Dynamic energy system modeling using hybrid physics-based and machine learning encoder–decoder models

Energy and AI 9 (2022) 100172

Contents lists available at ScienceDirect

Energy and AI

journal homepage: www.elsevier.com/locate/egyai

Dynamic energy system modeling using hybrid physics-based and machine learning encoder–decoder models

Derek Machalek [a], Jake Tuttle [b], Klas Andersson [a,c], Kody M. Powell [a,d,*]

[a] Department of Chemical Engineering, University of Utah, Salt Lake City, UT, United States of America
[b] Taber International, LLC, United States of America
[c] Department of Space, Earth, and Environment, University of Chalmers, Gothenburg, Sweden
[d] Department of Mechanical Engineering, University of Utah, Salt Lake City, UT, United States of America

ARTICLE INFO

Keywords:
Hybrid model
Encoder–decoder
Time series
Automatic differentiation
Thermal power plant

ABSTRACT

Three model configurations are presented for multi-step time series predictions of the heat absorbed by the water and steam in a thermal power plant. The models predict over horizons of 2, 4, and 6 steps into the future, where each step is a 5-minute increment. The evaluated models are a pure machine learning model, a novel hybrid machine learning and physics-based model, and the hybrid model with an incomplete dataset. The hybrid model deconstructs the machine learning into individual boiler heat absorption units: economizer, water wall, superheater, and reheater. Each configuration uses a gated recurrent unit (GRU) or a GRU-based encoder–decoder as the deep learning architecture. Mean squared error is used to evaluate the models compared to target values. The encoder–decoder architecture is over 11% more accurate than the GRU only models. The hybrid model with the incomplete dataset highlights the importance of the manipulated variables to the system. The hybrid model, compared to the pure machine learning model, is over 10% more accurate on average over 20 iterations of each model. Automatic differentiation is applied to the hybrid model to perform a local sensitivity analysis to identify the most impactful of the 72 manipulated variables on the heat absorbed in the boiler. The models and sensitivity analyses are used in a discussion about optimizing the thermal power plant.

1. Introduction

Integrating renewable energy onto the electrical grid is a critical step to reducing carbon emissions [1]. The fastest growing renewable energy sources, solar and wind, are also economically competitive with traditional electricity generation sources [2]. However, due to intermittency of solar and wind, they are not a direct substitute for dispatchable energy sources. As a result balancing authorities have economically incentivized some dispatchable thermal power plants (coal and natural gas) to transition from baseload to load following operations that ensure electricity generation meets demand [3].

Because thermal plants were traditionally designed to operate at baseload, transitioning to load following operations results in novel plant conditions and the need for new control strategies [4]. Instead of optimizing for steady-state conditions around the baseload, power plants now need to dynamically optimize operations as the facility ramps up and down as directed by the balancing authorities. Several papers have investigated how these new ramping conditions impacted NO_x emissions and how those emissions could be controlled [3,5]. In addition to emission controls, plants are interested in optimizing the heat rate [6]. A key step in optimizing changing processes is to first develop dynamic models that account for impacts on the process at the current time as well as multiple steps into the future.

For complex processes, with countless phenomena occurring simultaneously (e.g., a thermal power plant), an accurate physics-based dynamic model may be impossible to create. A rapid rise in the wealth of process data paired with machine learning (ML) techniques provides an alternative, empirical model development approach. *PyTorch*, *Jax*, *TensorFlow*, and many other libraries have made ML tools, namely automatic differentiation (backpropagation), accessible to a wide range of practitioners [7–9].

Empirical dynamic process models use available process measurements to predict values over multiple time steps. For empirical time series forecasting, deep learning is gaining favor over traditional techniques such as autoregressive integrated moving average (ARIMA) and support vector regression (SVR) [10–13]. Within deep learning, the preferred models are those that consider temporal relationships in the data using recurrent neural networks (RNNs), such as long short term memory units (LSTMs) and gated recurrent units (GRUs).

* Corresponding author at: Department of Chemical Engineering, University of Utah, Salt Lake City, UT, United States of America.
E-mail address: kody.powell@utah.edu (K.M. Powell).

https://doi.org/10.1016/j.egyai.2022.100172
Received 11 April 2022; Received in revised form 29 May 2022; Accepted 29 May 2022
Available online 6 June 2022

Nomenclature

Symbol

b	Biases
E	Energy
F	Forecast
h	Hidden state
$\dot{H}$	Enthalpy flow
n	New gate
$\dot{Q}$	Heat flow
r	Reset gate
S	State
T	Target
U	Manipulated variable
W	Weight
z	Update gate

Subscript

Ch	Chemical
EC	Economizer
El	Electrical
l	Lag
m	Total steps
n	Prediction steps
RH	Reheater
SH	Superheater
t	Time
Tot	Total
WW	Water wall

Abbreviation

AA	Auxiliary air
ARIMA	Autoregressive integrated moving average
ANN	Artificial neural network
CCOFA	Closely coupled overfire air
ED	Encoder–decoder
FA	Fuel air
GRU	Gated recurrent unit
LSTM	Long short term memory
ML	Machine learning
MSE	Mean square error
MV	Manipulated variable
RNN	Recurrent neural network
RMSE	Root mean square error
SOFA	Separated overfire air
SVM	Support vector machine
SVR	Support vector regression

GRU and LSTM models are now established tools for time series predictions. Rajagukguk et al. evaluated RNN, GRU, and LSTM models to predict solar irradiance and PV power [13]. They found that the deep learning models, RNN, GRU, and LSTM, outperformed SVMs and feed forward neural networks. Dubey et al. studied ARIMA, seasonal ARIMA, and LSTM models to forecast energy consumption to enhance smart grid operations [14]. The models considered weather features and past power consumption. The authors found that the LSTM model was the most accurate with a root mean square error of 0.23, compared to seasonal ARIMA with an RMSE of 0.55. The favorable accuracy of the LSTM model became more prominent with additional data and lag time steps. Wang et al. used an LSTM-based model for long-term energy consumption predictions [15]. In the one-step-ahead forecasting, the LSTM model had a lower prediction error compared to an ARIMA and a backpropagation neural network. Liu et al. used GRU to forecast Chinese energy consumption [16]. They compared the GRU model to support vector regression and multi-linear regression. The authors noted that the GRU model was the most accurate with an average absolute percentage error of 5.63. Tuttle et al. developed a time series forecasting model for combustion emissions of a coal fired boiler [4]. The authors compared ten data-driven dynamic modeling algorithms and found that GRUs were the best for predicting emissions over 60 steps. Cai et al. used forecasting techniques to predict building forecasts over 24 time steps in hour increments [17]. They compared deep learning models, GRUs and convolutional neural networks, with ARIMAs with exogenous inputs. They indicated that the ARIMA-based models generalized the best, but that the deep learning models were more accurate for individual cases with the convolutional neural network being the best decreasing error on average by 22.6%.

Newer time series models build on GRUs and LSTMs to construct sequence-to-sequence ML models, such as encoder–decoders (EDs). Du et al. used an LSTM-based encoder–decoder to predict air quality [18]. They used meteorological data and historical pollution values as inputs to the model. The LSTM-based ED outperformed all other ML models, including GRU and LSTM alone. Lu et al. built a GRU-based encoder–decoder to model combined heat and power heat loads [19]. The encoder in this case was an auto-encoder to extract the heat load. The result was about a 50% reduction in forecasting errors compared to a GRU only. Du et al. developed a multivariate attention-based encoder–decoder built on bi-directional LSTMs for five time series models [20]. The result of their analysis showed their model outperformed traditional approaches such as ARIMA and SVR as well as deep learning approaches like GRU and LSTM. Marino et al. compared the use of LSTM and LSTM-based encoder–decoder models for energy load forecasting of a building [21]. The authors demonstrated that the encoder–decoder model was accurate for both short term (1-minute) and long term (1-hour) forecasts, compared to the LSTM only model which was only accurate on 1-hour forecasts. Laubscher used an encoder–decoder model to predict tube temperature in a coal fired power plant [22]. The encoder and decoder were constructed with GRUs. An input sequence of length 8 min was used to predict an output sequence of 5-minutes. The result was a mean absolute percentage error below 1%. Raidoo et al. used an encoder–decoder mixture-density network to predict an air cooled condenser back pressure [23]. The encoder–decoder models were constructed as recurrent neural networks. The mixture-density network ED increased model accuracy from 68% to 82%. Rahman et al. used an encoder–decoder structure to predict electricity loads in buildings using an LSTM-based model [24]. They found that the encoder–decoder structure was more accurate than a multi-layer perceptron.

While newly developed machine learning tools and architectures continue to enable more accurate model development, physics-based or other statistical models do not need to be completely discarded. Willard et al. reviewed ML and physics-based hybridization and highlighted several places where physics-based information can be integrated into machine learning models, including loss functions, initialization, model architecture, and using residuals between the measured data and physical model predictions as the target for learning [25]. The authors are optimistic about the hybridization of ML and physics-based modeling and believe that the scope of the field is growing. Rangapuram et al. created hybrid deep learning and state space models that used deep learning to predict the parameters in a state space model [26]. The result was a hybrid model that outperformed matrix factorization, ARIMA, and DeepAR. Rajagukguk et al. evaluated a hybrid convolutional neural network (CNN) and LSTM model, which outperformed the individual deep learning models [13]. Kraft et al. developed a hybrid physics-based deep learning model used to predict global hydrological

modeling [27]. The model integrates constraints from the water cycle, such as water balances, and other domain specific information into an LSTM layer. The model reproduces measured water cycle variation in a manner consistent with scientific understanding. Machalek et al. created a hybrid model for a chemical reactor which used physics-based mass and energy balances and used ML models to learn reactor kinetics from empirical data [28]. The authors leveraged automatic differentiation to calculate the system Jacobian to enable the use of implicit numerical methods in evolving the system of ordinary differential equations. Smyl et al. used a hybrid LSTM and dynamic computation graph neural network for smoothing to create a hybrid model [29]. The model was the most accurate in forecasting over 100,000 time series in the M4 competition.

Smyl highlighted the use of automatic differentiation in starting to understand machine learning and deconstructing impacts of different black boxes. According to Baydin and Raissi, automatic differentiation is an underutilized tool in scientific computing, but it has seen a resurgence in the past few years [30,31].

In this paper, three time series prediction model configurations are developed with a GRU or GRU-based encoder–decoder architectures. Two configurations hybridize physics-based information from the case study into the ML model. The case study focuses on predicting heat rate in a thermal power plant by way of predicting the heat absorbed by steam and water in the boiler. The objective of the model development is to provide insight into how the unit operates and create a stepping stone to optimization.

The contribution of this paper is summarized below:

- The GRU-based encoder–decoder structure results in lower modeling error than the GRU only model for time series predictions on a thermal power plant in this paper. This is consistent with the findings of the literature review. The encoder–decoder structure is specifically constructed to separate learning the dynamics of the model (encoder) and predicting model output (decoder). The encoder–decoder structure is demonstrated to be on average 11% more accurate. That structure may benefit dynamic optimization by isolating optimization to the decoder.
- A novel hybrid physics-based encoder–decoder model is presented. Hybrid modeling leverages domain knowledge to extract intermediate model components for the deep learning models to focus on, instead of making predictions for the full system all at once. The result is a more accurate model as well as additional physical interpretability. The model hybridization increases accuracy by over 10% on average, without adding any additional data.
- Automatic differentiation is leveraged to perform local sensitivity analysis on the manipulated variables in the model to highlight important inputs. Paired with the hybrid model, automatic differentiation allows for insight on how different manipulated variables impact different intermediate components. The sensitivity analysis is discussed in light of providing a pathway for optimization.

The paper is organized as follows: Section 2 describes the background on the GRU and encoder–decoder models as well as automatic differentiation. Section 3 introduces the thermal power plant unit case study and shows how domain knowledge is integrated to create a hybrid model structure. Section 4 presents the models for comparison, including hyperparameters and evaluation metrics. Section 5 presents the modeling results and compares the different methodologies, before performing a local sensitivity analysis on the system to evaluate model inputs. Section 6 discusses the implications of the results and future work.

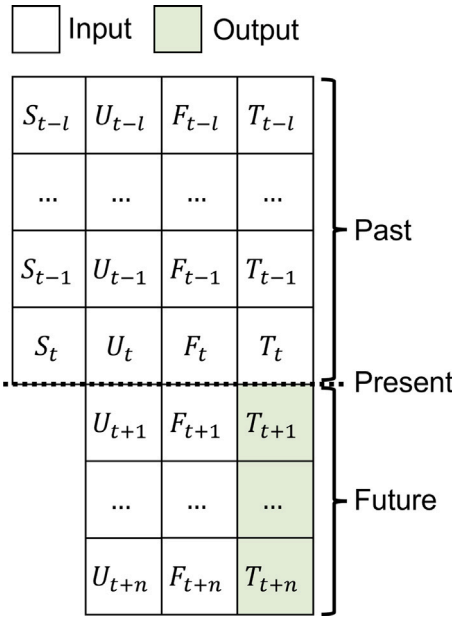

Fig. 1. Time series ML data frame.

2. Preliminaries of machine learning methods

In this section the structure of data and the associated deep learning models (GRU and GRU-based encoder–decoder) are described. Then the application of automatic differentiation for local sensitivity analysis is explained.

2.1. Data structure

A single instance of a data frame used to train the deep learning models is illustrated in Fig. 1. The columns of the data frame represent four categories of model variables: (1) S - states, values measured only to the present (e.g., temperatures, pressures), (2) U - manipulated variables, model inputs that can be manipulated into the future by operators or algorithms, (3) F - forecasts, model inputs that are known into the future, but cannot be manipulated (e.g., ambient temperature), (4) T - targets, values that are predicted. The subscript describes the time step t. The l subscript describes the model lag, how far back in time the model looks to make predictions, and the n subscript describes the number of forward steps the prediction makes.

2.2. Deep learning architecture

Fig. 2 represents the structure for an ML model made up of a GRU [32]. The model is initialized with a zero hidden state and moves through the GRU cell l times, where it is combined with information about the current state, X_t. X_t contains the current variables for the state, manipulated variables, forecasts, target value, as well as manipulated variables and forecasts n steps into the future. Once the hidden state is evolved to the current time step t, the output is turned into n target predictions using a linear transform. The p subscript indicates a target prediction.

Fig. 3 shows the mechanics happening within the GRU cell at each step.

In the GRU cell, first, the previous hidden state, h_{t-1} and the current state, x_t scale with their respective weight matrices, W_{hr} and W_{xr}, and biases b_{hr} and b_{xr}. The scaled values are passed through the non-linear sigmoid activation function, σ, to calculate r_t, the reset gate, in Eq. (1).

$$r_t = \sigma(W_{xr}x_t + b_{xr} + W_{hr}h_{t-1} + b_{hr}) \tag{1}$$

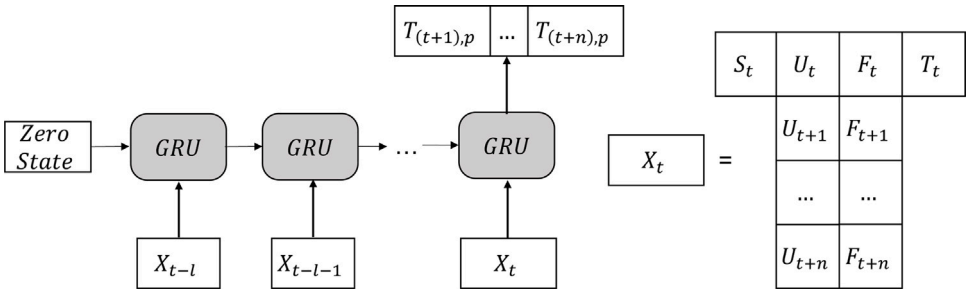

Fig. 2. GRU ML structure.

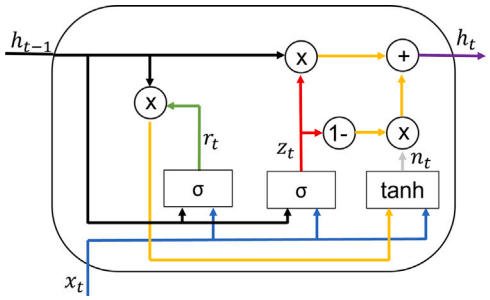

Fig. 3. Individual GRU cell.

Then the update gate, z_t, is calculated in a similar manner using weights, W_{hz} and W_{xz}, and biases b_{hz} and b_{xz} in Eq. (2).

$$z_t = \sigma(W_{xz}x_t + b_{xz} + W_{hz}h_{t-1} + b_{hz}) \tag{2}$$

The new gate, n_t, combines information from the h_{t-1}, r_t, and x_t with weights, W_{hn} and W_{xn}, and biases b_{hn} and b_{xn} in Eq. (3). Here hyperbolic tangent ($tanh$) is the non-linear activation function. The Hadamard product (elementwise multiplication) is represented as *.

$$n_t = tanh(W_{xn}x_t + b_{xn} + r_t * (W_{hn}h_{t-1} + b_{hn})) \tag{3}$$

Finally, the new gate and update gate are combined to calculate the new hidden state h_t, in Eq. (4).

$$h_t = (1 - z_t) * n_t + z_t * h_{t-1} \tag{4}$$

During model fitting, the weights, W, and biases, b, are updated to minimize the model and data mismatch. Each GRU cell contains the same W and b values. Fig. 2 displays the unrolled recurrent network with a hidden state passed between steps.

Fig. 4 illustrates the encoder–decoder model used for time series predictions. The encoder–decoder uses the same GRU cells from Fig. 3, but breaks the problem into two portions.

The encoder captures the system dynamics from the past to the present moment into a hidden or context state. The structure of the hidden state is dictated by the number of layers and nodes in the GRU. If the GRU has the 2 layers of 10 nodes, the number of hidden states is 20.

The encoder uses inputs, E_t, only from the current time of the model progression for S, U, F, and T. Then the decoder uses the hidden state passed from the final step of the encoder with future information available to the model D_{t+1} (U_{t+1} and F_{t+1}), to estimate future target values. The target values from the decoder at each time step are used as input to the next step. To clarify, the encoder and decoder have different weights and biases, but keep the same hidden state structure so that it can be passed from encoder to decoder.

The encoder–decoder explicitly deconstructs the data frame from Fig. 1 by time step, which may enhance model accuracy. If the model were to be used in optimization, optimizing decoder output with respect to future manipulated variables only requires the hidden state from the encoder to be calculated one time.

2.3. Sensitivity analysis

Reverse-mode automatic differentiation is the generalization of backpropagation used in the fitting of ML models, and allows for efficient access to the partial derivatives of differential computer programs compared to manual or symbolic differentiation [33]. The tool accumulates gradients of the elementary operations (e.g., addition, multiplication) and functions (e.g., sin, sigmoid) of the computer program as it moves backward from the output to the input to calculate derivatives. Fig. 5a shows the forward pass through a simple example of a computational graph and Fig. 5b shows the automatic differentiation steps used in the backward pass.

The computational graph represents the function shown in Eq. (5):

$$T = WUsin(U) \tag{5}$$

Symbolically the derivative of T with respect to U could be solved using the product rule of calculus in Eq. (6):

$$\frac{\partial T}{\partial U} = W(sin(U) + Ucos(U)) \tag{6}$$

For a simple example like this, symbolic derivatives are feasible, however for complex computation graphs (e.g., deep neural networks), maintaining symbolic expressions is not tractable.

Numeric differentiation, which investigates small changes in T with respect to small changes of U, is another option as shown in Eq. (7):

$$\frac{\partial T}{\partial U} \approx \frac{\Delta T}{\Delta U} \tag{7}$$

However, this approach does not consider intermediate values, such as a and b, and their respective derivatives which can be reused when there are multiple inputs, which is often the case in deep learning.

Automatic differentiation breaks the problem into the fundamental operations using the chain rule to recover the derivative across the computation graph, as shown in Eq. (8):

$$\frac{\partial T}{\partial U} = \frac{\partial T}{\partial a}\frac{\partial a}{\partial U} + \frac{\partial T}{\partial b}\frac{\partial b}{\partial U} \tag{8}$$

Each partial derivative on the right hand side is the derivative of an elementary operation in Fig. 5. Those derivatives can be substituted in Eq. (8), to yield Eq. (9):

$$\frac{\partial T}{\partial U} = Wb + acos(U) \tag{9}$$

The intermediate values, a and b, can be replaced with their values from the forward pass in Fig. 5 to recover the final form in Eq. (10):

$$\frac{\partial T}{\partial U} = Wsin(U) + WUcos(U) \tag{10}$$

Reverse-mode automatic differentiation scales efficiently with the large numbers of inputs, as is often the case in deep learning [34]. *PyTorch* and *TensorFlow* make reverse-mode automatic differentiation straightforward by managing the intermediate calculations and derivatives of elementary operations [35].

Automatic differentiation has many applications beyond backpropagation, including calculating the system Jacobians and Hessians, but

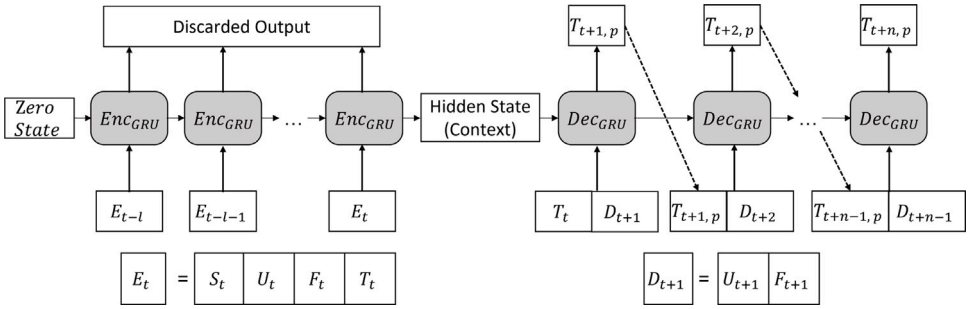

Fig. 4. Encoder–decoder ML structure.

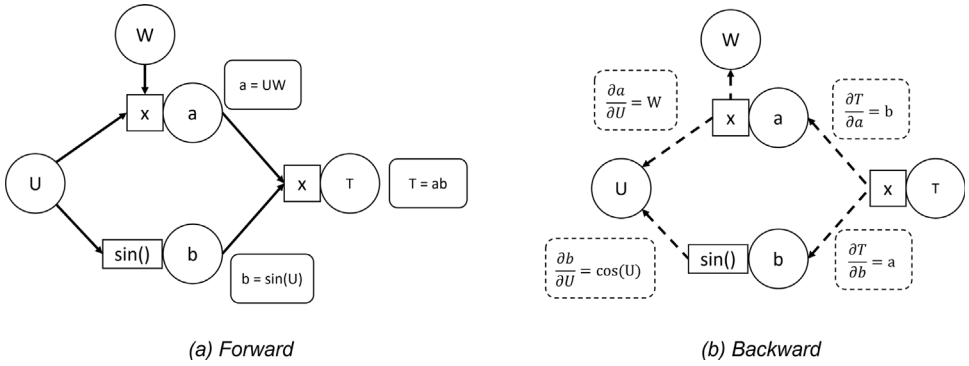

Fig. 5. Automatic differentiation example.

one simple, yet powerful, use of automatic differentiation is to evaluate the local sensitivity of model outputs to model inputs, including manipulated variables [36–38].

Because the deep learning models cover multiple time steps, the automatic differentiation can be applied to any outputs that a manipulated variable impacts. For instance, if the sensitivity of the output with respect to each manipulated variable or forecast, i, at the same time step, t, its desired the output would be:

$$\frac{\partial T_t}{\partial U_{t,i}}$$

and

$$\frac{\partial T_t}{\partial F_{t,i}}$$

The sensitivity analysis can also be applied over the sum of the next n time steps:

$$\frac{\partial \sum_{j=0}^{j=n-1} T_{t+j}}{\partial U_{t,i}}$$

and

$$\frac{\partial \sum_{j=0}^{j=n-1} T_{t+j}}{\partial F_{t,i}}$$

Automatic differentiation can be applied to physics-based models as well. There is nothing about deep learning models to uniquely qualify them for automatic differentiation. It is also worth noting that this sensitivity only applies around the inputs that passed through the neural network and that it is not a global sensitivity analysis.

3. Physical process, data, and hybrid model

In this section, the physical process of interest, thermal power generation is described. A simplified diagram of a coal fired boiler is illustrated in Fig. 6. After the physical process description, the available data set is presented. Finally, a hybrid approach, which integrates domain knowledge into the machine learning model, is described.

3.1. Process

In a coal-fired boiler, pulverized coal is injected along with sufficient air for combustion in the lower portion of the boiler. This lower boiler is surrounded by the water walls, which function to change the phase of the water from liquid to steam using the energy immediately released by combustion. High temperature combustion gases (flue gas) then exit the furnace along a designated path. Along this path, the high-temperature flue gas passes multiple tube sections designated as superheater, reheater, and economizer. Energy is transferred from the flue gas to these tube sections to ultimately provide superheated steam to the turbine for electricity generation.

The target for time series prediction is approximating the boiler heat rate, HR. Heat rate is an efficiency metric for boilers. The heat rate is defined in Eq. (11) as the energy, E, ratio of chemical energy, Ch, injected into the boiler from coal combustion to the electrical energy, El, produced by turbines from steam.

$$HR = \frac{E_{Ch}}{E_{El}} \tag{11}$$

Explicitly predicting the heat rate of a boiler is challenging because the coal combustion process is incredibly complex. Instead of focusing on coal combustion and energy released, the prediction model in this paper focuses on heat transferred from coal combustion into the water and steam loop which feeds the turbines and generates electricity. To reframe the problem of predicting heat rate, this model focuses on predicting the heat absorbed in the boiler, $\dot{Q}_{Tot}$, as a proxy for the electrical energy generated. The thermal heat produced is assumed to be a model disturbance and the coal mass feed rate is not a manipulated variable. That way the model can be used to inform how to maximize electricity generated without increasing the coal mass feed rate.

As shown in Fig. 6, heat is absorbed in four units of the boiler: the economizer (EC), the water walls (WW), the superheater (SH), and the reheater (RH). The total heat absorbed in the boiler can be calculated using an enthalpy flow, $\dot{H}$, balance around the entire boiler using Eq. (12).

$$\dot{Q}_{Tot} = \sum \dot{H}_{out} - \sum \dot{H}_{in} \tag{12}$$

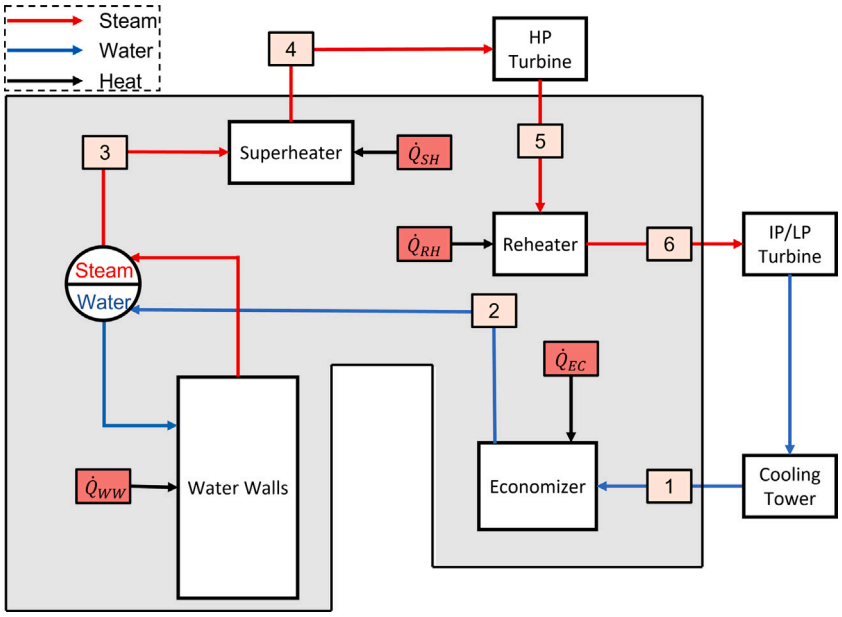

Fig. 6. Thermal power plant boiler steam loop.

The inlet streams to the boiler are 1 and 5 and the outlet streams are 4 and 6 in Fig. 6. The unit enthalpy of the water is determined using the pressure and temperature data of the streams in the *CoolProp* package with the Helmholtz equation of state [39]. The enthalpy flow is found by multiplying the unit enthalpy with the stream mass flows.

3.2. Data

The data for the model is categorized according to Fig. 1. Data is available over approximately 5-minute averages to reduce the noise in the sensor data. All data was collected from one unit at a single thermal power plant. The Python package *Pandas* is used to manage the data and resample data to be on exactly 5-minute increments [40]. Linear interpolation was used for variables with one or two missing data points in a row. The training, validation, and test data sets are selected from a large time series to avoid long periods with missing data points. As a result the training and validation consisted of 6400 data points total, which is split 80/20, respectively. The test set consisted of 500 data points—about 41.5 h of operation.

The state data for the system includes temperature, pressure, and flow rate data for streams 1–6, furnace temperature, excess oxygen measurements, total air flow rate, total coal feed rate, attemperator temperatures, attemperator pressures, and attemperator flow rates. The manipulated variables in the boiler are 16 separated overfire air (SOFA) dampers, 8 SOFA tilts, 4 closely coupled overfire air (CCOFA) dampers, 20 auxiliary air (AA) damper positions, 20 fuel air (FA) dampers, and 4 burner tilts—for a total of 72 manipulated variables. The forecast variables are the unit power generation set point and ambient air temperatures. The target is the total heat absorbed in the steam and water, $\dot{Q}_{Tot}$.

All data is normalized to protect the plant proprietary information.

3.3. Hybrid model

One option for predicting the heat absorbed by the water and steam in the boiler is to directly estimate $\dot{Q}_{Tot}$ from all of the available facility data using the ML structures described in Section 2. However, because this is a physical process there are underlying fundamentals that can be integrated into the ML model. The proposed hybrid for this model is to use energy balances within the boiler to isolate phenomena within each boiler component (economizer, water walls, superheater, and reheater),

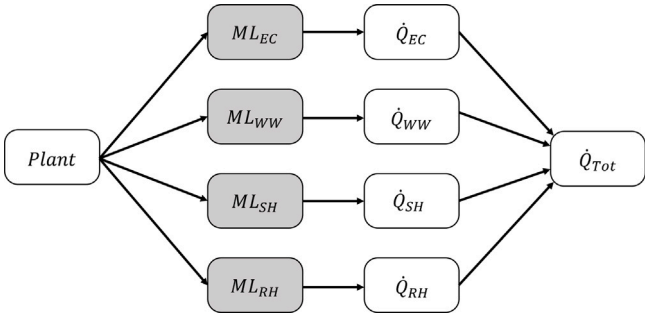

Fig. 7. Hybrid machine learning and physics-based model. In the hybrid model the individual heat absorption values ($\dot{Q}_{EC}$, $\dot{Q}_{WW}$, $\dot{Q}_{SH}$, $\dot{Q}_{RH}$) are learned separately and then summed together to estimate the total heat absorption.

using Eq. (12), to deconstruct the problem into intermediate targets. The sum of the energy absorbed in each component is the total energy absorbed in the boiler as shown in Eq. (13).

$$\dot{Q}_{Tot} = \dot{Q}_{EC} + \dot{Q}_{WW} + \dot{Q}_{SH} + \dot{Q}_{RH} \tag{13}$$

Two domain-specific assumptions are made for this hybrid model. The first is that the water and steam in the steam drum are at saturation conditions, which provides the necessary information to estimate enthalpy in stream 3. The second is that the energy accumulation within each component is not critical to estimate the total heat absorbed, because it is not accounted for in Eq. (12).

The proposed hybrid model is shown in Fig. 7. The ML model for each component is trained separately to learn the intermediate phenomena for the heat absorbed. Each individual component has access to all of the plant information. This allows the machine learning models to select the important plant information for the specific component. Because the individual components are not independent from each other, they may weight input information similarly. During testing, the outputs of each ML model are added together to estimate the total heat absorbed.

Two other hybrid structures were evaluated. One structure included $\dot{Q}_{Tot}$ in the loss function to connect the hybrid structure together. With noisy data, adding $\dot{Q}_{Tot}$ to the loss function may have amplified the noise. Another structure included passing the output of the ML

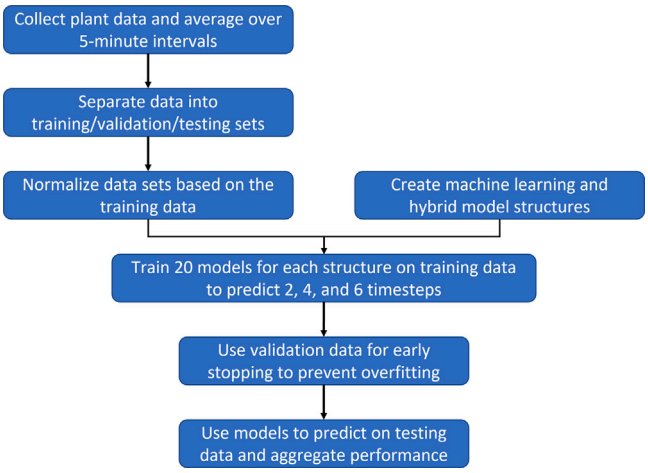

Fig. 8. Process flow diagram for machine learning data collection, model training, and prediction.

Table 1
Mean and minimum mean square error (MSE) for each model configuration. The model architecture is specified as a gated recurrent unit (GRU) or a GRU-based encoder–decoder (ED). The italics represent the lowest mean values and the bold represent the lowest minimum MSE values for each predicted step length.

			Predicted steps		
			2	4	6
Hybrid forecasts only	ED	Mean	0.031	0.109	0.178
		Min	0.025	0.086	0.154
	GRU	Mean	0.036	0.149	0.264
		Min	0.027	0.102	0.187
Full forecasts and MVs	ED	Mean	0.028	0.075	0.121
		Min	**0.020**	0.059	0.087
	GRU	Mean	0.031	0.090	0.151
		Min	0.024	0.056	0.114
Hybrid forecasts and MVs	ED	Mean	*0.024*	*0.063*	*0.109*
		Min	0.021	**0.052**	**0.083**
	GRU	Mean	0.026	0.071	0.128
		Min	0.021	0.056	0.108
Persistence	–	–	0.098	0.413	1.000

models into the subsequent model to parallel the steam water loop. The output of the economizer passed to the water wall, whose output passed to the superheater, whose output passed to the reheater. This model involved restructuring the model so that the information flow followed the physical structure of the plant. The proposed hybrid model in Fig. 7 had the lowest MSE, and was the only model carried through for the remainder of the analysis.

4. Model evaluation

Three different model configurations are evaluated in this paper. Each configuration is examined with a GRU and a GRU-based encoder–decoder as the deep learning architecture. The first configuration is the hybrid structure in Fig. 7, with the data from Section 3.2 where the individual components are learned separately ($\dot{Q}_{EC}$, $\dot{Q}_{WW}$, $\dot{Q}_{SH}$, $\dot{Q}_{RH}$). The second configuration is to learn the full model ($\dot{Q}_{Tot}$) at once, using the same data as the hybrid model. The full model includes the individual component heat gains as input data. This is done to focus on the model structure difference of the hybrid and full models as opposed to the input data difference. Finally, the third configuration uses the hybrid model structure without manipulated variables. The purpose is to evaluate how important the manipulated variables are as opposed to state variables, and especially the forecasting value of the unit power generation set point. Note that we also evaluated a model with only manipulated variables but the models performed poorly and were not considered for the remainder of the analysis.

A persistence model is used as a baseline for model comparison. The persistence model predicts values to be the same as the last known value and serves as a good lower bound on model accuracy.

Each configuration is evaluated for predicting multiple steps, n. Specifically, predictions are made over 2, 4, and 6 steps. Each step is considered in the loss function, not only the final step. The models are evaluated over the entire 500 time step, m, test set, using mean squared error (MSE) to calculate accuracy. The formula for MSE is shown in Eq. (14):

$$MSE = \frac{1}{nm}\sum_{j=1}^{m}\sum_{i=1}^{n}(y_{i,j} - \hat{y}_{i,j})^2 \tag{14}$$

where y is the measure value and $\hat{y}$ is the model predicted value. The models were coded in *PyTorch*. A validation set is used for early stopping to avoid overfitting. A patience parameter of 10 was used. The patience parameter is the number of training iterations to wait without seeing prediction improvement on the validation set before early stopping is triggered. Each model fitting is run 20 times to find the distribution of model errors that were produced.

The hyperparameters for each model were the same, although different values were tested for each to find the best fit. Values in parentheses indicate alternative hyperparameters that were investigated. Each deep learning model used 2 layers (1 layer) with 10 nodes (20, 40 nodes). The learning rate is 0.1 (0.01) with optimizer LBFGS (Adam). The models were initialized using the default *PyTorch* weight and bias initialization. The best lags for the models were 3, 4, or 5 (15–25 min), however, 4 lag steps were used for consistency.

Automatic differentiation was applied to the hybrid encoder–decoder to perform a local sensitivity analysis. Two sensitivity analyses are of interest. The first is to look at how the inputs to the decoder (manipulated variables and forecasts) impact the heat absorbed at the same time step. The second is to evaluate the impact of the manipulated variables on the next four time steps. The sensitivities are investigated for the model as a whole as well as each individual component.

A summary of the data collection and machine learning approach is shown in Fig. 8.

5. Results

The results section is separated into a modeling section that discusses model fitting and accuracy, and a model sensitivity analysis that discusses the outputs of the automatic differentiation.

5.1. Model comparisons

Table 1 shows a normalized mean and minimum MSE for each model configuration over 20 model fittings. The values are normalized to make the 6-step prediction error for the persistence model unit value. The lowest individual errors (Min) for each predicted step length are bolded and lowest averages (Mean) are italicized. Fig. 9 shows the distributions for the model fittings for each configuration for the 4-step prediction models. The whiskers represent 1.5 times the interquartile range or the further point within that range and outliers are shown as individual points. Several points stand out from the data. First, all deep learning models outperform the persistence model. Second, comparing the hybrid models with and without manipulated variables indicates the importance of manipulated variables to the model. While the power generation set point of the model is crucial for model fitting it is not sufficient to capture the model dynamics in the coal fired boiler. Third, the encoder–decoder (ED) structure is preferred over the GRU model in each case. For 2, 4, and 6 steps into the future, the

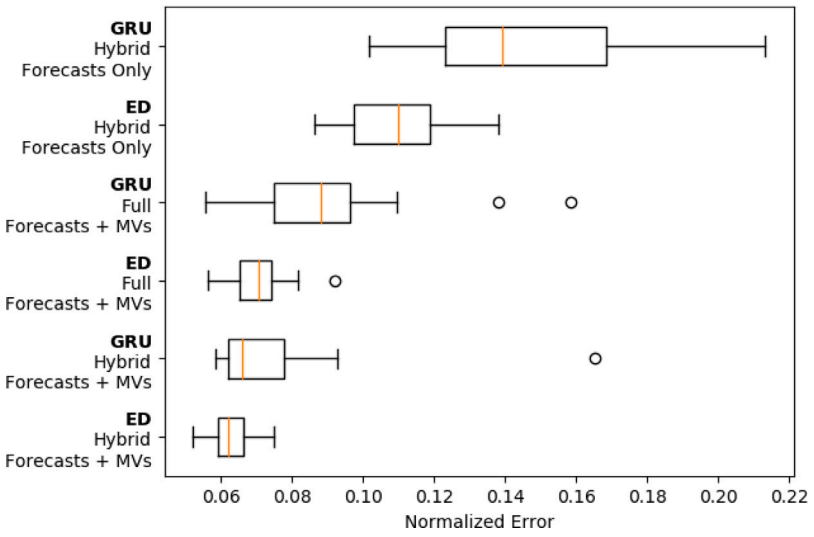

Fig. 9. Box and whisker plot for 4 prediction steps of each ML model. The boxes represent the interquartile range around the mean (orange line) and the whiskers represent the 5 to 95 distribution percentiles. Outliers are represented as circles. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

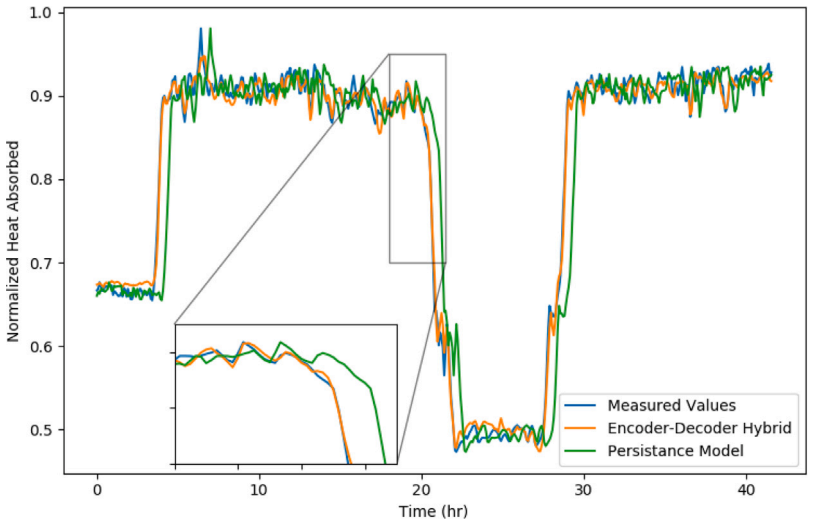

Fig. 10. Time series model prediction. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

encoder–decoder on average outperforms the GRU model by 11%, 21%, and 25%, respectively. Finally the hybrid encoder–decoder model is on average more accurate than the full encoder–decoder model for 2, 4, and 6 steps the improvement is 13%, 16%, and 10%, respectively. For the minimum values the full model is more accurate for 2 steps, but is overtaken by the hybrid model 4 and 6 steps into the future.

An example prediction for the model is shown in Fig. 10. The figure shows the outputs of the encoder–decoder hybrid model with all data inputs compared to the persistence model and measured values for the test set 4 prediction steps into the future. In the figure, the persistence model does fine when the unit operates at a steady set point, and therefore steady heat absorption rate. However, during transitions from a high heat absorption rate to a low heat absorption rate, the persistence model fails (this is highlighted in the plot inset). During the transition the deep learning encoder–decoder leverages information about model dynamics and future information to more closely follow the actual unit.

The mean errors at each step of the 6-step predictions (30 min) for the 20 iterations of each model are shown in Fig. 11. In addition to plotting the described model configurations, a steady-state model is added. The steady-state model only uses forecasts and manipulated information at each future time step through a feed forward neural network to predict the heat adsorbed. The purpose is to show how much the recurrent neural networks are relying on future information as opposed to the dynamics in the hidden state. For the mean trends in Fig. 11, as the most accurate model configuration predictions reach 6 steps they approach the steady-state value. This indicates that often the dynamics stored in the hidden state of the model are largely lost by step 6.

5.2. Sensitivity analysis

Fig. 12 shows the local sensitivity of the output of the encoder–decoder model (heat absorbed) at the current time step to the 72 manipulated variables and 2 forecast variables at the same time step. This is the same time frame shown in Fig. 10. The figure shows the changes in the sensitivity of model inputs over time and under different conditions. For example, the teal and orange lines at the bottom of the plot swap in magnitude depending on the operation of the unit. When the heat absorbed is high, orange is more negative, and visa versa. The figure also shows the stratification of model input importance

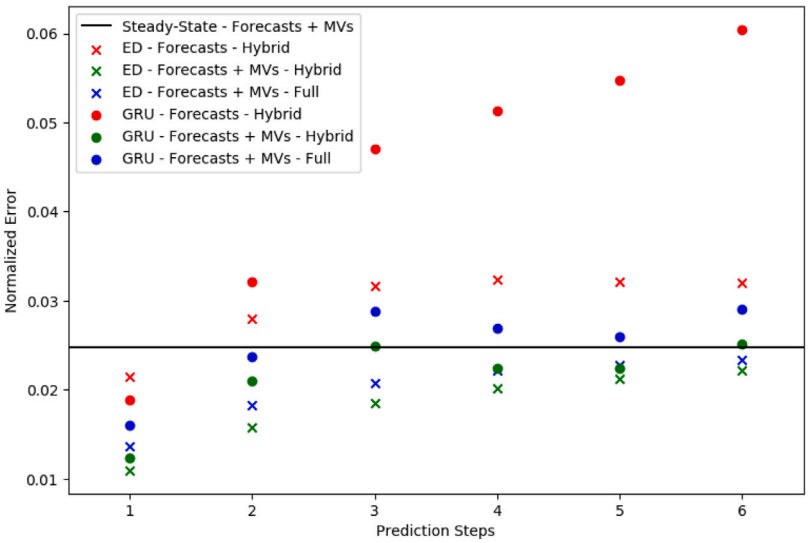

Fig. 11. Mean normalized error for each individual prediction step.

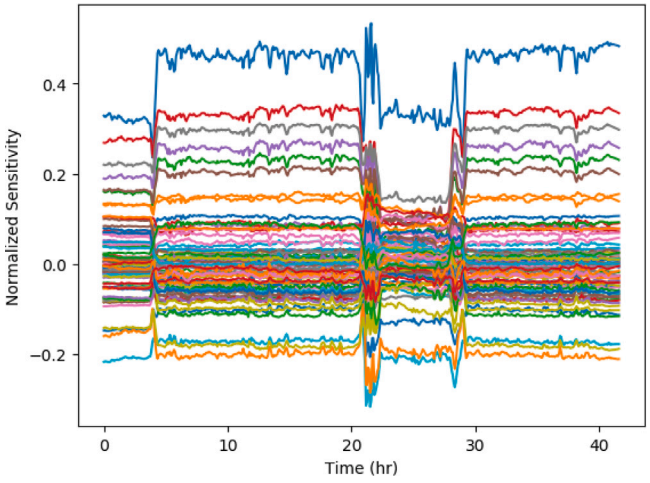

Fig. 12. Sensitivity of the model output to input at the same time step. The lines represent the 72 manipulated and 2 forecast variables and the normalized sensitivity of the most important variables is summarized in Tables 2 and 3. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

(magnitude). A handful of variables have an outsized positive and negative impact on the model, while most variables hover around 0.

Table 2 averages the sensitivities shown in Fig. 12 for the overall trend, as well as for the sensitivities of the individual boiler components. The facility tags for each model input are shown. SOFA, CCOFA, AA, and FA correspond to the definitions from Section 3.2 and the numbers and letters that follow indicate positioning within the boiler. Unsurprisingly, the unit set point (MW Start Remote) is the most important model input, but some manipulated variables also play an important role. Namely, certain SOFA dampers correspond to a positive sensitivity and CCOFA dampers correspond to a negative sensitivity while the FA and AA dampers are in between. Burner tilts appear to be less important.

The economizer is not as impacted by MW Start Remote as the overall system and depends more on SOFA values. Meanwhile, the water walls most closely follow the overall sensitivity to the heat absorbed. This makes sense because the water walls absorb the most heat in the boiler and are most directly impacted by combustion, which the manipulated variables directly impact. This highlights that manipulated variables impact components differently, which means one component can be targeted for heat absorption. For example, the heat absorption may need to be increased in the water walls, but not increased too much in the superheater to avoid excessively hot tubes.

Table 3 shows the sensitivity of the model outputs over the next four time steps (20 min) to the inputs at the current time step. For the thermal power plant, a 20 min horizon is large enough to plan for the large power generation swings the plant must undertake. Compared to Table 2, this table shows that the auxiliary air (AA) dampers have a larger magnitude in the sensitivity analysis overall and for each component for the longer time horizon.

The varied sensitivity of different model variables in many ways reflects understood domain expertise regarding the process. Many of these sensitivity values can be understood by recognizing the effect changes in these variables have on the physical position of the combustion process as well as combustion efficiency and performance. For instance, SOFA dampers have the effect of stretching the combustion process, making it occur over a larger area. This results in more combustion occurring higher in the furnace, which assists in explaining the dominant positive effect of many SOFA dampers relative to the economizer heat absorption. Similarly, due to the method of providing air to each damper, increasing CCOFA damper position "robs" air from the SOFA dampers, having a negative impact on this relationship.

Concerning air introduction directly at the point of fuel injection, it is common for there to be imbalances in fuel feed rate from injection port to injection port. These disturbances are unmeasured, but often advanced modeling systems can recognize and attempt to compensate for such imbalances. It is not uncommon for individual AA or FA dampers to have varied effects on the combustion process, and this is a likely cause for the apparent range of sensitivities assigned to these parameters.

The local sensitivity does not provide the entire picture for the model because it does not consider correlations or non-linearities [41]. Nonetheless, automatic differentiation is the most effective method for fitting non-linear deep neural networks. The tool is at least worth considering for investigating the impacts of manipulated variables on the output of the developed deep learning models, which are already in the form of differentiable models.

Automatic differentiation can only be applied to manipulated variable trajectories that already passed through the model and the resultant sensitivities are only accurate locally. Therefore the tool is most effective given an initial trajectory, possibly from facility operators or another algorithm. Within bounds around the initial trajectory, the automatic differentiation could be integrated with a gradient descent

Table 2
10 most positive and most negative sensitivities of model outputs to inputs at the same time step.

Sign	Overall		Economizer		Water wall		Superheater		Reheater	
	Input	Sens.	Input	Sens.	Input	Sens.	Input	Sens.	Input	Sens.
+	MW Start Remote	0.34	SOFA 4 RR	0.07	MW Start Remote	0.14	SOFA 4 RR	0.14	MW Start Remote	0.13
	SOFA 4 LR	0.24	SOFA 4 LF	0.07	SOFA 4 LR	0.12	MW Start Remote	0.13	SOFA 1 RF	0.10
	SOFA 4 RR	0.21	SOFA 3 RF	0.06	SOFA 3 RR	0.11	AA 34 LR	0.13	FA 1 LR	0.08
	SOFA 3 RR	0.21	SOFA 4 LR	0.06	SOFA 4 RR	0.07	AA 23 LF	0.10	SOFA 3 LF	0.07
	SOFA 4 LF	0.19	SOFA 3 LF	0.05	SOFA 4 LF	0.07	SOFA 4 LF	0.10	AA 23 LR	0.06
	AA 23 LF	0.15	SOFA 3 LR	0.03	SOFA 4 RF	0.07	SOFA 4 LR	0.10	SOFA 3 LR	0.05
	AA 34 LR	0.15	SOFA 1 LR	0.03	FA 1 RR	0.07	SOFA 3 RR	0.08	AA 45 RF	0.05
	SOFA 4 RF	0.14	AA 56 LR	0.03	AA 34 LR	0.07	AA 12 RR	0.05	AA 34 RR	0.05
	SOFA 3 LF	0.13	MW Start Remote	0.03	SOFA 3 LF	0.05	AA 12 RF	0.05	SOFA 2 RR Tilt	0.05
	AA 34 RR	0.08	FA 2 LF	0.03	SOFA 2 RR	0.05	SOFA 4 RF	0.05	AA 56 LR	0.04
−	FA 3 RF	−0.08	AA 34 RR	−0.03	FA 4 LF	−0.04	FA 5 RF	−0.04	FA 4 RF	−0.03
	FA 4 LF	−0.08	FA 4 LF	−0.03	SOFA 2 RF	−0.04	FA 1 LR	−0.04	AA 12 LF	−0.03
	AA 34 LF	−0.08	FA 2 RR	−0.03	FA 5 LR	−0.04	FA 5 LF	−0.04	CCOFA 1 RR	−0.04
	AA 45 LR	−0.09	FA 3 LR	−0.03	AA 45 LR	−0.04	FA 3 RF	−0.04	AA 45 LF	−0.05
	CCOFA 1 RF	−0.09	CCOFA 1 LF	−0.04	FA 3 RF	−0.05	AA 12 LF	−0.07	SOFA 1 RF Tilt	−0.05
	AA 12 LF	−0.09	AA 23 RR	−0.04	CCOFA 1 LF	−0.05	CCOFA 1 LR	−0.08	AA 34 LR	−0.05
	AA 23 RF	−0.17	CCOFA 1 LR	−0.04	FA 1 LR	−0.06	AA 34 LF	−0.08	CCOFA 1 RF	−0.05
	AA 45 RR	−0.18	CCOFA 1 RF	−0.05	AA 34 RF	−0.06	CCOFA 1 RR	−0.09	FA 2 RR	−0.06
	CCOFA 1 LR	−0.18	CCOFA 1 RR	−0.05	CCOFA 1 RR	−0.07	AA 45 RR	−0.12	AA 45 LR	−0.09
	CCOFA 1 RR	−0.19	AA 23 LR	−0.07	CCOFA 1 LR	−0.08	AA 23 RF	−0.15	AA 45 RR	−0.10

Table 3
10 most positive and most negative sensitivities of the next four model outputs to inputs at the current time step.

Sign	Overall		Economizer		Water wall		Superheater		Reheater	
	Input	Sens.	Input	Sens.	Input	Sens.	Input	Sens.	Input	Sens.
+	MW Start Remote	0.60	SOFA 4 LF	0.12	MW Start Remote	0.20	MW Start Remote	0.29	SOFA 1 RF	0.21
	SOFA 4 LR	0.28	SOFA 4 RR	0.11	SOFA 3 RR	0.11	AA 34 LR	0.23	MW Start Remote	0.20
	SOFA 4 RR	0.26	SOFA 4 LR	0.11	SOFA 4 LR	0.10	SOFA 4 RR	0.19	SOFA 3 LF	0.14
	AA 34 LR	0.24	SOFA 3 RF	0.09	SOFA 4 RF	0.08	AA 23 LF	0.13	AA 45 RF	0.11
	SOFA 3 RR	0.24	MW Start Remote	0.08	AA 34 LR	0.08	SOFA 4 LR	0.11	SOFA 3 LR	0.10
	SOFA 4 LF	0.24	AA 56 LR	0.07	FA 1 RR	0.07	AA 12 RF	0.10	SOFA 2 RR Tilt	0.10
	AA 23 LF	0.21	SOFA 3 LF	0.06	SOFA 4 LF	0.07	SOFA 3 RR	0.10	AA 56 LR	0.10
	SOFA 3 LF	0.19	SOFA 3 LR	0.05	SOFA 4 RR	0.07	SOFA 4 LF	0.09	SOFA 4 LR	0.09
	AA 34 RR	0.17	AA 23 RF	0.04	AA 23 LF	0.07	CCOFA 1 LF	0.09	Burner RF Tilt	0.09
	SOFA 4 RF	0.16	FA 2 LF	0.04	FA 2 RR	0.06	AA 12 RR	0.09	SOFA 4 RF	0.09
−	FA 5 RF	−0.08	FA 1 LF	−0.04	AA 45 LR	−0.04	FA 4 LR	−0.06	FA 4 RF	−0.06
	AA 56 LF	−0.09	FA 5 LR	−0.05	AA 12 RF	−0.04	CCOFA 1 LR	−0.07	CCOFA 1 LR	−0.07
	AA 45 LR	−0.10	FA 4 LR	−0.05	SOFA 3 RF	−0.04	FA 5 LF	−0.08	FA 1 RR	−0.07
	AA 34 LF	−0.10	AA 34 RR	−0.05	CCOFA 1 LF	−0.05	AA 56 LF	−0.09	FA 4 LF	−0.07
	FA 4 LF	−0.13	CCOFA 1 LF	−0.07	FA 5 LR	−0.05	AA 34 LF	−0.11	CCOFA 1 RF	−0.08
	AA 12 LF	−0.13	CCOFA 1 RF	−0.08	FA 3 RF	−0.06	AA 12 LF	−0.12	SOFA 1 RF Tilt	−0.08
	CCOFA 1 LR	−0.20	AA 23 RR	−0.08	CCOFA 1 RR	−0.07	SOFA 1 LR	−0.12	CCOFA 1 RR	−0.10
	AA 23 RF	−0.25	CCOFA 1 LR	−0.09	FA 1 LR	−0.07	CCOFA 1 RR	−0.14	FA 2 RR	−0.13
	CCOFA 1 RR	−0.28	CCOFA 1 RR	−0.10	CCOFA 1 LR	−0.07	AA 45 RR	−0.19	AA 45 RR	−0.15
	AA 45 RR	−0.30	AA 23 LR	−0.12	AA 34 RF	−0.09	AA 23 RF	−0.25	AA 45 LR	−0.15

algorithm to slightly modify the manipulated variables to optimize the output of the hybrid model. Notably, the optimization would need to be recalculated at every time step from a new initial trajectory.

6. Conclusion

In this paper, three model configurations with two different deep learning models at the core, GRU and GRU-based encoder–decoder, are presented. The models perform multi-step time series predictions (steps of 2, 4, and 6). The time series predictions are made on the heat absorbed by water and steam in a boiler. The GRU-based encoder–decoder deep learning structure is more accurate than the GRU alone by over 11% on average for 20 iterations of each model. The novel hybrid physics-based model constructed with the encoder–decoder is the most accurate model overall. That configuration is over 10% more accurate than the full encoder–decoder model on average over 20 iterations. Automatic differentiation is used to perform a local sensitivity analysis of the hybrid encoder–decoder model. The output shows which manipulated variables in the thermal power plant are most important around the model input trajectory. Further, the hybridization allows for the sensitivity analysis to be applied to individual components (economizer, water wall, superheater, reheater) in the unit to evaluate how manipulated variables impact the heat absorbed in each component. In the future, the gradients of heat absorbed with respect to manipulated variables could be used to modify unit operations to maximize the heat absorbed by steam and water. The caveat is that the sensitivity analyses need to be constructed around an initial input trajectory. If the encoder–decoder architecture is used in an optimization routine, the encoder hidden state only needs to be calculated one time because all future manipulated variables are only inputs into the decoder. Future work will also expand both the available data set evaluated as well as the hyperparameter selection to enhance the models. Future work will also expand the types of models investigated including such as ARIMA and Deep autoregression to evaluate more tools for this complex forecasting problem.

CRediT authorship contribution statement

Derek Machalek: Methodology, Writing – original draft, Software, Formal analysis, Visualization, Conceptualization, Investigation. **Jake Tuttle:** Writing – review & editing, Data curation, Validation. **Klas**

Andersson: Writing – review & editing. **Kody M. Powell:** Funding acquisition, Project administration, Supervision, Writing – review & editing, Resources, Conceptualization.

Declaration of competing interest

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.

Acknowledgments

This research is funded by the United States Department of Energy project DE-FE0031754. The funding sources did not contribute to study design or the writing of this report.

References

[1] Energy Information Administration. How much carbon dioxide is produced per kilowatthour of U.S. electricity generation?. 2021, https://www.eia.gov/tools/faqs/faq.php?id=74&t=11, Online; accessed 18 November 2020.
[2] Energy Information Administration. Levelized costs of new generation resources in the annual energy outlook 2021. 2021, https://www.eia.gov/outlooks/aeo/pdf/electricity_generation.pdf, Online; accessed 18 November 2020.
[3] Tuttle Jacob F, Powell Kody M. Analysis of a thermal generator's participation in the western energy imbalance market and the resulting effects on overall performance and emissions. Electr J 2019;32(5):38–46.
[4] Tuttle Jacob F, Vesel Ricky, Alagarsamy Siva, Blackburn Landen D, Powell Kody. Sustainable NOx emission reduction at a coal-fired power station through the use of online neural network modeling and particle swarm optimization. Control Eng Pract 2019;93:104167.
[5] Safdarnejad Seyed Mostafa, Tuttle Jake F, Powell Kody M. Dynamic modeling and optimization of a coal-fired utility boiler to forecast and minimize NOx and CO emissions simultaneously. Comput Chem Eng 2019;124:62–79.
[6] Dong Yuliang, Jiang Xu, Liang Zhihong, Yuan Jiahai. Coal power flexibility, energy efficiency and pollutant emissions implications in China: A plant-level analysis based on case units. Resour Conserv Recy 2018;134:184–95.
[7] Abadi Martın, Agarwal Ashish, Barham Paul, Brevdo Eugene, Chen Zhifeng, Citro Craig, et al. Tensorflow: Large-scale machine learning on heterogeneous systems, software available from tensorflow. org (2015). 2015, URL https://www.tensorflow.org.
[8] Paszke Adam, Gross Sam, Massa Francisco, Lerer Adam, Bradbury James, Chanan Gregory, et al. Pytorch: An imperative style, high-performance deep learning library. Adv Neural Inf Process Syst 2019;32:8026–37.
[9] Bradbury James, Frostig Roy, Hawkins Peter, Johnson Matthew James, Leary Chris, Maclaurin Dougal, et al. JAX: composable transformations of python+numpy programs. 2018, URL http://github.com/google/jax.
[10] Sezer Omer Berat, Gudelek Mehmet Ugur, Ozbayoglu Ahmet Murat. Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. Appl Soft Comput 2020;90:106181.
[11] Deb Chirag, Zhang Fan, Yang Junjing, Lee Siew Eang, Shah Kwok Wei. A review on time series forecasting techniques for building energy consumption. Renew Sustain Energy Rev 2017;74:902–24.
[12] Lim Bryan, Zohren Stefan. Time-series forecasting with deep learning: a survey. Phil Trans R Soc A 2021;379(2194):20200209.
[13] Rajagukguk Rial A, Ramadhan Raden AA, Lee Hyun-Jin. A review on deep learning models for forecasting time series data of solar irradiance and photovoltaic power. Energies 2020;13(24):6623.
[14] Dubey Ashutosh Kumar, Kumar Abhishek, García-Díaz Vicente, Sharma Arpit Kumar, Kanhaiya Kishan. Study and analysis of SARIMA and LSTM in forecasting time series data. Sustain Energy Technol Assess 2021;47:101474.
[15] Wang Jian Qi, Du Yu, Wang Jing. LSTM based long-term energy consumption prediction with periodicity. Energy 2020;197:117197.
[16] Liu Bingchun, Fu Chuanchuan, Bielefield Arlene, Liu Yan Quan. Forecasting of Chinese primary energy consumption in 2021 with GRU artificial neural network. Energies 2017;10(10):1453.
[17] Cai Mengmeng, Pipattanasomporn Manisa, Rahman Saifur. Day-ahead building-level load forecasts using deep learning vs. traditional time-series techniques. Appl Energy 2019;236:1078–88.
[18] Du Shengdong, Li Tianrui, Horng Shi-Jinn. Time series forecasting using sequence-to-sequence deep learning framework. In: 2018 9th international symposium on parallel architectures, algorithms and programming (PAAP). IEEE; 2018, p. 171–6.
[19] Lu Kuan, Meng Xiang Rong, Sun Wen Xue, Zhang Rong Gui, Han Ying Kun, Gao Song, et al. GRU-based encoder-decoder for short-term CHP heat load forecast. In: IOP conference series: materials science and engineering. Vol. 392, IOP Publishing; 2018, 062173.
[20] Du Shengdong, Li Tianrui, Yang Yan, Horng Shi-Jinn. Multivariate time series forecasting via attention-based encoder–decoder framework. Neurocomputing 2020;388:269–79.
[21] Marino Daniel L, Amarasinghe Kasun, Manic Milos. Building energy load forecasting using deep neural networks. In: IECON 2016-42nd annual conference of the IEEE industrial electronics society. IEEE; 2016, p. 7046–51.
[22] Laubscher Ryno. Time-series forecasting of coal-fired power plant reheater metal temperatures using encoder-decoder recurrent neural networks. Energy 2019;189:116187.
[23] Raidoo Renita, Laubscher Ryno. Data-driven forecasting with model uncertainty of utility-scale air-cooled condenser performance using ensemble encoder-decoder mixture-density recurrent neural networks. Energy 2022;238:122030.
[24] Rahman Aowabin, Srikumar Vivek, Smith Amanda D. Predicting electricity consumption for commercial and residential buildings using deep recurrent neural networks. Appl Energy 2018;212:372–85.
[25] Willard Jared, Jia Xiaowei, Xu Shaoming, Steinbach Michael, Kumar Vipin. Integrating physics-based modeling with machine learning: A survey. 2020, arXiv preprint arXiv:2003.04919.
[26] Rangapuram Syama Sundar, Seeger Matthias W, Gasthaus Jan, Stella Lorenzo, Wang Yuyang, Januschowski Tim. Deep state space models for time series forecasting. Adv Neural Inf Process Syst 2018;31:7785–94.
[27] Kraft Basil, Jung Martin, Körner M, Reichstein Markus. Hybrid modeling: Fusion of a deep approach and physics-based model for global hydrological modeling. Int Arch Photogramm Remote Sens Spat Inf Sci 2020;43:1537–44.
[28] Machalek Derek, Quah Titus, Powell Kody M. A novel implicit hybrid machine learning model and its application for reinforcement learning. Comput Chem Eng 2021;155:107496.
[29] Smyl Slawek. A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. Int J Forecast 2020;36(1):75–85.
[30] Raissi Maziar, Perdikaris Paris, Karniadakis George E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J Comput Phys 2019;378:686–707.
[31] Baydin Atilim Gunes, Pearlmutter Barak A. Automatic differentiation of algorithms for machine learning. 2014, arXiv preprint arXiv:1404.7456.
[32] Cho Kyunghyun, Van Merriënboer Bart, Gulcehre Caglar, Bahdanau Dzmitry, Bougares Fethi, Schwenk Holger, et al. Learning phrase representations using RNN encoder-decoder for statistical machine translation. 2014, arXiv preprint arXiv:1406.1078.
[33] Baydin Atilim Gunes, Pearlmutter Barak A, Radul Alexey Andreyevich, Siskind Jeffrey Mark. Automatic differentiation in machine learning: a survey. J Mach Learn Res 2018;18.
[34] Elliott Conal. The simple essence of automatic differentiation. Proc ACM Program Lang 2018;2(ICFP):1–29.
[35] Paszke Adam, Gross Sam, Chintala Soumith, Chanan Gregory, Yang Edward, DeVito Zachary, et al. Automatic differentiation in pytorch. 2017.
[36] Rackauckas Christopher, Nie Qing. Differentialequations.jl–a performant and feature-rich ecosystem for solving differential equations in julia. J Open Res Softw 2017;5(1).
[37] Molkenthin Christian, Scherbaum Frank, Griewank Andreas, Leovey Hernan, Kucherenko Sergei, Cotton Fabrice. Derivative-based global sensitivity analysis: Upper bounding of sensitivities in seismic-hazard assessment using automatic differentiation. Bull Seismol Soc Am 2017;107(2):984–1004.
[38] Yang Yilin, Achar Siddarth, Kitchin John. Evaluation of the degree of rate control via automatic differentiation. 2021, Authorea Preprints.
[39] Bell Ian H, Wronski Jorrit, Quoilin Sylvain, Lemort Vincent. Pure and pseudo-pure fluid thermophysical property evaluation and the open-source thermophysical property library CoolProp. Ind Eng Chem Res 2014;53(6):2498–508, URL http://pubs.acs.org/doi/abs/10.1021/ie4033999.
[40] The pandas development team. Pandas-dev/pandas: Pandas. 2020, http://dx.doi.org/10.5281/zenodo.3509134.
[41] Saltelli Andrea, Aleksankina Ksenia, Becker William, Fennell Pamela, Ferretti Federico, Holst Niels, et al. Why so many published sensitivity analyses are false: A systematic review of sensitivity analysis practices. Environ Model Softw 2019;114:29–39.