used ($w = 1$). However, the more temporal information is used, the better the performance becomes. This suggests that there is relevant information captured in the temporal patterns of the power metric that compensates for the lack of information. When the window is set to 32 hours, the performance is better than that of the model using all metric information. This can be explained by the larger input size which typically increases the chances of overfitting while requiring more training epochs.

## C. Resampling Effects

The dataset used in the experiments uses a resampling interval of 60 minutes. The mean value is calculated within the interval. Fig. 8 provides an example of the effect of using a smaller resampling interval on the raw data. At higher resolutions, e.g., intervals of 15 minutes, the signals become noisier. When using a resampling interval of 15 minutes, the mean MCC performance (over 100 runs) for the best performing variants of the TCN models, TCN-all ($w = 8$) and TCN-pwr ($w = 32$), are $0.749$ and $0.794$ on the test set, respectively. This performance is similar to the original TCN models trained on the dataset using a larger resampling interval of 60 minutes, as shown in Fig. 7.

Providing the limited effect of using a smaller resampling interval, and the advantage of reducing noise, outliers and the size of the dataset at larger resampling intervals, justifies our choice for resampling the data at hourly intervals in this particular case. However, for other datasets with a higher power usage frequency, smaller resampling intervals are recommended.

## VI. Conclusion

In our paper, we showed how to use a Temporal Convolutional Neural Network (TCN) for a Digital Twin to predict CPES behavior. We used battery data from a real-life flexible energy system that are noisy, contains outliers and has large periods of missing data, to compare a baseline method, namely, the mean expected responsiveness to our TCN method. We have demonstrated that the model more accurately predicts whether batteries will respond to instructions they receive.

Our method benefits from using more temporal information, i.e., using larger window sizes. Accuracy performance, i.e., being correct in the predicted response, and precision performance, i.e., selecting relevant batteries that will actually respond to the negative setpoint of the model, improved. This implies that there is relevant information captured in the temporal patterns that cannot be inferred directly from the reported state of the battery.

Our results suggest that the temporal patterns can compensate for the lack of information, e.g., only knowing the power level in time, in CPES systems. This is useful to improve management of energy ecosystems that are composed of non-transparent or poorly documented systems. Despite not knowing important battery features, like battery type and age, knowing only battery behavior prior to prediction, the model still predicts well.

Our research proposes how Machine Learning can help, by way of the Digital Twin, to better utilize an existing system. The method minimizes the need to acquire or disclose potentially sensitive detailed information such as firmware, age or sensitive user information. Effectively, we extended Pileggi et al.'s work [1], shown in Fig. 1, to include an additional Machine Learning black-box model for the Digital Twin. This information can be sent to the controller via the flexibility information and assists in energy flexibility profile calibration.

Trust is an important aspect, particularly for energy control systems. The system should be safe and secure. Our model predicts behavior using only prior data, without needing sensitive information about the battery itself. Knowing the relationship between the Machine Learning model and the expert-model twins improves trust. Amongst others, it would help address system verification and validation.

In future work, we intend to experiment with energy assets in a controlled environment. Learning asset features and using that for energy flexibility profile calibration would make such systems more efficient and safer. Moreover, the aim would be to do this in a confidential and privacy-secure (safe) way, i.e., without having to expose sensitive information about the asset.

Furthermore, we aim to explore the possibilities of not just making both the expert model and Machine Learning model complementary to each other to provide different insights, like identifying anomalies and predicting behavior, but also to have them update each other. The aim of the latter is to make the Machine Learning model more explanatory while improving the performance and validating the rules of the expert model.

## Acknowledgment

The authors like to thank all parties involved in the Heerhugowaard flexible energy system for generating and providing the real-life battery data. The authors also like to thank their colleagues Coen van Leeuwen, Wilco Wijbrandi and Mente Konsman in specific for their insightful discussions and advice.

## References

[1] P. Pileggi, J. Verriet, J. Broekhuijsen, C. Van Leeuwen, W. Wijbrandi, and M. Konsman, "A digital twin for cyber-physical energy systems," *7th Workshop on Modeling and Simulation of Cyber-Physical Energy Systems, MSCPES 2019 - Held as part of CPS Week, Proceedings*, 2019.

[2] A. Rassa, C. van Leeuwen, R. Spaans, and K. Kok, "Developing Local Energy Markets: A Holistic System Approach," *IEEE Power and Energy Magazine*, vol. 17, no. 5, pp. 59–70, 2019.

[3] J. K. Kok, C. Warmer, and I. Kamphuis, "PowerMatcher: Multiagent Control in the Electricity Infrastructure," in *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, 2005, pp. 75–82.

[4] J. K. Kok, B. Roossien, P. MacDougall, O. van Pruissen, G. Venekamp, R. Kamphuis, J. Laarakkers, and C. Warmer, "Dynamic pricing by scalable energy management systems - Field experiences and simulation results using PowerMatcher," in *2012 IEEE Power and Energy Society General Meeting*, July 2012, pp. 1–8.

[5] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[6] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, p. 484, 2016.

[7] S. Bai, J. Z. Kolter, and V. Koltun, "An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling," *arXiv preprint arXiv:1803.01271*, 2018. [Online]. Available: http://arxiv.org/abs/1803.01271

[8] Z. Hameed, Y. Hong, Y. Cho, S. Ahn, and C. Song, "Condition monitoring and fault detection of wind turbines and related algorithms: A review," *Renewable and Sustainable Energy Reviews*, vol. 13, no. 1, pp. 1–39, 2009.

[9] M. Borth and L. Barbini, "Probabilistic health and mission readiness assessment at system-level," *Proceedings of the Annual Conference of the PHM Society*, vol. 11, 2019.

[10] C. J. Valant, J. D. Wheaton, M. G. Thurston, S. P. McConky, and N. G. Nenadic, "Evaluation of 1D CNN autoencoders for lithium-ion battery condition assessment using synthetic data," in *Proceedings of the Annual Conference of the PHM Society*, vol. 11, no. 1, 2019.

[11] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework," in *5th International Conference on Learning Representations (ICLR 2017)*, vol. 2, no. 5, 2017, p. 6.

[12] W. van der Aalst, *Process Mining: Data Science in Action*. Springer, 2016.

[13] P. Pileggi, A. Rivero-Rodriquez, and O. Nykanen, "Using context overlays to analyse the role of a priori information with Process Mining," *9th Annual IEEE International Systems Conference, SysCon 2015 - Proceedings*, pp. 639–644, 2015.

[14] W. Kritzinger, M. Karner, G. Traar, J. Henjes, and W. Sihn, "Digital Twin in manufacturing: A categorical literature review and classification," *IFAC-PapersOnLine*, vol. 51, no. 11, pp. 1016–1022, 2018.

[15] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, 2003.

[16] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," *arXiv preprint arXiv:1603.07285*, 2016.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[18] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016. [Online]. Available: https://arxiv.org/abs/1609.03499

[19] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," *Journal of Machine Learning Research*, vol. 15, pp. 315–323, 2011.

[20] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014. [Online]. Available: https://arxiv.org/abs/1412.6980

[21] F. Chollet *et al.*, "Keras: The Python deep learning library," *Astrophysics Source Code Library*, 2018.

[22] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[23] S. Boughorbel, F. Jarray, and M. El-Anbari, "Optimal classifier for imbalanced data using matthews correlation coefficient metric," *PloS one*, vol. 12, no. 6, p. e0177678, 2017.