Energy Proceedings
Vol 44, 2024

ISSN 2004-2965

Interpretable Machine Learning for Reinforcement Learning based Control Strategy Optimization of Industrial Energy Supply Systems

Arthur Stobert*, Heiko Ranzau, Inga Pfenning, Matthias Weigold

Technical University of Darmstadt, Institute of Production Management, Technology and Machine Tools (PTW), Otto-Berndt-Str. 2, 64287 Darmstadt, Germany
(*Corresponding Author: A.Stobert@ptw.tu-darmstadt.de)

ABSTRACT

In the face of current global energy challenges and the growing significance of energy efficiency and carbon neutrality, the optimization of control strategies for industrial energy supply systems has gained importance. Deep Reinforcement Learning (DRL) presents a promising opportunity for control strategy optimization, leading to substantial reductions in operating costs and carbon emissions. The lack of interpretability of such black box models however is hindering broader application in practice.

This paper introduces an Interpretable Machine Learning approach aiming to reconcile advanced optimization with interpretability. Initially, DRL algorithms are deployed on a simulation model of an industrial energy supply system for control strategy optimization. Training and validation data sets of the DRL based control strategy are then used to train Decision Trees. Being intrinsically interpretable, Decision Trees enable representation of conditional logic and the extraction of rules for both local and global interpretability. Through optimization procedures such as depth limitation and input feature selection, the Decision Trees can either operate as controllers for energy supply systems directly or as tools to adapt the conventional rule-based control strategy.

Tested successfully on a white-box model – the conventional rule-based control strategy for an industrial energy supply system – the proposed approach correctly identifies the underlying rules of the strategy. When applied on a black box model - the DRL based control strategy of the exemplary use case - in various experiments this paper showcases promising results as well as remaining challenges of the approach.

Keywords: explainable artificial intelligence, interpretable machine learning, intelligent energy, industrial energy supply systems, control strategy optimization, deep reinforcement learning

NOMENCLATURE

Abbreviations

RL	Reinforcement Learning
IESS	Industrial Energy Supply Systems
DNN	Deep Neural Network
OS	Operational Strategies
DRL	Deep Reinforcement Learning
AI	Artificial Intelligence
XAI	Explainable Artificial Intelligence
ML	Machine Learning
IML	Interpretable Machine Learning
DT	Decision Tree
CHP	Combined Heat and Power Unit
CCP	Cost-Complexity Pruning
B	Boiler
HP	Heat pump
CT	Cooling Tower
IH	Immersion Heater
HS	Hot Water Storage
CS	Cold Water Storage
CC	Compression Chiller

1. INTRODUCTION

In industry, energy carriers like heating and cooling water, electricity, gas, and compressed air are essential for factory processes such as powering production machines and regulating building temperature [1,2]. While some carriers like electricity and gas can be sourced directly from higher-level networks, others must

# This is a paper for 15th International Conference on Applied Energy (ICAE2023), Dec. 3-7, 2023, Doha, Qatar.

be provided from the factory's industrial energy supply systems (IESS), necessitating an operational strategy (OS) that needs to be reliable, cost-effective, and environmentally conscious [3,4]. In the context of increasingly volatile energy prices, a growing complexity of IESS and competing objectives, energy and cost savings can be achieved through an optimized OS [1].

For this, Deep Reinforcement Learning (DRL) methods are a promising alternative to conventional, rule-based OS. DRL methods have been applied in simulation-based IESS, showing reductions in energy consumption and greenhouse gas emissions [1,5,6].

Despite these promising results, there are few real-world applications in which DRL methods apply the OS on real systems. We attribute this to their lack of interpretability, stemming from the deep neural networks involved, leading to missing trust and acceptance for the application in high stakes decision making in critical environments like supply engineering (compare section 2.4). To solve this problem the research field of Explainable Artificial Intelligence (XAI) deals with methods and techniques to make black-box models like DRL interpretable [7]. Thus, achieving interpretability can improve trust and transparency of AI-based systems.

Against this background, this paper presents an Interpretable Machine Learning (IML) approach aiming to make DRL based OS optimization of IESS interpretable. Following this introduction, section 2 presents the fundamentals of IESS and their conventional OS, Reinforcement Learning (RL), XAI, and Decision Trees (DT). Section 3 covers the overall method and its implementation. The use case and the application of the approach in different experiments are shown in section 4 followed by their evaluation in Section 5. A conclusion and outlook are finally given in section 6.

2. FUNDAMENTALS

2.1 Industrial energy supply systems

The IESS of a factory are all facilities required for conversion of final and environmental energy as well as the storage and transportation of the useful energy required for the operation of the production processes and the factory building [8].

In different industries, IESS account for a significant portion of the end energy demand. While the production processes mostly rely on specialized equipment, IESS predominantly use cross-sectional technologies (e.g., heat pumps, refrigeration units, heat exchangers, cooling towers, combined heat and power plants, air compressors, ventilation systems) enabling OS optimization measures to be applied to a wide variety of industrial plants. [1]

2.2 Conventional control strategies for IESS

In light of their straightforward design and economical implementation expenses, conventional rule-based techniques, including two-point controllers and PID controllers, are predominantly employed for the regulation and automation of IESS [9].

Two-point controllers, often known as on-off controllers, are the simplest form of feedback controllers. These controllers operate by switching the controlled device either fully on or fully off, based on the difference between the process variable and the set point. When the process variable deviates from the desired set point, the controller responds by toggling the system's status, ensuring basic regulation without the fine-tuned adjustments of more intricate controllers.

For complex IESS, however, it is challenging to identify optimal PID or two-point controller based OS due to a variety of challenges (such as multivalent energy forms, intricate facilities-environment interactions, stochastic disturbances, integrated energy storages, cost efficiency, environmental considerations, and supply security [1]) and also to translate the OS into programmable rules. This limits the performance of conventional control processes [9,10], signifying the need for more complex optimization approaches like Reinforcement Learning (RL).

2.3 Reinforcement Learning

RL is a machine learning paradigm with the primary goal to map system states S_t to actions A_t in order to maximize a scalar reward signal R_t over the long term. This is achieved using an iterative trial-and-error approach. Within the RL framework, the entity responsible for decision-making is denoted as the *agent*, and the system with which this agent engages is referred to as the *environment* (compare Fig. 2). [11]

DRL is an extended form of reinforcement learning in which the value of a state, the policy or the system model is approximated by a deep neural network (DNN). [1]

For a more detailed introduction to RL and DNN we refer to [11] and [12] respectively.

2.4 XAI

According to the Defense Advanced Research Projects Agency's (DARPA) the goal of the research field of Explainable AI is to ”produce more explainable models, while maintaining a high level of learning performance (prediction accuracy); and enable human users to understand, appropriately trust, and effectively manage the emerging generation of artificially intelligent partners" [13]. Being able to explain their rationale and to point out their strengths and weaknesses new machine-learning systems will also have the ability to give its human end users an understanding of its future behavior. To achieve this goal, DARPA's strategy is to combine modified or totally new developed machine learning techniques into models with both high accuracy and interpretability. [13] For one of the main reasons and at the same time the biggest challenge in producing interpretable models is the trade-off between interpretability and accuracy in current machine learning models [14]. Simple machine learning models like Linear Regression or DTs and their decision-making process have a high interpretability. This is why they are called white-box models. Compared to Deep Neural Networks or Support Vector Machines, they tend to show low accuracy. On the other hand, the black-box models have little to no interpretability at all.

Based on Adadi et al. [7] existing XAI methods are classified into three groups according to the *complexity* of a machine learning model to be explained, the *scope* of interpretability, and the methods *dependency* on the machine learning model.

The complexity of a machine learning model decides on whether the model itself is intrinsically interpretable or if methods are necessary to explain it after training and decision making in a post-hoc manner. Intrinsically interpretable machine learning models use simple structures and algorithms and thus are directly interpretable. Therefore, these models are also referred to as transparent or white-box models. [15] Typical examples of intrinsically interpretable machine learning models are linear regression or DT models. Post-hoc XAI methods are used to explain complex black box machine learning models which are not directly interpretable. These methods are applied to the machine learning models as separate sets of techniques after model training and decision making and provide their explanations without knowing the mechanism or the inner workings of the model. In comparison to intrinsically interpretable models the goal of post-hoc methods is to extract information from the trained model. Most of the recent works done on XAI belong to post-hoc methods.

Based on the scope of interpretability XAI methods are separated into whether the goal is to explain and understand the behavior of an entire model, globally, or just a single specific prediction, also referred to as an instance, locally.

Model specific XAI methods are limited to the application on specific machine learning models only. Model agnostic methods in contrast can be applied on any type of machine learning model.

Furthermore, XAI methods can be classified by their underlying techniques and results i.e., visualization, knowledge extraction, influence methods and example-based methods. The results of visualization methods are readable and comprehensible by visualization. The goal of knowledge extraction methods is to extract the knowledge learned by the black box model during training and to explain it in a comprehensible way. The approach proposed in this paper is combining the techniques of visualization and knowledge extraction.

Although *Explainable* is the key word in the term XAI it is rather used as a general umbrella term most notably in public settings. In scientific environments and especially in the machine learning community the term *interpretable* is more common. In this work the terms *interpretability* and *explainability* are used interchangeably.

2.5 Decision Trees

DTs are a non-parametric supervised machine learning method that is used for classification and regression problems. The goal of a DT is a model that makes its predictions by learning decision rules or conditions from the features of a data set. This goal is reached by a piece wise constant approximation, that splits the feature space into different subsets for multiple times according to specific cut off values inferred from the features. This leads to each instance of the data set belonging to one subset. In a visualized recursive binary DT, the final subsets are called leaf nodes. The intermediate subsets which are further split into a left branch for satisfying the splitting condition of that node or into a right branch, likewise, are called split nodes. There are various algorithms to create Decision Trees. One of the most popular is the Classification And Regression Trees (CART) algorithm. CART creates binary DTs with the features and cut off values yielding the highest gain of information at each node. [16–18]. The DTs in the proposed approach are trained for classification using the Gini index as measure of impurity. The goal is to find the split that makes the nodes maximally pure, meaning to minimize the number of different classes in a node by selecting the split that minimizes the impurity. The reason for choosing DTs in the proposed approach is the high interpretability since

DTs are intrinsically interpretable. The empirical study of [19] found DTs to be one of the two most interpretable machine learning models. Visualizations of binary DTs are often used for the highly interpretable representation of conditional logic. Furthermore, DTs enable the extraction of decision rules.

3. METHOD AND IMPLEMENTATION

3.1 Methodical procedure

The goal of the proposed IML approach is to reconcile advanced optimization of IESS with interpretability. In a post-hoc manner DTs, as intrinsically interpretable machine learning models, are used to represent and visualize the OS of the controller to be explained, enabling both global and local interpretability. Furthermore, being totally independent of the controller's model this approach is model agnostic, relying only on the model's output data and is based on five main steps (compare Fig. 1).

(1) Initially, optimization algorithms are deployed on a simulation model of an IESS for OS optimization. The used optimization framework is introduced in section 3.3.

(2) After training and testing of the optimized strategy, the input and output data of the controller to be explained is used to train DTs in a second step. The DTs in the proposed approach are built with the DecisionTreeClassifier package from the open source tool kit scikit-learn [20] for machine learning applications in the programming language Python. The output data contains (a) the states S_t of the environment for each timestep, also called *observation*, including all input features, and (b) the resulting *action* A_t, including the control signal for each element of the IESS as the environment. The aim of training the DTs on the observations and actions of the controller to be explained is to identify the rules of the optimized OS and to represent and visualize the controller's decision policy in binary DTs for higher interpretability. Depending on the controller's objective, included in the rewards R_t, the common method of evaluating and optimizing the trained DTs solely based on their accuracy is not sufficient as explained in detail in section 3.2.

(3) Therefore, in the third step the trained DTs must be implemented as a controller into the optimization framework and simulate the same scenario as the controller to be explained in step (1). This enables considering the controller's objective for evaluation as well.

(4) Step four is the optimization of the DTs and is represented by a loop between steps (2) and (3). The main optimization procedures and approaches contain training of several DTs on different subsets of data, depth limitation, i.e., limiting the longest path from a root to a leaf, and feature selection. The evaluation of the DTs is based on the output data of the simulated scenario, containing the observations, actions, and rewards of the implemented DT-controller.

(5) The loop is repeated until a DT is found that represents the optimized OS effectively and can potentially replace the controller to be explained. If the DT is not too complex and allows the extraction of rules, these can be used to adapt the conventional OS. In any of these two cases, represented by the final step, the decision process of the controller is interpretable at any time.

As this approach is model agnostic, it is applicable to any controller model. This paper demonstrates the application of the approach on a conventional and DRL-based controller (compare sections 4.3 and 4.4). As a white-box model the conventional controller allows for simple comparison of its rules with the identified rules of the DTs. Therefore, it serves the purpose of validating the proposed approach prior to its application on the DRL-

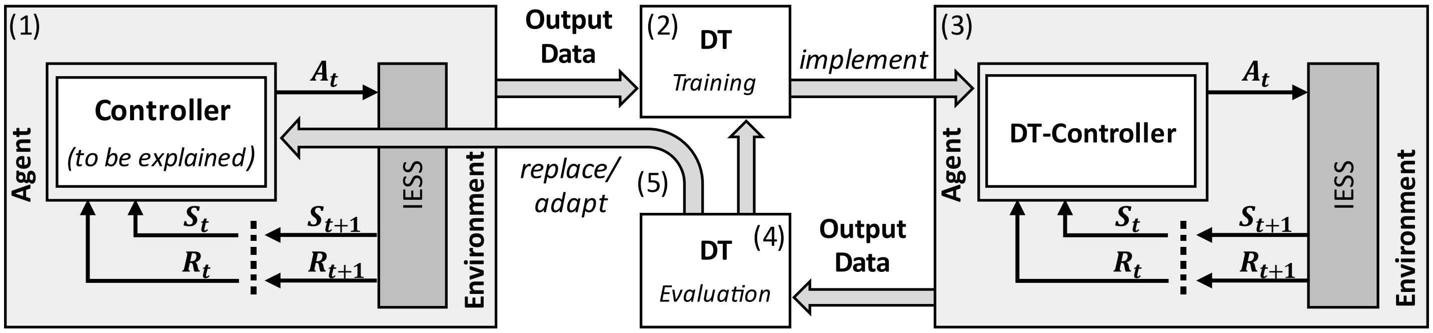

Fig. 1. Methodical procedure of the proposed IML approach

based controller and enhances the credibility of the obtained results.

3.2 DT optimization

The evaluation of DTs commonly relies on their accuracy in the context of training as well as for test data sets. In the specific scenario of the proposed approach, however, accuracy does not serve as an indicator of whether the underlying OS of the controller to be explained has been correctly identified, but rather as a measure of how effectively a trained DT can replicate the same results for a test data set. Given the need to identify the underlying OS of the controller under examination, an additional evaluation metric is essential. In this context, the rewards, representing the objective of the controller, are applied as an additional measure for performance evaluation and for DT optimization consequently. The main objective is to evaluate the DTs after implementation as a controller (compare step (4) in Fig. 1). The goal is to determine whether the controller's OS is correctly replicated over the same scenario, and whether the DTs achieve equivalent results with respect to the rewards, representing the controller's objective, in comparison to its underlying strategy. We interpret fulfilling these criteria as a correct identification and mapping of the underlying strategy. For the conventional and DRL-based controller the rewards represent costs in dimensionless units, being minimized in the controller's objective.

3.3 Framework

For the training and inference of the DRL-algorithms the energy optimization framework eta-utility [21] is utilized. It provides a standardized interface to research factory operations and uses the environment interface specified by the gym framework [22] and the agents provided by the Stable-Baselines3 [23] library, which provides state-of-the-art DRL-algorithms. Since eta-utility also enables the use of non-DRL algorithms, it is utilized for both the inference of the trained DTs and implementation of the conventional controller. To train the DTs, output data from the conventional controller or the trained DRL-algorithm is used, but the DT training itself is done outside of the framework (see step (2) in Fig. 1). The IESS simulations are imported as Functional Mock-Up Units (FMUs) [24]. The overall functional diagram is depicted in Fig. 2.

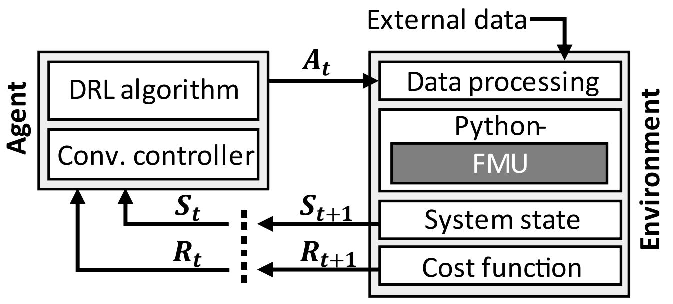

Fig. 2. Functional representation of the eta-utility framework and its components [6]

4. USE CASE AND EXPERIMENTS

4.1 *Use case*

The use case for the IML approach proposed in this paper is a state-of-the-art IESS for both heating and cooling. The heating grid consists of a Combined Heat and Power (CHP) unit and a Condensing Boiler (B) and is extended by an Immersion Heater (IH) and a Hot Water Storage (HS). The cooling supply consists of a Cooling Tower (CT), a Compression Chiller (CC) and a Cold Water Storage (CS). A Heat Pump (HP) links the heating and cooling grid. A schematic representation of the IESS is shown in Fig. *3*.

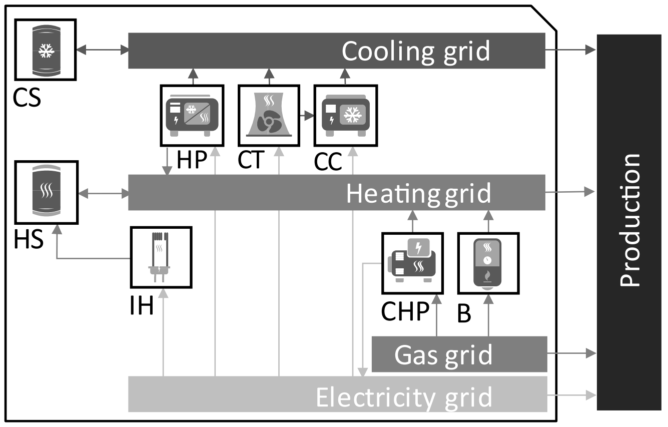

Fig. 3. Representation of the IEES chosen as the use case for the proposed IML approach (compare [5])

4.2 *Data preparation*

The DTs for the approach are generated with the sci-kit learn tool [16]. The DT are trained using the Gini index, with a deliberate exclusion of parameters that could introduce randomness into the algorithm. Parameters that might restrict the quantity of data points, nodes, or introduce feature weighting to the DT are similarly disregarded [25]. This approach is chosen to provide the highest level of unconstrained DT training, ensuring the independent and unconstrained detection of the conventional OS by the DT.

Initial experiments have shown the optimization effort of DTs through cost-complexity pruning (CCP) resulting in the inability of detecting switching operations within the OS. Consequently, CCP cannot be used as an optimization parameter leaving the variable *maximum depth* of the DTs as the only optimization parameter within the scikit-learn tool.

The selection of features chosen for DT training can also affect the outcome. Omitting certain features essential to the underlying operating system may hinder or even enhance accurate identification of OS. Based on this, the following white-box experiments exclusively apply the same 10 features that are used for the conventional controller. Similarly, in the black-box experiments, the same 22 features used for the DRL-based controller are utilized for DT training.

For all experiments mentioned hereafter, the training data is derived from the application of the conventional or DRL-based controllers on the same training dataset (*training year*), representing data of a whole year with a time resolution of 3 minutes per observation. The tests are done on a different test dataset (*test year*).

4.3 *White-box experiments*

For validation of the proposed approach in a first experiment, DTs are trained on the output data of the conventional controller. The parameters for the DT training are set as previously described with no constraints on maximum depth. In all following experiments, two different DT variants are considered:

a. DT_{total}: One single DT accounts for the entirety of the IESS by including all the control rules for all heating and cooling units.
b. $DT_{elements}$: Each unit of the IESS is represented by its own DT, providing a total of six DTs. For implementation as a controller, all six DTs are combined, allowing the simulation to be executed with various DTs acting as one controller.

Within this first series of tests, the underlying conventional operating system is compared to the two DT variants. While the conventional operating system incurs costs (equivalent to negative rewards) of 31,648 units for a full year of training, both DT variants each generate costs of around 32,000 units. When applied to a test year, identical costs can also be achieved. The decision paths of the developed DT correspond to the rules of the conventional OS.

The experiments on the white-box model reveal that the DTs can replicate the underlying OS in its application. This serves as a validation for the proposed IML approach.

4.4 *Black-box experiments*

Since the experiments with DTs for the conventional OS have shown promising results, the approach is analogously applied to the DRL-based controller as a black-box in the subsequent experiments.

4.4.1 DT_{total} and $DT_{elements}$ with depth limitation

The first black-box experiment is carried out on a complete training year analogously to the white-box tests. The DRL controller is then compared to the two DT variants as controllers that are trained on the same 22 features also provided to the DRL algorithm. Given the anticipated complexity of the DT, the depth of the DT is systematically adjusted and evaluated in this context.

While the DRL algorithm results in costs of 25,322 units throughout the training year, it is apparent upon closer examination that the costs of the DT are significantly higher and are also depending on the depth of the DT (compare Tab. *1*). We find that considering two different DT depths (depth 10 and fully grown) is sufficient as it becomes evident that for both variants the total costs decrease as the DTs grow larger. The two DT variants limited to a depth of 10 result in nearly tenfold increased costs compared to the DRL controller during the training year. The two fully grown DTs reduce this significant increase to six times the DRL costs. At this stage, however, it also remains unclear which variant (DT_{total} and $DT_{elements}$) generates better results.

Tab. 1. DRL controller and DRL output based DT controllers "DT_{total}" and "$DT_{elements}$" with different tree depths

	DRL	**DT_{total}**	**$DT_{elements}$**
Depth: 10	25,322	222,590	264,538
Fully grown		159,775	147,891

The fully grown DT_{total} has a depth of 39, while the $DT_{elements}$ range from 13 to 41. For the training year, the immersion heater has a depth of 0, indicating that it is not used in the training year.

Following this series of experiments, it is not yet possible to assert that the DT method is capable of identifying the DRL operational strategy. Trials on the test year are therefore not yet meaningful at this point. However, initial conclusions can be drawn that may be useful for future experiments. It is evident that the depth of the DT is highly relevant: larger trees seem to result in better performance. However, an increased depth of the DTs also leads to trade-offs in interpretability because there are more control rules to be extracted.

Since using a full year of training data does not seem to provide sufficient results, we are refining the training to smaller sections of the year, based on the assumption that fully grown DTs based on smaller data sections (e.g., separate months) will produce significantly better results.

4.4.2 DT_{total} and $DT_{elements}$ for individual months

The objective is for the DTs to capture a wide range of scenarios related to IESS use, including various factors such as off-seasons or prevailing weather conditions during a calendar year. However, as the previous experiment demonstrates, the method struggles replicating an entire training year for a complex DRL algorithm.

Therefore, the scope of this experiment is extended to individual months within the training year for a distinct evaluation of the numerous individual scenarios. Hence, output data of separate months serve as basis for the DT training, while the same parameters and inputs as in the previous experiments are used, with no restrictions on maximum depth.

The DRL controller generates costs of between 1,586 and 2,502 units per month. The $DT_{elements}$ lead to costs between 1,966 and 4,610 units, while the DT_{total} controllers generate costs of between 1,966 and 7,270 units (compare rows on training year in Tab. 2). When calculating cumulative sums over the individual months, the annual costs of the DRL controller are 25,158 units, whereas $DT_{elements}$ results in costs of 31,966, and DT_{total} amounts to 58,882. Throughout the year, different depths are observed for DT_{total} and $DT_{elements}$. While DTs are larger in the winter months (DT_{total} with a depth of 39 and $DT_{elements}$ between 8 and 34), their depth decreases in the summer (DT_{total} with a depth of 25 and $DT_{elements}$ between 0 and 20).

In six out of twelve months $DT_{elements}$ is successfully able to mimic completely identical costs from the DRL controller. In addition to the matching rewards, the control signal heat map (compare Fig. 4) of DRL and DT controllers displays identical patterns. Fig. 4 shows the heatmap of the DRL algorithm (top) and $DT_{elements}$ (bottom) for the month of March. The control signals of both controllers show identical trajectories.

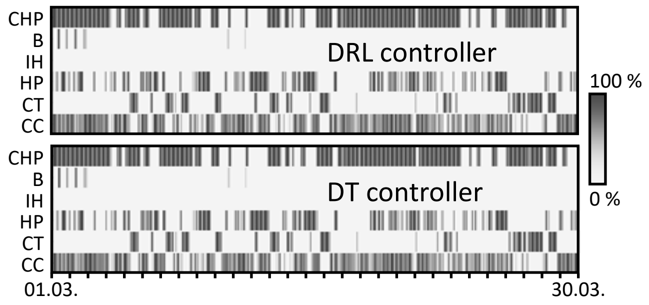

Fig. 4. Control signals during one test month with DRL controller and DT controller based on DRL output

This experiment demonstrates the success of the method when applied to the training year: The pattern of control signals and the resulting costs fully match those of the DRL-based controllers for individual months.

When the DTs are applied as controllers for the test year, however (compare rows *on test year* in Tab. 2), it becomes apparent that neither $DT_{elements}$ nor DT_{total} can approximate the costs of the DRL controller. In addition, the cumulative sums of the monthly costs show an increase in the annual costs compared to the application over a full year (compare Tab. *1* row *fully grown*).

Consequently, it cannot be assumed that the OS of the DRL controller is completely and accurately identified in the six months mentioned above. Moreover, the existing DT structures are very large, which significantly increases the complexity, compromising the interpretability of the DTs.

Tab. 2. Costs of $DT_{elements}$, DT_{total} and DRL controller for individual months on a training and test year

		Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
On Training Year	**DRL**	1,823	2,462	**2,455**	1,586	1,916	**2,385**	1,649	**2,612**	**1,966**	**1,884**	**1,918**	2,502
	$DT_{elements}$	2,863	2,012	**2,455**	2,649	4,547	**2,385**	2,065	**2,612**	**1,966**	**1,884**	**1,918**	4,610
	DT_{total}	7,270	3,776	5,651	3,826	2,908	3,404	9,748	2,982	1,966	4,202	6,606	6,542
On Test Year	**DRL**	8,377	2,557	1,761	1,869	2,068	2,430	2,080	2,400	1,828	1,542	2,188	2,029
	$DT_{elements}$	26,838	11,018	19,626	27,274	13,573	8,885	15,274	29,783	11,099	21,644	18,992	15,006
	DT_{total}	42,946	17,022	22,106	38,232	21,365	14,636	8,384	12,740	33,805	13,169	23,924	18,958

4.4.3 Feature selection

These results suggest that prior knowledge of the quantity and importance of input features could be beneficial for DRL output-based DT training and thus their transferability to unknown processes needs.

When adjusting the input features, we find that not only costs, but also the size of the DTs can be reduced, thereby enhancing interpretability. We thus infer that information concerning the importance of each feature within the entire IESS holds significant potential for the DT method and will investigate this effect in further work.

5. EVALUATION

We showed that DTs trained on the input and output data of the conventional OS (a white-box) in both the DT_{total} and $DT_{elements}$ variant, achieve the same behavior compared to the conventional control strategy when applied as a controller for both, training and test year.

Furthermore, the decision rules extracted from the DTs exactly correspond to the rules of the conventional OS. Moreover, visualization of the binary DTs makes the strategy and its rules highly interpretable, globally and locally. These results suggest that the presented approach is effective and that identifying operational strategies through output data-based training of DTs is feasible, at least when applied to relatively simple white-box models.

Regarding the DRL-based OS (a black-box), the results are more nuanced. When the DTs are applied as controllers on the test year, better performance (lower costs) can be observed with increasing depth of the DTs. Fully developed DTs without depth limitation achieve the lowest costs, but still do not match those of the DRL controller. When comparing DT_{total} and $DT_{elements}$ with the same depth limit, the results are inconclusive, implying that depth limitation alone is insufficient for a performance assessment. However, when fully grown DTs of both variants are considered, $DT_{elements}$ achieve better results.

Since splitting the data in terms of the IESS elements (one DT for each element), results in cost reduction and thus better approximation of the comparative value of the DRL strategy, we also split the training data in terms of its temporal dimension into individual months.

For DT_{total} this corresponds to an ensemble of 12 DTs in total, and 72 DTs for $DT_{elements}$. In the training scenario, $DT_{elements}$ match the results of the DRL controller precisely in 6 out of 12 months. Notably, this occurs during a continuous block of 4 months spanning from August to November. In February, $DT_{elements}$ even incurs lower costs than the DRL controller. The following five months see an inconsistent deviation, with July showing the lowest and December exhibiting the highest deviation. In each month, $DT_{elements}$ produce significantly better outcomes than DT_{total}. Except for September, where the same value is achieved as for DRL and $DT_{elements}$, and May, where DT_{total} is better than $DT_{elements}$ but still significantly worse than DRL. Applied to the test year, no outcome approaches the performance of the DRL algorithm, but $DT_{elements}$ outperforms DT_{total} in all months except July, August, and October.

We attribute these large deviations of the test year compared to the training year to overfitting. Despite this, the rules seem to be accurately reproduced by $DT_{elements}$ for at least 6 months and thus enable the interpretability of the DRL-based strategy. The variances observed across various months are indicative of seasonal dependencies, in which individual features have different influences. Early experiments (compare 4.4.3) indicate that modifying the feature selection significantly impacts the outcome of the DTs.

6. CONCLUSION AND OUTLOOK

This paper presents an interpretable machine learning approach for reinforcement learning based control strategy optimization of industrial energy supply systems. Therefore, DTs are trained on the input and output data of the DRL-based controller to find the rules of the optimized control strategy and to represent the algorithms' decision policy by visualizing it in binary DTs for higher interpretability.

Implementing the trained DTs as controllers for the same scenario as the DRL-algorithm enables evaluation of the identified strategy for further optimization. The approach was successfully demonstrated on the conventional control strategy of a state-of-the-art IESS consisting of a heating and cooling grid, with the trained DTs leading to the same results as the conventional OS for both train and test year. Applied on the DRL-based controller, neither a single DT trained on the whole dataset, nor separate DTs for each unit of the IESS were able to represent the control strategy in its entirety, although the latter achieved the highest similarities.

When training individual DTs for all IESS elements for individual months, the DTs were shown to replicate the performance of the DRL-based control strategy for 6 out of 12 months on the training year. When applied on the test year, however, the results did not approach those of the DRL algorithm.

In future research, further investigation will be conducted into the influence and the importance of individual input features since minor modifications to the feature selection have already shown significant impacts. An example of the practical applications of this could be the SHAP XAI method [26]. Furthermore, the presented approach should be extended to other DT methods, such as random forests or extreme gradient boosting, which are generally more data efficient yet still interpretable.

ACKNOWLEDGEMENT

The authors gratefully acknowledge financial support by the German Federal Ministry of Economic Affairs and Climate Action (BMWK) for the project EISKIG (grant agreement No. 03EN6012A) and project supervision by the project management Jülich (PtJ).

DECLARATION OF INTEREST STATEMENT

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper. All authors read and approved the final manuscript.

REFERENCE

[1] Panten N. Deep Reinforcement Learning zur Betriebsoptimierung hybrider industrieller Energienetze. Dissertation; 2019.

[2] Abele E, Schneider J, Beck M, Maier A. ETA – die Modell-Fabrik, Energieeffizienz weiter gedacht. Darmstadt; 2018.

[3] Sauer A, Abele E, Buhl HU. Energieflexibilität in der deutschen Industrie: Ergebnisse aus dem Kopernikus-Projekt - Synchronisierte und energieadaptive Produktionstechnik zur flexiblen Ausrichtung von Industrieprozessen auf eine fluktuierende Energieversorgung (SynErgie). Stuttgart: Fraunhofer Verlag; 2019.

[4] Bachmann A, Bank L, Bark C, Bauer D, Blöchl BB, Brugger M et al. Energieflexibel in die Zukunft - Wie Fabriken zum Gelingen der Energiewende beitragen können: VDI Verlag; 2021.

[5] Kohne T, Ranzau H, Panten N, Weigold M. Comparative study of algorithms for optimized control of industrial energy supply systems. Energy Inform 2020;3(S1):1–19.

[6] Weigold M, Ranzau H, Schaumann S, Kohne T, Panten N, Abele E. Method for the application of deep reinforcement learning for optimised control of industrial energy supply systems by the example of a central cooling system. CIRP Annals 2021;70(1):17–20.

[7] Adadi A, Berrada M. Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI). IEEE Access 2018;6:52138–60.

[8] Buoro D, Casisi M, Nardi A de, Pinamonti P, Reini M. Multicriteria optimization of a distributed energy supply system for an industrial area. Energy 2013;58:128–37.

[9] Panten N, Strobel N, Sossenheimer J, Abele E. Framework for an Energy Efficient and Flexible Automation Strategy and Control Optimization Approach of Supply Systems within a Thermally-Linked Factory. Procedia CIRP 2018;72:526–32.

[10] Atabay D. An open-source model for optimal design and operation of industrial energy systems. Energy 2017;121:803–21.

[11] Sutton RS, Barto A. Reinforcement learning: An introduction. Cambridge, Massachusetts, London, England: The MIT Press; 2018.

[12] Goodfellow I, Courville A, Bengio Y. Deep learning. Cambridge, Massachusetts: The MIT Press; 2016.

[13] DARPA. Broad Agency Announcement: Explainable Artificial Intelligence (XAI) 2016.

[14] Rudin C. Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead. Nature machine intelligence 2019;1(5):206–15.

[15] Lipton ZC. The Mythos of Model Interpretability: arXiv; 2016.

[16] scikit-learn. Decision Trees. [October 08, 2023]; Available from: https://scikit-learn.org/stable/modules/tree.html#id2.

[17] Molnar C. Interpretable Machine Learning: A Guide for Making Black Box Models Explainable. [October 08, 2023]; Available from: https://christophm.github.io/interpretable-ml-book/.

[18] Hastie T, Tibshirani R, Friedman J. The Elements of Statistical Learning. New York, NY: Springer New York; 2009.

[19] Huysmans J, Dejaeger K, Mues C, Vanthienen J, Baesens B. An empirical evaluation of the comprehensibility of decision table, tree and rule based predictive models. Decision Support Systems 2011;51(1):141–54.

[20] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O et al. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research 2011(12):2825–30.

[21] Grosch B, Ranzau H, Dietrich B, Kohne T, Fuhrländer-Völker D, Sossenheimer J et al. A

framework for researching energy optimization of factory operations. Energy Inform 2022;5(S1):1–13.
[22] Brockman G, Cheung V, Pettersson L, Schneider J, Schulman J, Tang J et al. OpenAI Gym; 2016.
[23] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, Noah Dormann. Stable-Baselines3: Reliable Reinforcement Learning Implementations. Journal of Machine Learning Research 2021;22(268):1–8.
[24] Modelica Association. Functional Mock-up Interface 2: FMI for Model Exchange and Co-Simulation; 2022.
[25] Müller AC, Guido S. Introduction to machine learning with Python: A guide for data scientists. Beijing, Boston, Farnham, Sebastopol, Tokyo: O'Reilly; 2017.
[26] Lundberg S, Lee S-I. A Unified Approach to Interpreting Model Predictions: arXiv; 2017.