Proceedings of the 2025 Winter Simulation Conference
E. Azar, A. Djanatliev, A. Harper, C. Kogler, V. Ramamohan, A. Anagnostou, and S. J. E. Taylor, eds.

ENERGY AND ACCURACY TRADE-OFFS IN DIGITAL TWIN MODELING: A COMPARATIVE STUDY OF AN AUTOCLAVE SYSTEM

Stijn Bellis[1], Joost Mertens[1], and Joachim Denil[1]

[1]Cosys-Lab, University of Antwerp, Antwerp, BELGIUM

ABSTRACT

Digital twinning is becoming increasingly prevalent and is creating significant value for organizations. When creating a digital twin, there are many modeling formalisms and levels of detail to choose from. However, designing, running, and deploying these models all consume energy. This paper examines the trade-offs between energy consumption and accuracy across different modeling formalisms, using an autoclave as a case study. We created a high-fidelity model of an autoclave, and from this model, several approximations were developed, including a 2D model and neural networks. The energy consumption of these models—as well as energy-intensive steps such as training the neural network—was measured and compared. This provides insights into the trade-offs between energy usage and accuracy for the selected modeling formalisms.

1 INTRODUCTION

Sustainable development focuses on development with three dimensions in mind: (a) Economic, (b) Social and (c) Environmental (Purvis et al. 2018). Sustainable development is defined as the development that fulfills today's needs without compromising future generations' needs. However, human emission of greenhouse gases and aerosols creates an imbalance in the Earth's energy system (Allan et al. 2021). Consequently, we see an increase in the earth's temperature, melting polar caps and permafrost regions. Computing as an industry is currently responsible for 2% to 6% of the emissions of greenhouse gases globally, with a predicted share of 6% - 22% in 2030 (Copenhagen Centre on Energy Efficiency 2020).

One of the key transformational technologies for the industry is the digital twin. Several definitions of a digital twin exist in the literature; as such, we will only state one: "A set of virtual information constructs that mimics the structure, context and behavior of an individual / unique physical asset, or a group of physical assets, is dynamically updated with data from its physical twin throughout its life cycle and informs decisions that realize value." (AIAA Digital Engineering Integration Committee 2020). Digital twins create value for companies by integrating the physical and digital world to address complexities and high demands from the market (Liu et al. 2021).

Digital twins have multiple functionalities and can be used to accomplish several goals. Applications include but are not limited to real-time monitoring, system optimization, quality control and waste management (Barricelli et al. 2019; Huang et al. 2021; Matta and Lugaresi 2024). Digital twins are applied in all industrial sectors like aeronautics, medical, smart city, and manufacturing.

While digital twins help in sustainable development's social and economic dimensions by optimizing system usage, the design of twins largely ignores its impact on electricity consumption. Digital twin implementations heavily rely on computing and networking infrastructure to monitor, predict, and optimize their analog counterparts. One of the pivotal choices in the design of a digital twin is which modeling formalism is going to be used. There are many options, from classical PDE models to machine learning models. Each of these modeling formalisms has its own benefits and drawbacks.

When choosing a modeling formalism, the value proposition of the digital twin for the user should be taken into account. This value proposition dictates the allowable uncertainty that the model can have. We

could always go for a complex model that has high validity with the real world, but these kinds of models are often very energy intensive. If the allowable uncertainty permits, we can use more approximate models. These models can be generated separately or, if it is available, a high validity model can be approximated to create surrogate models. While the modeling formalism can be very different in their execution, they will play the same role in the digital twin. To reason about the energy consumption of the resulting digital twin we introduced a simple additive model, as shown in equation 1, in a previous paper (Bellis and Denil 2022):

$$E_{total} = E_{design} + E_{local} + E_{networking} + E_{cloud} + E_{update}. \tag{1}$$

Where:

- E_{design} is the energy consumed for creating the twin. Building a simulation model for a twin might not have a large impact on this factor. However, this term might have a significant impact when using data-driven methods.
- E_{local} is the energy consumption at the analog side of the system (e.g., by storing the data, preprocessing the data, and executing a part of the twin model locally).
- $E_{networking}$ is the system's energy consumption by sending and receiving messages on the network.
- E_{cloud} is the energy consumption by executing the twin in the cloud environment.
- E_{update} is the energy necessary to redesign and update the model during the system's life-cycle.

Knowing and predicting the different energy usage for different modeling formalisms with different levels of abstraction and approximation will help us choose the most energy efficient modeling formalisms for a particular digital twin. In this paper, we are going to look at the E_{design} and the E_{local} for different modeling formalisms to get a better overview of the cost/benefit of different modeling formalisms.

For this, we use the running example of a small autoclave. Using the COMSOL software, a complex model of this autoclave was made. From this model, several other models were approximated. The energy usage of these different models was measured in the design and deployment phase. Then, the results of these simulations were also compared to give a good overview of the cost/benefit relation for each model.

To give an overview of the paper, in section 2 we go over some related work. In Section 3 we explain the running example in more detail, and in Section 4 we discuss the different modeling techniques used and the energy measurement setup. In section 5, we show and discuss the results and lastly, in section 6 we go over some future improvements.

2 RELATED WORK

In sustainable computing, there exist two notions of sustainability: (i) sustainability by IT and (ii) sustainability of IT (Pazienza et al. 2024). The former looks at how IT may positively affect the sustainability of a solution, while the latter looks at the sustainability of the IT system itself. In sustainable AI research a similar split exists between AI for sustainability and sustainability of AI (van Wynsberghe 2021).

In the field of digital twins, literature focuses almost solely on digital twins for sustainability (e.g. in agriculture (Purcell et al. 2023)) and not on the sustainability of digital twins themselves. On the latter topic, David and Bork (2023) present a 7R taxonomy of digital twin evolution for technical sustainability. It comprises 7 R-imperatives (the most well-known R-imperative is the 3R of reduce, reuse, recycle) applicable to a digital twin scenario: re-calibrate, re-model, re-collect, reconcile, re-deploy, re-configure and reuse. The taxonomy helps identify actions Digital Twin frameworks can take to support evolution in a sustainable manner. Furthermore, the notion of bipartite sustainability exists, that is, sustainable systems need to be engineered through sustainable methods to reach true sustainability. In this context, David, Bork, and Kappel (2024) explicitly mention Digital Twin as one of the methods that requires sustainability considerations.

To our knowledge, this is the first paper to specifically examine the carbon footprint of different modeling formalisms used in a digital twin context. As digital twins themselves are computing systems,

and machine learned models are often used in digital twins, this is a gap in research. Understanding the impact of modeling choices on energy usage and carbon footprint is essential to design sustainable digital twins. Where the related work which we mention next focuses on quantifying the energy consumption in the broader context of computing, this work goes further by reflecting back our findings to formulate guidelines for digital twin developers.

2.1 Carbon Footprint of Computing

Even with the tremendous advances in the energy efficiency of computing systems (Muralidhar et al. 2022), computing makes up a large part of a scientist's carbon footprint, especially when using supercomputers (Allen 2022). The key to reducing one's carbon footprint is the carbon intensity (g CO2/kWh) of the energy mix that is used. Computing during periods of high renewable energy generation reduces the carbon intensity. Consequently, the carbon footprint of the supercomputer or datacenter shifts from operational expenses (hardware operation) to capital expenses (hardware production) (Gupta et al. 2021). As a user, we can then focus on strategically scheduling our algorithms. Additionally, optimizing algorithms to be more efficient also impacts their carbon footprint(Allen 2022; Gupta et al. 2021).

Different tools exist to aid users in this regard. Green Algorithms (Green Algorithms 2025) is an online tool that estimates a computer program's carbon footprint based on the hardware and the location of the server (Lannelongue et al. 2021). Another way of expressing the carbon emissions is the Green Software Foundation's Software Carbon Intensity (SCI) specification. The SCI represents the rate of carbon emission of a software system (Green Software Foundation 2025a) and is formalized in the ISO/IEC 21031:2024 standard. To help users with scheduling the execution of their software, the Carbon Aware SDK (Green Software Foundation 2025b) or other similar API's such as WattTime (WattTime 2025), Electricity Maps (Electricitymaps 2025) and Climatiq (Climatiq 2025) can make the application carbon-aware.

2.2 Carbon Footprint of Machine Learning

In machine learning, the carbon footprint of training and inference is also a growing concern. The Machine Learning Emissions Calculator (ML CO2 Impact 2025) is a calculator that estimates the carbon footprint of training a machine learning model (Lacoste et al. 2019). The Experiment Impact Tracker (Henderson et al. 2020) is a framework that incorporates with Python code to track the carbon impact of a computer system. Similarly Carbontracker (Anthony et al. 2020) tracks and predicts energy use at runtime. Dodge et al. (Dodge et al. 2022) present a framework for estimating the carbon impact of GPU computation on cloud instances using the SCI metric. All of these frameworks share the same end goal: quantifying the carbon footprint of ML applications and raising awareness about its impact.

Within the AI community, there is also a movement towards Green AI, that is, AI models that takes into account the computational cost of the AI besides the traditional performance metrics like accuracy (Schwartz et al. 2020). At the current pace, the computational burden of deep learning is becoming technically and economically prohibitive (Thompson et al. 2020), making techniques that make machine learning more efficient even more important (Thompson et al. 2020; Mehlin et al. 2023).

3 RUNNING EXAMPLE

The running example that we use in this paper is a small autoclave. The autoclave consists of a wooden box with three horizontal copper plates at equidistant heights. The plates are connected to two opposing walls of the box and have an air gap on the two remaining sides. This air gap helps convection distribute the heat around the autoclave. On one of the sides with the air gap, there is a heating element that heats up the autoclave. On top of the copper plates, the objects that need to be sterilized are placed. We abstract the objects as plates that lay on top of the copper plates. The autoclave is shown in figure 1. There are two changeable parameters in this model: the heating rate of the heating element and the specific heat capacity of the objects in the autoclave.

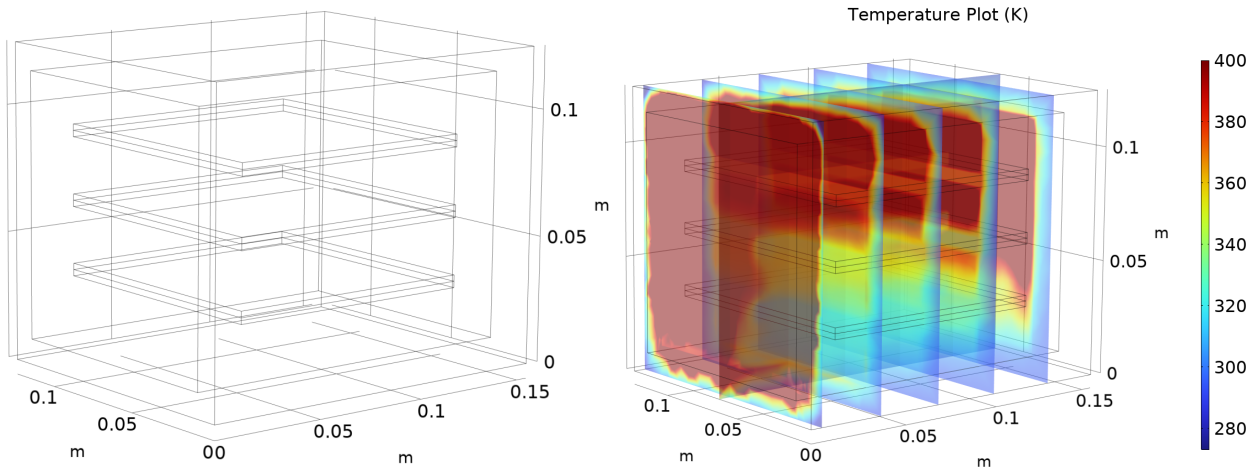

Figure 1: a) The geometry of the 3D COMSOL model. b) Temperature plot of the 3D COMSOL model.

The goal of the autoclave model is to sterilize the objects by heating them up to a temperature of 100°C. We use the model to predict when this will happen, so that we can stop the autoclave at the right time and conserve energy.

We modeled the autoclave in three dimensions using COMSOL. In this model, we included heat transfer by conduction, convection, and radiation. We use this model of the autoclave as a baseline and presume it to be the ground truth. We then approximated other models from this COMSOL model. We measured the energy consumption during the design and deployment of all the approximate models to see the cost/benefit for all the different approximate models.

4 SETUP OF EXPERIMENT

In this section, we discuss the setup of our experiments. First, we provide more information about the different approximate models we created. Then, we explain how we validate these models. Lastly, we describe the setup with which we measured the energy consumption of the different models.

4.1 2D COMSOL Model

As mentioned before, we approximate the model in several other approximate modeling formalisms. The first approximation we use is also a COMSOL model. We create a 2D model of the autoclave using COMSOL. We use the inherent symmetry of the autoclave and take a center 2D slice of the model. In this model, we also include heat transfer via conduction, convection, and radiation.

4.2 Lumped Parameter Model

The second approximate model we create is a lumped parameter model using Simulink. For this model, we define several thermal zones (as shown in Figure 2). In each thermal zone, we assume the temperature is uniform. These thermal zones interact with each other and the surroundings. The modeled heat transfers in this lumped parameter model are conduction and convection.

For conduction, the influence of thermal zone j on thermal zone i is given by equation 2.

$$\frac{1}{C_i} I_{ij}(T_j - T_i) \tag{2}$$

Here, I_{ij} is a constant that we tune to model the conduction between thermal zones of type i and j. T_i and T_j represent the temperatures of the respective thermal zones, and C_i is the heat capacity of

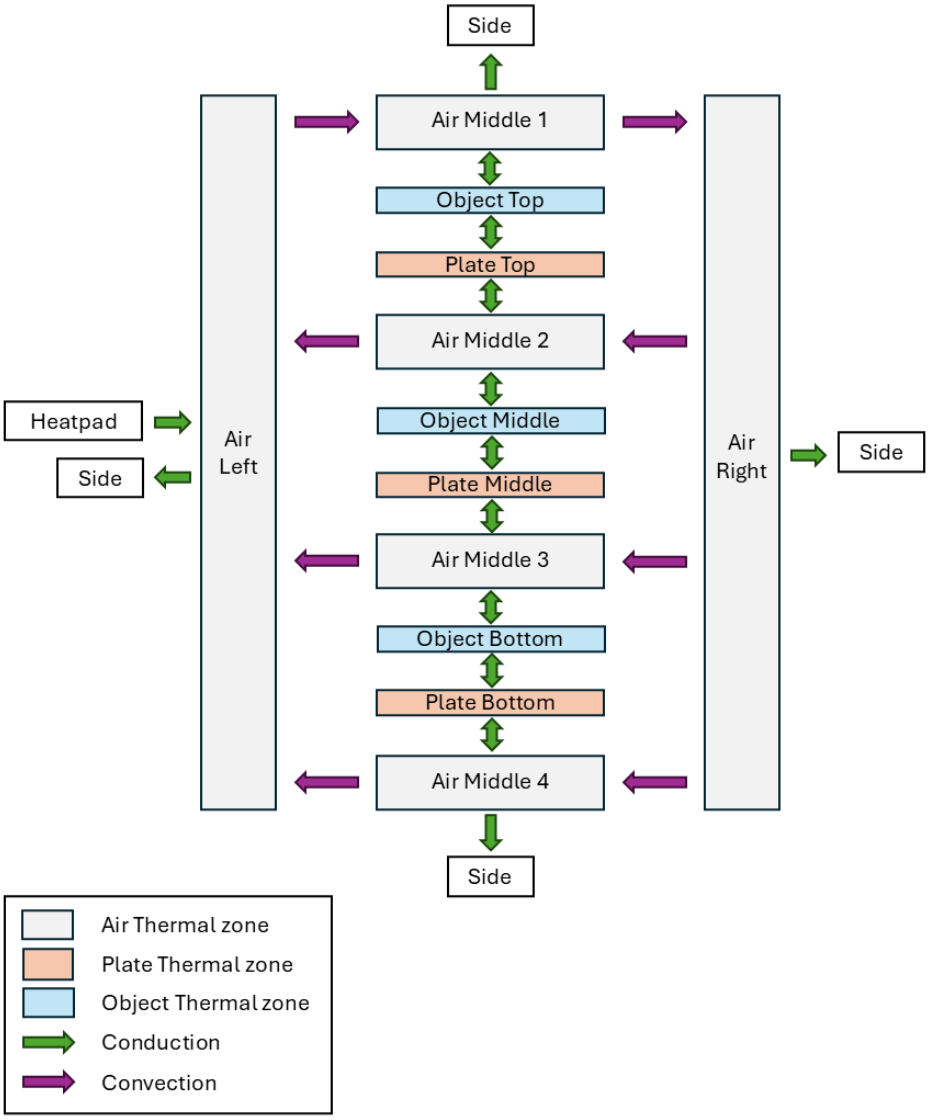

Figure 2: Overview of the lumped parameter model.

thermal zone i. We define three constants for the conduction between thermal zones: $I_{\text{air-plate}}$, $I_{\text{air-object}}$, and $I_{\text{object-plate}}$, representing conduction between air and plate zones, air and object zones, and object and plate zones, respectively.

Additionally, we introduce two constants for conduction with the surroundings: I_{heat} and I_{side}. I_{heat} models the conduction between the heat pad and interacting thermal zones, while I_{side} models the conduction between the air thermal zones on the sides and the wooden box. For conduction with the heat pad, the constant is multiplied by the heat rate of the heat pad rather than a temperature difference. For conduction to the sides, the constant is multiplied by the length of the contacting surface (which differs for different thermal zones) and the temperature difference between the thermal zone and room temperature (20°C).

For convection, the air flow is fixed and follows a set path: from thermal zone air left to thermal zone air middle 1, then to thermal zone air right. Here, the flow splits towards thermal zones air middle 2, 3, and 4. From there, the different flows return to thermal zone air left. The effect of heat transfer via convection

from thermal zone j to thermal zone i is modeled by (3).

$$\frac{1}{C_i} F_i (T_j - T_i), \tag{3}$$

Where F_i represents the flow rate from zone j to zone i. We assume incompressible air, so we only need to calibrate three flow rates: the flow from thermal zone air right to thermal zones air middle 2, 3, and 4. The remaining flow rates are derived from these.

We calibrate the Simulink model using the simulation trace from the 3D COMSOL model with a heat rate of 250 W and an object-specific heat capacity of 700 ($\frac{J}{kgK}$). For calibration, we use the built-in parameter estimator in Simulink with the cost function defined as the sum of squared errors.

4.3 Neural Networks

For this paper, we train two types of neural networks. The first type is the *simulation neural network*. These networks take the current simulation step as input and predict the difference to the next simulation step. For this paper, the current simulation step consists of the time, the heat rate, the specific heat capacity of the object, and the temperature at the center of each object placed on the top, middle, and bottom plates. The output of the neural network will be the difference in the temperature at the center of each object in 0.5 s. We only track the temperatures of the objects, as we found this yields more accurate results. Using these neural networks, we can simulate the autoclave dynamics. To study the effect of the amount of training data, we train two simulation neural networks: one using the full training dataset and one using only half of the training data.

The second type is the *end-time neural network*. This network takes the heat rate and object-specific heat capacity as inputs and outputs the time when all objects reach 100°C.

For neural network training, we chose to use smaller fully connected neural networks. There are many architectures and changes that could be tried for the neural networks. But we use these fully connected neural networks as a starting point. we use TensorFlow to implement the neural networks. We perform hyperparameter tuning for three parameters: the learning rate, the number of nodes per layer, and the number of layers. The learning rate is sampled logarithmically between 10^{-5} to 0.1. The number of nodes per layer is sampled from 16 to 96 with a step size of 16. The number of layers ranges from 4 to 6 for the simulation neural networks, and from 2 to 4 for the end-time neural network. The activation function for the neural networks is ReLU. Hyperparameter tuning is performed using the Hyperband algorithm, with a max epochs of 80 and a learning factor of 3.

Table 1: Found hyperparameters for the neural networks.

Model	**Amount of nodes**	**Amount of hidden layers**	**Learning rate**
Big simulation NN	96	6	0.0003267294024466933
Small simulation NN	80	4	0.00017768967038033528
End time NN	96	2	0.04485067855198114

The hyperparameters we found for all neural networks are summarized in table 1. The test loss for the big and small simulation neural network are 2.35×10^{-6} and 1.97×10^{-6} respectively. The test loss for the end time neural network is 15.41.

4.4 Validation Setup

All the models were validated across several combinations of heat rate and object-specific heat capacity. These combinations were sampled from a 10x10 grid, with heat rates ranging from 200 W to 300 W, and object-specific heat capacities ranging from 500 $\frac{J}{kgK}$ to 900 $\frac{J}{kgK}$. Additionally, the centers of every four

points in this grid were sampled to create an additional 9x9 grid of combinations. In total, this results in 181 different combinations.

All models were simulated for each of these combinations, and the simulations were stopped once all objects reached a temperature of 100°C. For training and testing the simulation neural networks, simulation traces were generated using the 3D COMSOL model. The points from the 10x10 grid were split into training and test datasets. The training dataset included the same number of points as the 9x9 grid. The remaining points from the 10x10 grid were assigned to the test dataset. The big simulation neural network was trained on the combined points from the training dataset of the 10x10 grid and the full dataset of the 9x9 grid. The small simulation neural network was trained only on the training dataset derived from the 10x10 grid. For the end time NN the training dataset contains 145 data points and the test dataset contains 36 data points.

4.5 Energy Measurement Setup

Energy measurements were conducted on a standalone computer equipped with a Intel Core i9-14900K CPU and a MSI GeForce RTX 4080 GPU. For measuring energy usage, we employed powerstat to record the energy usage of the processor package, DRAM controller, CPU core and graphics uncore. Powerstat uses the Intel RAPL interface to measure the energy consumption (King, Colin 2025). GPU energy consumption was recorded using nvidia-smi. Measurements were taken at a frequency of once per second.

The energy consumption was recorded for running all the models, as well as during hyperparameter tuning, training of all the neural networks, and the calibration of the lumped parameter model constants. Each complete set of measurements was repeated 34 times, with the order of the experiments randomized in each run to account for the potential variability in system performance. The computer was also disconnected from screens, keyboards, Wi-Fi and Bluetooth appliances to reduce the variance in the energy measurements.

5 RESULTS AND DISCUSSION

In this section we discuss the results of the experiments and draw some conclusions from them. We start with the results of the validation experiments followed by the results of the energy measurements. We combine these results to draw conclusions about the efficiency of the different models in function of the number of simulations. Finally, we discuss other impacting factors on the energy efficiency.

5.1 Validation Results

For model validation, we simulated 181 combinations of heat rate and the specific heat capacity of the object. The 2D COMSOL model, however, failed to converge for two of these combinations, so only 179 simulation traces were used in its evaluation. We recorded the time at which the objects reached 100°C in each simulation and compared these times to the corresponding values from the 3D COMSOL model. We also computed the maximum absolute difference in end times between each model and the 3D COMSOL model. These results are presented in Table 2.

We use the maximum end time difference as a safety margin for each model. To estimate the additional energy cost associated with this safety margin, we multiplied it by the heat rate. The mean extra energy cost is reported in table 2 as well.

We observe that the 2D COMSOL model has the largest mean absolute end time difference among all models. This is likely because it was the only model not calibrated or tuned using data from the 3D COMSOL simulations. The discrepancy between the 2D and 3D COMSOL models likely stems from the 2D model's omission of heat loss through the sides of the autoclave, which causes a quicker heat up.

The lumped-parameter model shows a mean absolute end time difference of 90.22 seconds, which can be attributed to the simplifying assumptions made during its construction.

The neural network models demonstrate high accuracy overall. The most accurate is the big simulation neural network, likely due to its training on the largest dataset. Next in accuracy is the small simulation

Table 2: Validation results of the various models.

Model	**Mean absolute end time difference**	**Max absolute end time difference**	**Mean extra energy cost**
2D COMSOL model	212.84 s	291.5 s	72875 J
Lumped parameter model	90.22 s	133.0 s	33250 J
Big simulation NN	0.92 s	3.0 s	750 J
Small simulation NN	3.58 s	9.5 s	2375 J
End time NN	2.38 s	16.4 s	4100 J

neural network, followed by the end time neural network. Despite being trained on only 145 data points, the end time NN still outperforms both the 2D COMSOL and lumped-parameter models.

5.2 Energy Measurement Results

This section discusses the energy measurement results.. The mean energy usage for all the experiments with the standard error, is shown in figure 3.

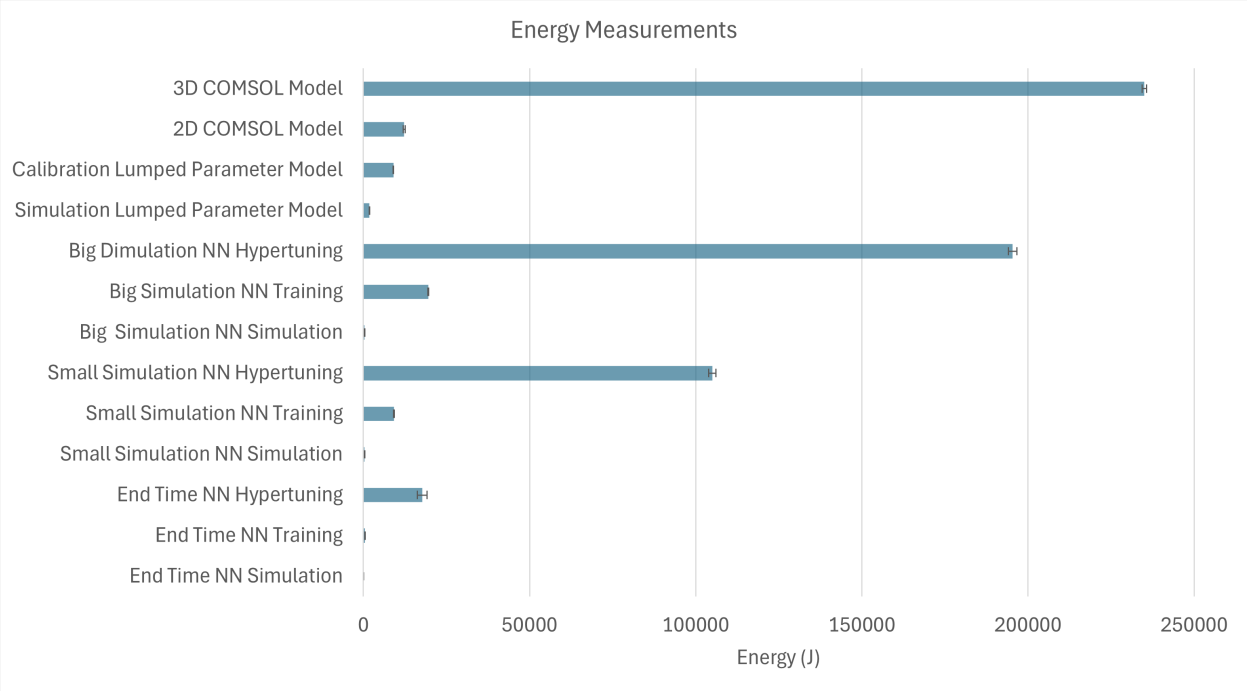

Figure 3: Energy measurements for the various models.

The experiments with the highest energy usage were the 3D COMSOL model, the big simulation NN hyperparameter tuning, and the small simulation NN hyperparameter tuning. This result is expected, as both 3D simulations and hyperparameter tuning are computationally expensive processes.

In contrast, the end time NN hyperparameter tuning, training, and simulation consumed significantly less energy than the simulation NN experiments. This is because the simulation NNs were trained on 190091 and 95046 data points respectively, whereas the end time NN was trained on only 145 data points.

5.3 Discussion

Our results demonstrate that energy consumption varies significantly across different modeling formalisms, with distinct trade-offs between design-time and runtime. Specifically, the 3D COMSOL model and neural network hyperparameter tuning are the most energy-intensive during the design phase, whereas operational energy consumption is primarily driven by physics-based simulation executions. These findings emphasize that modeling decisions made early in the design process have lasting impacts on total energy consumption throughout the digital twin's life cycle.

Accuracy also varies notably across modeling approaches. The 2D COMSOL model exhibits the lowest accuracy, with a mean absolute end-time difference of 212.84 s, while the big neural network achieves the highest accuracy, with just a 0.92 s deviation. These accuracy disparities directly influence energy consumption through the mean extra energy cost metric, which quantifies the additional energy required to maintain safety margins that compensate for model inaccuracies. For example, the 2D COMSOL model requires an average of 72,875 J in extra energy, while the neural network demands only 750 J. This illustrates a key insight: greater accuracy reduces the need for conservative safety margins, thereby lowering energy use during operation.

Using this information, we calculate total energy cost using an adapted additive energy model (4).

$$E_{total} = E_{design} + S * E_{local} \tag{4}$$

Where E_{total} is the total energy usage, E_{design} is the design energy cost, E_{local} is the local operational energy cost and S is the number of simulations with the model.

Figure 4 illustrates how total energy varies with the number of simulations. The results reveal a distinct

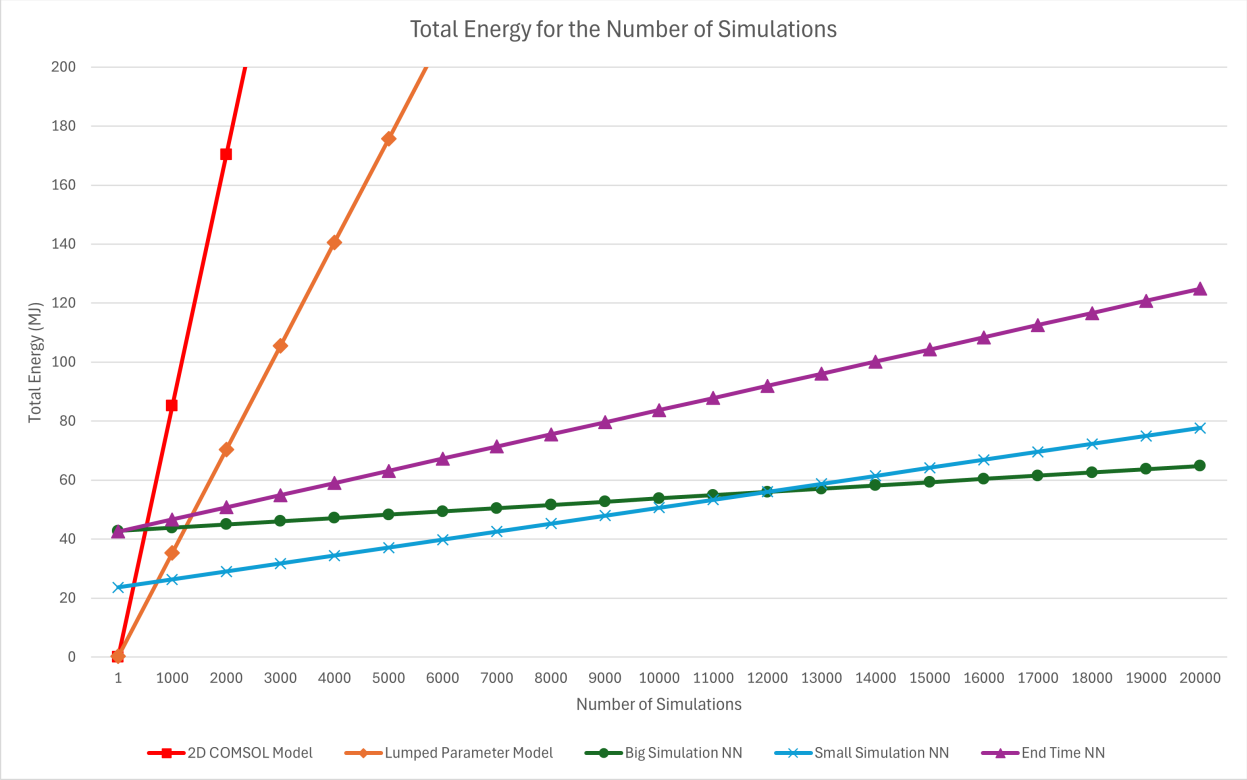

Figure 4: Total energy measurements in function of the number of simulations.

energy-accuracy trade-off pattern. For scenarios involving a limited number of simulations, classical models like the 2D COMSOL and lumped parameter models are the most energy-efficient, despite their lower

accuracy. Notably, the 2D COMSOL model is optimal for the first four simulations, after which the lumped parameter model becomes more efficient. However, as the number of simulations grows, the operational energy savings and superior accuracy of neural networks eventually compensate for their high training costs. In our study, the small neural network becomes the most energy-efficient option after approximately 792 simulations. The big neural network becomes the most energy-efficient option after approximately 11961 simulations. These crossover points mark critical decision thresholds for digital twin deployment strategies.

We also examined the influence of the safety margin definition on these thresholds. Our primary analysis used the maximum absolute end-time difference as the safety margin, but if we instead consider only undershoot errors—those that pose actual safety risks—the transition points shift. The crossover from 2D COMSOL to the lumped parameter model occurs after just three simulations (down from five), while the big neural network surpasses the lumped parameter model after 58379 simulations. This shift occurs because the lumped parameter model typically overestimates end times. As a result, additional safety energy is often unnecessary.

These transition points offer essential guidance for digital twin developers aiming to optimize energy use under varying deployment scales and accuracy demands. In large-scale deployments, such as factories with multiple identical systems (e.g., 50 autoclaves), the cumulative number of simulations accelerates the advantage of neural networks. In such a scenario, the energy-efficiency threshold is met after just 26 simulations per device.

Model selection can also be understood through a formalism transformation graph (Vangheluwe 2000), which outlines relationships and approximations between modeling approaches. While our analysis emphasizes energy consumption and accuracy, comprehensive model selection must consider additional factors: required precision, deployment context, model validity regions, and acceptable uncertainty levels. For instance, in our autoclave case study, high accuracy within the operational region is essential to avoid hazardous outcomes, minimize uncertainty, and reduce the need for costly compensatory heating. However, applications with higher tolerance for uncertainty may benefit from simpler, less energy-intensive models.

System evolution introduces further complexity. modeling approaches vary in their adaptability to changes in boundary conditions, physical components, and intended use. Physics-based models typically require only parameter adjustments when operating conditions shift, whereas data-driven models often demand complete retraining—an energy-intensive process. In our autoclave scenario, significant changes to thermal load characteristics would necessitate neural network retraining, potentially erasing their operational energy advantages. Therefore, classical models may offer more sustainable long-term solutions in dynamic environments due to their adaptability and lower reconfiguration energy costs.

Based on our findings, we offer several guidelines for selecting energy-efficient modeling approaches:

1. **Expected simulation volume** is a key determinant, with clear transition points between optimal models.
2. **System evolution frequency** should be considered, as frequent changes favor more adaptable classical models.
3. **Accuracy requirements** must be balanced against the energy cost of achieving high accuracy.
4. **Resource availability** during design and operation phases may constrain feasible modeling choices.
5. **Deployment scale** should factor into total energy cost assessments, especially in replicated systems.

Besides these guidelines, designers should also use common sense and best engineering practices. For example, when high accuracy is needed, e.g. to properly design a product, it would be best to stick to the 3D COMSOL model. For a digital twin that predicts the duration of a sterilization task, using the most energy-intensive approach is not useful. Likely the product also lacks the computing capabilities to run it. For such a case, our results can help with making an appropriate choice between the derived models.

In summary, neural networks are ideal for mass-produced systems with stable conditions and high simulation counts, while classical models are better suited to low-volume or frequently evolving applications.

Sustainable digital twin implementation hinges on choosing the right modeling formalism based on a holistic understanding of energy trade-offs, system dynamics, and operational context.

6 FUTURE WORK AND CONCLUSION

In this paper, we examined the trade-offs between energy consumption and accuracy across different modeling formalisms, using an autoclave system as a case study. Starting with a high fidelity 3D COMSOL model, we derived several lower-fidelity approximations, including both classical physics-based and data-driven neural network models. We measured and compared the energy consumption of each approach, not only during operational simulation but also accounting for energy-intensive steps such as neural network hyperparameter tuning and training. This analysis revealed clear trade-offs between model validity, energy use, and predictive accuracy, as well as the critical thresholds at which it becomes advantageous to switch from one modeling formalism to another based on deployment scale and accuracy requirements.

While the findings offer valuable insights into the relationship between energy consumption and modeling accuracy, there are several opportunities for future work. This study focused on a limited set of modeling formalisms, and extending the analysis to a broader range of classical and machine learning approaches would help generalize the results. Additionally, the parameter space explored was relatively narrow, primarily constrained to a small range of heat rates and object-specific heat capacities. Within this range, the models demonstrated relatively consistent behavior across input variations. Expanding the parameter space to include more extreme operating conditions could reveal new trade-offs and potentially shift the cost-benefit dynamics between modeling approaches.

ACKNOWLEDGMENTS

We want to thank Ward Goossens for his help with setting up the energy measurement set up.

REFERENCES

AIAA Digital Engineering Integration Committee 2020. "Digital Twin: Definition & Value—An AIAA and AIA Position Paper". https://www.aia-aerospace.org/publications/digital-twin-definition-value-an-aiaa-and-aia-position-paper/.

Green Algorithms 2025. https://www.green-algorithms.org/, accessed 26.06.2025.

Allan, R. P., E. Hawkins, N. Bellouin, and B. Collins. 2021. "IPCC, 2021: Summary for Policymakers".

Allen, M. 2022, August. "The Huge Carbon Footprint of Large-Scale Computing". *Physics World* 35(3):46–50 https://doi.org/10.1088/2058-7058/35/03/32.

Anthony, L. F. W., B. Kanding, and R. Selvan. 2020. "Carbontracker: Tracking and Predicting the Carbon Footprint of Training Deep Learning Models" https://doi.org/10.48550/ARXIV.2007.03051.

Barricelli, B. R., E. Casiraghi, and D. Fogli. 2019. "A Survey on Digital Twin: Definitions, Characteristics, Applications, and Design Implications". *IEEE access* 7:167653–167671.

Bellis, S., and J. Denil. 2022. "Challenges and Possible Approaches for Sustainable Digital Twinning". In *Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*. October 23th-28th, Montreal, Canada, 643-648 https://doi.org/10.1145/3550356.3561551.

Climatiq 2025. https://www.climatiq.io/, accessed 26.06.2025.

Copenhagen Centre on Energy Efficiency 2020. "Greenhouse Gas Emissions in the ICT Sector: Trends and Methodologies". https://c2e2.unepccc.org/wp-content/uploads/sites/3/2020/03/greenhouse-gas-emissions-in-the-ict-sector.pdf, accessed 26.06.2025.

David, I., and D. Bork. 2023. "Towards a Taxonomy of Digital Twin Evolution for Technical Sustainability". In *2023 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, October 1st–6th, Västerås, Sweden, 934–938 https://doi.org/10.1109/MODELS-C59198.2023.00147.

David, I., D. Bork, and G. Kappel. 2024, April. "Circular systems engineering". *Software and Systems Modeling* 23(2):269–283 https://doi.org/10.1007/s10270-024-01154-4.

Dodge, J., T. Prewitt, R. Tachet des Combes, E. Odmark, R. Schwartz, E. Strubell, *et al*. 2022. "Measuring the Carbon Intensity of AI in Cloud Instances". In *2022 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '22, June 21th–24th, Seoul, South Korea, 1877–1894 https://doi.org/10.1145/3531146.3533234.

Electricitymaps 2025. https://www.electricitymaps.com/, accessed 26.06.2025.

Green Software Foundation 2025a. https://sci.greensoftware.foundation/, accessed 26.06.2025.

Green Software Foundation 2025b. https://github.com/Green-Software-Foundation/carbon-aware-sdk, accessed 26.06.2025.

Gupta, U., Y. G. Kim, S. Lee, J. Tse, H.-H. S. Lee, G.-Y. Wei, *et al*. 2021. "Chasing Carbon: The Elusive Environmental Footprint of Computing". In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, February 27th– March 3th, Seoul, South Korea, 854–867 https://doi.org/10.1109/HPCA51647.2021.00076.

Henderson, P., J. Hu, J. Romoff, E. Brunskill, D. Jurafsky, and J. Pineau. 2020. "Towards the Systematic Reporting of the Energy and Carbon Footprints of Machine Learning" https://doi.org/10.48550/ARXIV.2002.05651.

Huang, Z., Y. Shen, J. Li, M. Fey, and C. Brecher. 2021. "A Survey on AI-Driven Digital Twins in Industry 4.0: Smart Manufacturing and Advanced Robotics". *Sensors* 21(19):6340.

ML CO2 Impact 2025. https://mlco2.github.io/impact/, accessed 26.06.2025.

King, Colin 2025. "Powerstat". https://github.com/ColinIanKing/powerstat, accessed 26.06.2025.

Lacoste, A., A. Luccioni, V. Schmidt, and T. Dandres. 2019. "Quantifying the Carbon Emissions of Machine Learning" https://doi.org/10.48550/ARXIV.1910.09700.

Lannelongue, L., J. Grealey, and M. Inouye. 2021, May. "Green Algorithms: Quantifying the Carbon Footprint of Computation". *Advanced Science* 8(12) https://doi.org/10.1002/advs.202100707.

Liu, M., S. Fang, H. Dong, and C. Xu. 2021. "Review of Digital Twin About Concepts, Technologies, and Industrial Applications". *Journal of Manufacturing Systems* 58:346–361.

Matta, A., and G. Lugaresi. 2024. "An Introduction to Digital Twins". In *2024 Winter Simulation Conference (WSC)*, 1281–1295 https://doi.org/10.1109/WSC63780.2024.10838793.

Mehlin, V., S. Schacht, and C. Lanquillon. 2023. "Towards Energy-Efficient Deep Learning: An Overview of Energy-Efficient Approaches Zlong the Deep Learning Lifecycle" https://doi.org/10.48550/ARXIV.2303.01980.

Muralidhar, R., R. Borovica-Gajic, and R. Buyya. 2022, January. "Energy Efficient Computing Systems: Architectures, Abstractions and Modeling to Techniques and Standards". *ACM Computing Surveys* 54(11s):1–37 https://doi.org/10.1145/3511094.

Pazienza, A., G. Baselli, D. C. Vinci, and M. V. Trussoni. 2024, February. "A Holistic Approach to Environmentally Sustainable Computing". *Innovations in Systems and Software Engineering* 20(3):347–371 https://doi.org/10.1007/s11334-023-00548-9.

Purcell, W., T. Neubauer, and K. Mallinger. 2023, April. "Digital Twins in Agriculture: Challenges and Opportunities for Environmental Sustainability". *Current Opinion in Environmental Sustainability* 61:101252 https://doi.org/10.1016/j.cosust.2022.101252.

Purvis, B., Y. Mao, and D. Robinson. 2018, September. "Three Pillars of Sustainability: in Search of Conceptual Origins". *Sustainability Science* 14(3):681–695 https://doi.org/10.1007/s11625-018-0627-5.

Schwartz, R., J. Dodge, N. A. Smith, and O. Etzioni. 2020, November. "Green AI". *Communications of the ACM* 63(12):54–63 https://doi.org/10.1145/3381831.

Thompson, N. C., K. Greenewald, K. Lee, and G. F. Manso. 2020. "The Computational Limits of Deep Learning" https://doi.org/10.48550/ARXIV.2007.05558.

van Wynsberghe, A. 2021, February. "Sustainable AI: AI for Sustainability and the Sustainability of AI". *AI and Ethics* 1(3):213–218 https://doi.org/10.1007/s43681-021-00043-6.

Vangheluwe, H. 2000. "DEVS as a common denominator for multi-formalism hybrid systems modelling". In *CACSD. Conference Proceedings. IEEE International Symposium on Computer-Aided Control System Design (Cat. No.00TH8537)*, September 25th–27th, Anchorage, AK, USA, 129–134 https://doi.org/10.1109/CACSD.2000.900199.

WattTime 2025. https://watttime.org/, accessed 26.06.2025.

AUTHOR BIOGRAPHIES

STIJN BELLIS is a PhD student at the University of Antwerp, Cosys-Lab. His research looks at the sustainability of digital twins, and in particular the way different modeling formalisms influence the sustainability of a digital twin. His email address is stijn.bellis@uantwerpen.be.

JOOST MERTENS is a post-doctoral researcher at the University of Antwerp, Cosys-Lab. His research interest is the continuous validation of Digital Twins in the context of evolving Cyber-Physical Systems. Particularly, he is interested in the synchronization of digital twin systems. His email address is joost.mertens@uantwerpen.be.

JOACHIM DENIL is an associate professor at the University of Antwerp, Cosys-Lab. His research interests are enabling methods, techniques and tools to design, verify and evolve Cyber-Physical Systems. Specifically, he is interested in the performance modeling and simulation, model-based systems engineering, and verification and validation of models and systems. He serves as an associate editor of the Transaction of the SCS: Simulation. His email address is joachim.denil@uantwerpen.be.