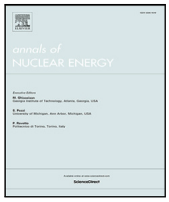
Annals of Nuclear Energy 211 (2025) 110893

Contents lists available at ScienceDirect

Annals of Nuclear Energy

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



Deep heterogeneous joint architecture: A temporal frequency surrogate model for fuel performance codes

Wenhan Zhou, Gustav Robertson *, Henrik Sjöstrand

Institution for Physics and Astronomy, Uppsala University, Uppsala, Sweden

ARTICLE INFO

Keywords:
Deep learning
Fuel performance modeling
Transuranus Code
TCN
FNO

ABSTRACT

Fuel performance codes, such as Transuranus, predict fuel behavior and are used to ensure the safe operation of nuclear reactors. These codes are moderately time-consuming and affordable in many applications but may be limited in others, primarily when many fuel rods must be evaluated simultaneously. This work presents how the temporal neural network techniques, Temporal Convolutional Networks, and a Fourier Neural Operator can be combined to form a deep heterogeneous joint architecture as a surrogate model for fuel performance modeling in time-critical situations. We train the model using realistic power histories and corresponding outputs generated using the fuel performance code Transuranus. The ultimate result is a surrogate model for use in time-critical situations that take milliseconds to evaluate for thousands of fuel rods and have a mean test error of unseen data around a few percent.

1. Introduction

Fuel performance codes (Van Uffelen et al., 2019) play a crucial role in forecasting fuel behavior to demonstrate safe operation. These codes consume a moderate amount of time, typically between seconds and a minute, depending on the complexity of the inputs. Such computational cost is adequate for most applications but can be limiting in applications that require rapid analysis of many fuel rods simultaneously. Examples of such applications include when fuel performance parameters are included in core optimization, and the algorithm must evaluate an entire fuel rod population in each iteration. Another example arises when predictions for a core of fuel rods are needed for online core monitoring. Similarly, there may be instances where a quick assessment of safety criteria fulfillment is necessary for an upcoming fuel cycle to prevent re-iteration of the core design. In addition to the time aspect, a simultaneous evaluation of many fuel rods usually requires an additional framework that prepares input files and processes output, which involves unnecessary overhead and may be impractical in certain implementations.

This paper addresses those limitations by proposing a neural network as a surrogate model trained on simulated data from the fuel performance code Transuranus. As fuel performance codes, such as Transuranus receive inputs regarding the fuel rod's initial geometrical design, material composition, and time-dependent irradiation conditions, the only difference in input between two rods of the same design is the irradiation conditions. Of those time-dependent parameters, the most important input is the Linear Heat Generation Rate (LHGR) (as negligible variations are expected in other parameters) for which the code calculates corresponding time-dependent output, including rod internal pressure, fuel centerline temperature, fission gas release, cladding oxidation, pellet-to-clad gap, and hydrogen pick-up. Therefore, one can generate training and validation data for developing a surrogate model for a given design by simulating many input power histories to obtain corresponding output predictions by keeping all other inputs fixed.

Developing a surrogate on such sequential data can be difficult. In the present work, we suggest a surrogate model that models power histories for a given rod design and provides corresponding time-dependent outputs. The proposed architecture combines a Temporal Convolutional Network (TCN) (Oord et al., 2016; Bai et al., 2018) and a Fourier Neural Operator (FNO) (Chi et al., 2020; Li et al., 2021). I.e., it uses a combination of causal and spectral convolution to form a joint model that effectively captures the input and output data's sequential features and frequency characteristics. The model is trained and validated using realistic power histories and corresponding output predictions generated using the fuel performance code Transuranus (Lassmann and Blank, 1988; Lassmann, 1992; Magni et al., 2021). Of particular interest is investigating whether we can develop a faster alternative to traditional fuel performance codes to facilitate the quick analysis of many fuel rods without preparing input files or processing outputs, as such computational efficiency is especially beneficial in situations demanding rapid decision-making, like the abovementioned applications.

* Corresponding author.
E-mail address: gustav.robertson@physics.uu.se (G. Robertson).

https://doi.org/10.1016/j.anucene.2024.110893
Received 11 July 2024; Received in revised form 26 August 2024; Accepted 28 August 2024
Available online 3 September 2024

2. Multivariate sequence-to-sequence modeling

Multivariate sequence data are observations where multiple variables are recorded at consistent sequence lengths. In the setting of multivariate sequence-to-sequence (seq2seq) modeling, the time-dependent input $\mathbf{X} \in \mathbb{R}_+^{N\times\tau\times v_{\text{in}}}$ is a tensor, where $1 \le s \le N$ is the number of data samples, $1 \le t \le \tau$ is the sequence time steps with maximal sequence length τ, and v_{in} is the number of input features. $\mathbf{Y}, \mathbf{Y}' \in \mathbb{R}_+^{N\times\tau\times v_{\text{out}}}$ represents the simulated outcomes of Transuranus and estimated outcomes respectively, with v_{out} being the number of output features. The task of a seq2seq neural network model:

$$\mathbf{Y}' = f(\mathbf{X}; \boldsymbol{\Theta}), \tag{1}$$

is to learn a mapping from $\mathbf{X}$ to $\mathbf{Y}'$ by finding the parameters $\boldsymbol{\Theta}$ from data such that an objective is minimized between $\mathbf{Y}'$ and $\mathbf{Y}$, see Fig. 4. The trained model with the optimal parameters can then infer unseen inputs.

A distinctive feature of seq2seq models is their ability to estimate multivariate time-dependent outcomes. Unlike Transuranus, which produces a single future output based on the current state, seq2seq models can create a parallel array of future values for each variable. This type of architecture can be executed very efficiently on accelerators such as GPUs.

Several deep neural network architectures model time-dependent data well (Vaswani et al., 2023; Yi et al., 2023). These models may be adapted to a seq2seq framework to suit our application. One well-established option is the WaveNet (Oord et al., 2016), also known as the TCN (Bai et al., 2018). This type of architecture is known for handling time-dependent data well due to the causal dilated convolution mechanism, see Section 2.1. Another approach is to model the temporal features of the data as frequency components by transforming the sequence to the frequency domain using the Fourier transform (Chi et al., 2020; Li et al., 2021), see Section 2.2. We use both approaches as baselines and build components for a more powerful co-trained model; see Section 3.2.

2.1. Temporal convolutional network

An important aspect we want to model in seq2seq tasks is the causal relationships within the sequential data. This causal relationship is primarily modeled through convolution operations. However, standard convolutional approaches face limitations when dealing with long sequences where boundary condition changes may significantly influence future predictions. The TCN addresses this challenge with dilated convolutions. This mechanism substantially expands the receptive field, enabling the model to incorporate a broad range of past information. We provide a minimal mathematical description of the TCN in the following section. Interested readers may find (Oord et al., 2016; Bai et al., 2018) useful for a more comprehensive understanding.

Let $1 \le l \le L$ be the number of layers of the model, $b \in \mathbb{Z}_+$, set to 2, be the dilation base and:

$$d^{(l)} = b^l, \tag{2}$$

be the dilation rate at the l:th layer. We denote the dilated convolution operation (Paszke et al., 2023) at the l:th layer as:

$$\mathbf{X}^{(l+1)} = \boldsymbol{\Theta}^{(l)}_{k,d^{(l)}} * \mathbf{X}^{(l)} + \mathbf{b}^{(l)}, \tag{3}$$

where $\boldsymbol{\Theta}^{(l)}_{k,d^{(l)}} \in \mathbb{R}^{O\times I\times k}$ is the weight kernel with O and I denoting the number of output and input channels of the operation respectively. Furthermore, $1 \le k \le \tau$ is the kernel size, and $\mathbf{b}^{(l)} \in \mathbb{R}^O$ is the offset. The convolution operation can be intuitively understood with Fig. 1, and the dilation mechanism can be understood with Fig. 2, without the offset. We use $h_{\text{model}} = O = I$ as a hyperparameter representing the model's latent dimension from the next section.

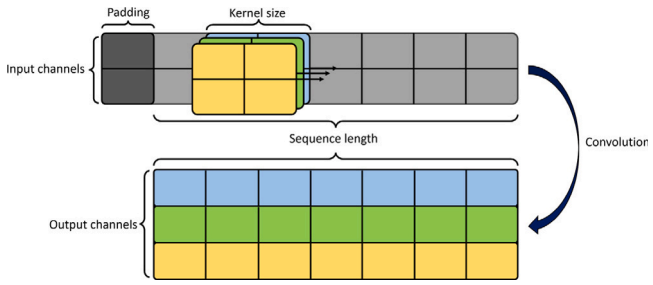


Fig. 1. Convolution with a kernel of dilation rate $d^{(l)} = 1$ and kernel size $k = 2$ performed on a input sequence with input channels $I = 2$. The input dimension of the kernel is also $I = 2$, in agreement with the input channels of the input sequence. There are $O = 3$ distinct convolution filters of the kernel. The operation is performed in parallel for each one of the channels, resulting in an output sequence with $O = 3$ output channels. Zero padding is used at the start of the input sequence to ensure that the sequence length is preserved.

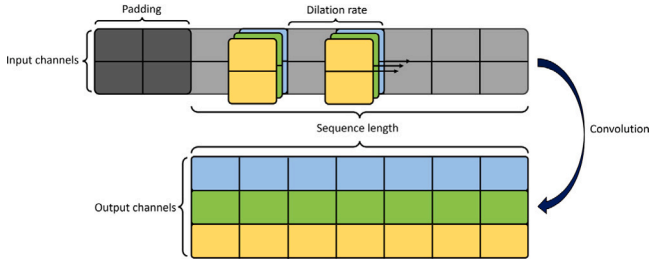


Fig. 2. Convolution with a kernel of dilation rate $d^{(l)} = 2$ and kernel size $k = 2$. This corresponds to the kernel being separated by one space. The padding is increased to compensate for the increased dilation rate.

Due to the exponentially increasing dilation rate as suggested by Eq. (2), the perceptive field of the dilated convolution will increase exponentially concerning the number of dilated convolutional layers stacked sequentially. We incorporate a sufficient number of layers so that the perceptive field covers the whole sequence. The relationship between the number of layers for full history coverage depends on the kernel size, sequence length, and dilation base (Bai et al., 2018):

$$L = \left\lceil \log_b \left(1 + \frac{(\tau - 1)(b - 1)}{2(k - 1)} \right) \right\rceil. \tag{4}$$

Another factor to consider is adjusting the size of the padding $p^{(l)}$ to ensure that the sequence length of the transformed input is maintained consistently throughout the layers (Bai et al., 2018):

$$p^{(l)} = d^{(l)} \cdot (k - 1). \tag{5}$$

To efficiently describe the workings of a TCN, we first consider the transformation of the initial input into a latent variable. This is achieved by:

$$\mathbf{X}^{(1)} = \left(\mathbf{X}\mathbf{W}_{\text{in}} + \mathbf{b}_{\text{in}} \right)^{\top}, \tag{6}$$

where the input $\mathbf{X}$ is multiplied by a weight matrix $\mathbf{W}_{\text{in}} \in \mathbb{R}^{v_{\text{in}}\times h_{\text{model}}}$, and $\mathbf{b}_{\text{in}} \in \mathbb{R}^{h_{\text{model}}}$ is summed. The weight matrix mixes the time variables with the LHGRs into the latent variable. The TCN then performs a series of non-linear transformations through its layers. Each layer l, executes the following operation:

$$\mathbf{X}^{(l+1)} = \sigma^{(l)} \left(\boldsymbol{\Theta}^{(l)}_{k,d^{(l)}} * \mathbf{X}^{(l)} + \mathbf{b}^{(l)} \right), \tag{7}$$

where $\sigma^{(l)}$ denotes a non-linear activation function applied element-wise. These operations progressively transform the data, integrating non-linear temporal features at each layer. Finally, the output of the TCN is obtained by projecting the transformed data from the last layer

to match the desired output dimensions and applying a Rectified Linear Unit (ReLU) function to ensure that the outputs are positive:

$$\mathrm{TCN}(\mathbf{X}; h_{\text{model}}) = \mathrm{ReLU}\left(\mathbf{X}^{(L)^\top}\mathbf{W}_{\text{out}} + \mathbf{b}_{\text{out}}\right), \tag{8}$$

where $\mathbf{W}_{\text{out}} \in \mathbb{R}^{h_{\text{model}} \times v_{\text{out}}}$ and $\mathbf{b}_{\text{out}} \in \mathbb{R}^{v_{\text{out}}}$. This final step ensures that the TCN's output aligns with the dimensionality of the target $\mathbf{Y}$.

2.2. Fourier neural operator

A different method for seq2seq modeling is to train a model using the frequency domain representations of the data, which differs from the causal time-domain training achieved with convolution. Training in the frequency domain allows for a model capable of capturing the global characteristics of the sequences. In the following section, we describe the core mechanism of FNO. Interested readers can find more details in Li et al. (2021).

The core component of the FNO architecture is the spectral convolution (Rippel et al., 2015), also known as the Fast Fourier Convolution (FFC) (Chi et al., 2020). This is an alternative way of performing the temporal (or spatial) convolution, but in the frequency domain granted by the convolution theorem:

$$\boldsymbol{\theta}^{(l)}_{1,1} * \mathbf{X}^{(l)} = \mathcal{F}^{-1}\left[\mathcal{F}[\boldsymbol{\theta}^{(l)}_{1,1}] \odot \mathcal{F}[\mathbf{X}^{(l)}]\right], \tag{9}$$

where $\mathcal{F}$ denotes the real fast Fourier transform, and $\odot$ denotes the element-wise multiplication with respect to the frequency components. Building on the foundational concept of the convolution theorem, the parameters $\boldsymbol{\theta}^{(l)}_{\xi} \in \mathbb{C}^{O \times I \times M}$, can be represented in the frequency domain without explicitly performing the Fourier transform. Note that we only keep the first M non-zero frequency components, sorted from low to high, of the frequency kernel and the Fourier-transformed latent variable $\mathcal{F}[\mathbf{X}^{(l)}]$. The motivation is to eliminate some less important computations and to prevent overfitting (Li et al., 2021). The FFC operation is thus given by:

$$\mathrm{FFC}^{(l)}(\mathbf{X}^{(l)}) = \mathcal{F}^{-1}\left[\boldsymbol{\theta}^{(l)}_{\xi} \odot \mathcal{F}[\mathbf{X}^{(l)}]\right]. \tag{10}$$

In practice, we generalize the convolution theorem such that the input channels are mapped to the output channels in parallel, shown as follows:

$$\boldsymbol{\theta}^{(l)}_{\xi} \odot \mathcal{F}[\mathbf{X}^{(l)}] = \sum_{i=1}^{I} \hat{\theta}^{(l)}_{OiM} \cdot \hat{x}^{(l)}_{NiM}, \tag{11}$$

where the lower-case letters denote the elements of their respective tensors. The operation in Eq. (11) intuitively represents a 1D convolution with $k = 1$, weighted for each frequency component. This mechanism allows the model to learn the importance of each frequency component while keeping the property of performing a convolution operation.

After performing the FFC operation, an affine transformation is applied in the temporal domain, followed by a non-linear activation function, expressed as:

$$\mathbf{X}^{(l+1)} = \sigma^{(l)}\left(\boldsymbol{\theta}^{(l)}_{1,1} * \mathrm{FFC}^{(l)}(\mathbf{X}^{(l)}) + \mathbf{b}^{(l)}\right). \tag{12}$$

In essence, the FNO architecture consists of solving Eq. (12) followed by a feed-forward layer and applying ReLU:

$$\mathrm{FNO}(\mathbf{X}; h_{\text{model}}) = \mathrm{ReLU}\left(\mathbf{X}^{(L)^\top}\mathbf{W}_{\text{out}} + \mathbf{b}_{\text{out}}\right). \tag{13}$$

Apart from the dilated convolution mechanism, the FNO may be interpreted as an alternative formulation of a convolutional neural network where the training occurs in the frequency domain. Once again, we omitted some technical details in the architectural description.

3. Dual domain modeling

LHGRs that show rapid changes are more informative than smoother ones because they provide localized information about how the fuel rod system reacts to critical events. This sharpness in the signal allows for pinpointing specific moments when the physical behavior changes abruptly. In contrast, smooth LHGRs distribute information over a broader time scale. While this provides a general understanding of trends, it increases the uncertainty about the exact timing and magnitude of changes in the output features. Generally, smoother signals in the time domain are more concentrated in the frequency domain, and vice versa (William Beckner, 1975). Therefore, smooth LHGRs possess fewer dominant frequency components, making them more distinct and easier to analyze in the frequency domain. This scenario would benefit FNO, which has most parameters embedded in the frequency domain. However, it is difficult to determine in advance whether the time or frequency is the advantageous domain for a general dataset. Choosing an architecture based on the characteristics of LHGRs is, therefore, impractical.

3.1. Deep ensembles and joint architectural models

Using an ensemble in machine learning has been widely adopted to increase the generability of the models. Similar work has been done in deep learning where the ensemble members are deep neural networks of the same architecture with different weight initialization trained on the same dataset. The combined results of the various trained ensemble members showed significantly better performance than the individual ensemble members (Fort et al., 2020).

An ensemble consisting of the same architectures is known to be *homogeneous*, and an ensemble with mixed architectures is known to be *heterogeneous*. The main advantage of the latter is that different architectures may learn different features from the dataset, thus being able to represent more complex relationships once combined. This motivates a joint architecture comprising TCN and FNO due to their complementary nature. However, not every combination of base models will result in improved performance (Wang et al., 2023). We have employed a solution to eliminate the possibility of performance degradation due to poor choice of model combination by co-training the joint architecture. This motivates a blending mechanism that combines the models such that the training process can automatically decide the importance of each candidate.

3.2. Temporal frequency network

The resulting joint architecture we propose consists of two complementary architectures, TCN and FNO,[1] with the shared similarity of the convolution operation, but where the parameters are represented in different domains. These two models are combined through a feed-forward layer to achieve a data-driven architectural combination. We refer to this model as the Temporal Frequency Network (TFN), schematically depicted in Fig. 3. To construct the TFN, the TCN and FNO are first concatenated in the channel dimension:

$$\mathbf{Z} = \mathrm{Concat}\begin{pmatrix}\mathrm{TCN}(\mathbf{X}; h_t)\\ \mathrm{FNO}(\mathbf{X}; h_{\xi})\end{pmatrix}, \tag{14}$$

where h_t and h_{ξ} denote the latent dimension of the temporal and frequency model respectively, and $\mathbf{Z} \in \mathbb{R}^{N \times r \times (h_t + h_{\xi})}$. This tensor is then projected to the desired output dimension using a feed-forward layer without bias:

$$\mathrm{TFN}(\mathbf{X}; h_{\xi}, h_t) = \mathbf{Z}\mathbf{W}_{\text{combine}}, \tag{15}$$

where $\mathbf{W}_{\text{combine}} \in \mathbb{R}^{(h_{\xi}+h_t) \times v_{\text{out}}}$.

[1] Although arbitrary causal seq2seq model can be used as a drop-in replacement for the TCN. Similarly, variants of FNO are, in principle, also allowed.

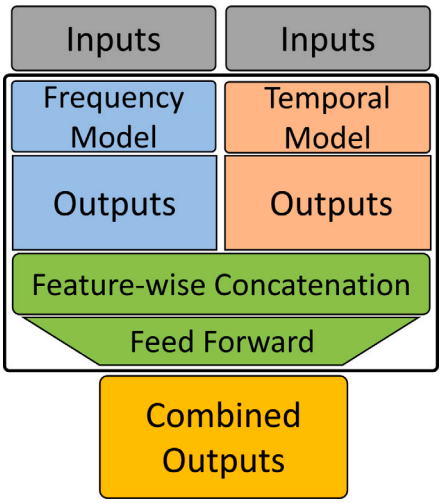


Fig. 3. Architectural design of TFN. The temporal model and frequency model are co-trained end-to-end in a joint fashion. Their outputs are weighted through a feed-forward layer to produce the final outputs.

4. Methodology

This section outlines the methodology for deriving a surrogate model of Transuranus. The main idea is to train several variants of TCNs and FNOs and compare them against TFNs in a similar computational class. Then, performing an extended training on the most promising architecture and hyperparameters. To perform such training and validation, the methodology must also include steps for data generation, pre-processing, splitting, and selecting appropriate loss metrics. These steps, as well as how we set up the candidate architectures and select the final model, are described in the following sections. Moreover, Fig. 4 depicts how the training process components connect.

4.1. Data generation

First, we generate the dataset by inputting a set of 12941 realistic power histories corresponding to those from typical pressurized water reactors (PWRs). The data is provided by Westinghouse Electric Sweden AB and given as the LHGR offered for ten axial nodes and expressed in $\mathrm{kW\,m^{-1}}$. The received power histories are used with a typical Transuranus PWR input file to generate corresponding time-dependent output data. The outputs are selected to represent key safety output parameters normally estimated in fuel performance analyses and are listed below. We selected these quantities because they are typically critical in fuel licensing evaluations, allowing us to demonstrate the methodology in a realistic context. However, this study serves as a proof of concept, and the variables can be adjusted based on future needs. These variables consist of c_v features, which could be one or ten. In the former case, $c_v = 1$ resembles a global measure of the whole rod, while $c_v = 10$ resembles the ten nodal axial positions of the rod.

1. Fuel centerline temperature ($c_1 = 10$): This is the temperature at the centerline of the pellet stack. This varies axially and is highly correlated with the LHGRs.
2. Central void pressure ($c_2 = 1$): This is the rod's internal pressure, which is constituted by the fill gas pressure and is exerted by fission gas released during operation.
3. Oxidation thickness ($c_3 = 10$): The thickness of oxidation that grows on the fuel cladding.
4. Gap width ($c_4 = 10$): Describes the space width between the fuel and cladding.
5. Hydrogen absorption ($c_5 = 10$): A common coolant is water molecules H_2O. When oxidation occurs, the cladding picks up the Oxygen, leaving Hydrogen radicals that the cladding may absorb. This variable is, therefore, highly correlated with the oxidation thickness.
6. Integral fission gas release ($c_6 = 1$): The total amount of fission gas released during an operation.
7. Integral fractional gas release ($c_7 = 1$): The total amount of fission gas released compared to the amount created.

Given that we have four nodal outputs, each with ten axial nodes, and three global variables describing the whole rod, we have in total $v_{\text{out}} = 4 \cdot 10 + 3 \cdot 1 = 43$ output features. The entire data set, including the input power histories and the output variables, are depicted for each channel in Fig. 5. The data is plotted against time, which is an input variable in the data set.

4.2. Data pre-processing

The resulting samples in the generated dataset from Transuranus have varying sequence lengths. Therefore, linear interpolation using `F.interpolate` is employed (Paszke et al., 2024), with keyword argument `align_corners=True` to interpolate the data to have consistent sequence lengths. Here, we use the sequence length 80. In addition to the interpolation, each variable is scaled by dividing by its global maximum value. For quantities provided for in several nodes, each node is normalized individually. Using normalization guarantees that the dataset is unitless. We use global normalization to retain the relative difference between rods and features.

4.3. Data shuffling and splitting

After the data pre-processing, the dataset is shuffled and divided into a training set with $N_{\text{train}} = 9992$ samples and a test set of $N_{\text{test}} = 2499$ samples (one sample here is one set of power histories and corresponding output curves for one fuel rod). This step ensures that the test data remains independently and identically distributed (IID) with the training set, which is necessary for the model to generalize beyond the training data. The train-test-split is repeated five times so that different parts of the dataset become the test set once (5-fold cross-validation), see Fig. 6. This allows for performing five independent training runs with the same model architecture on different validation folds, thus allowing us to measure the variation in the error caused by splitting the dataset differently.

4.4. Evaluation loss

In this work, the nature of the training and test data requires carefully selecting loss metrics. This is partly because the scaling results show that some samples are close to zero for variables that span several orders of magnitude. It is also because the number of axial nodes inherently weighs some output features. For example, variables such as central void pressure and integral fission gas release are underrepresented compared to other variables, as there is only one output channel per variable, as opposed to variables such as temperature that are given for several axial nodes. This results in, for example, a standard mean square error loss that prioritizes minimizing the whole-rod quantities in favor of nodal ones. To alleviate those issues, we, therefore, consider a metric that both scales according to the variables' magnitudes and balances importance between the output features. We do this in a two-stage process where we first choose a variant of the Normalized Root Mean Squared Error (NRMSE) to calculate a feature-specific loss for each fuel rod according to:

$$\mathcal{L}_{s,v,c}(\mathbf{Y}, \mathbf{Y}') = \frac{100\sqrt{\frac{1}{\tau}\sum_{t=1}^{\tau}(y_{s,t,v,c} - y'_{s,t,v,c})^2}}{\frac{1}{\tau}\sum_{t=1}^{\tau} y_{s,t,v,c}}, \tag{16}$$

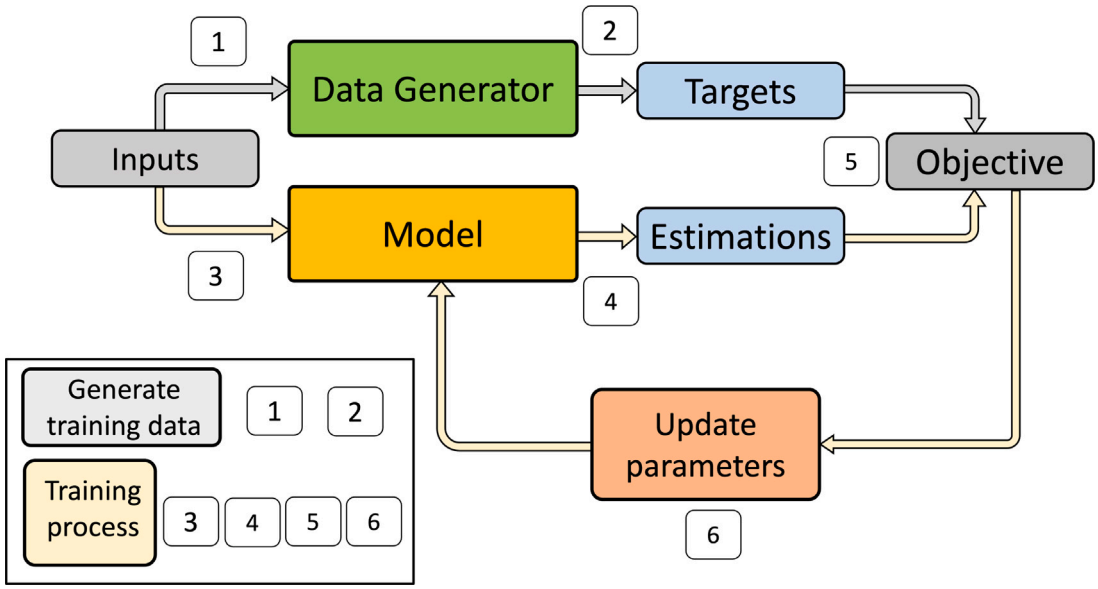


Fig. 4. Simplified diagram of how training a neural network. We begin by passing the inputs, e.g., a batch of LHGRs with corresponding time steps, into the data generator Transuranus (1) from which a target dataset is obtained (2). The inputs are then passed into the model (3), and produce a set of estimations (4). The outcomes are then compared to the targets, and an objective function is used to evaluate the quality of the estimations (5). Finally, the parameters of the model are updated to optimize the given objective (6). This procedure is iterated until the convergence of the objective function.

where s, v, and $1 \leq c \leq 10$ denote the sample (one fuel rod case), the variable, and the channels, respectively, and $\mathbf{Y}'$ is the output tensor with the model estimations. I.e., Eq. (16) reduces the time dimension for each rod and output feature and makes a rod-wise normalization to the average magnitude of each feature. To compute a variable-specific loss, we further rebalance each variable's loss contribution by normalizing it to that variable's number of channels (either ten or one), which results in one loss per variable and sample (fuel rod):

$$\mathcal{L}_{s,v}(\mathbf{Y}, \mathbf{Y}') = \frac{1}{c_v} \sum_{c=1}^{c_v} \mathcal{L}_{s,v,c}(\mathbf{Y}, \mathbf{Y}'). \tag{17}$$

We obtain the sample-wise loss by computing the average over the variables:

$$\mathcal{L}_s(\mathbf{Y}, \mathbf{Y}') = \frac{1}{V} \sum_{v=1}^{V} \mathcal{L}_{s,v}(\mathbf{Y}, \mathbf{Y}'), \tag{18}$$

for $V = 7$ variables. Training typically requires dividing the data into several subsets, commonly known as mini-batches B_i is a subset of the dataset, with a sample size of $|B_i|$ for the i:th mini-batch. In this work, the target loss is calculated by averaging Eq. (18) over the samples (i.e., the fuel rods) present in the mini-batch:

$$\mathcal{L}^{(i)}(\mathbf{Y}, \mathbf{Y}') = \frac{1}{|B_i|} \sum_{s=1}^{|B_i|} \mathcal{L}_s(\mathbf{Y}, \mathbf{Y}'), \tag{19}$$

which is the loss function used to update the weights at each iteration.

Since there are $N_B = \lceil N_{\text{train}}/|B_i| \rceil$ mini-batches in the training set, we also need a metric that measures the loss per epoch. This is used only for a performance measure, not for training, and is given by computing an average of Eq. (19) over the number of mini-batches:

$$\mathcal{L}_{\text{epoch}}(\mathbf{Y}, \mathbf{Y}') = \frac{1}{N_B} \sum_{i=1}^{N_B} \mathcal{L}^{(i)}(\mathbf{Y}, \mathbf{Y}'). \tag{20}$$

Note that the same metric is also used to evaluate the test loss without dividing the data into mini-batches. Hence, for the test data, the loss per epoch is computed with $|B_i| = N_{\text{test}}$ and $N_B = 1$.

4.5. Surrogate model candidates

Before selecting a final architectural design, we evaluate a set of candidates, and the best-performing candidate will be further selected for more extensive training. First, we propose two base designs: a single TCN, FNO, and a combined TFN (TCN-v1, FNOv-1). Note that we intentionally choose the first $M = 32$ non-zero frequency components to match the number of parameters of FNOs with TCNs (for details, see Section 2.2). In those, as well as in all other designs, the number of channels used between the first and last layer is subject to choice, but a fully connected layer is required in the final step so that the output shape adapts to that of the output. TCN-v1, FNO-v1, and TFN-v1 are all designed to have the same number of latent dimensions. Still, as the TFN is more complex with more parameters, it requires a more significant computation cost to be evaluated. Therefore, we also construct two single models, TCN-v2 and FNO-v2, with a similar number of trainable parameters and a more comparable computational cost. Since the FNO design is shallow and needs only one layer ($L = 1$), it is typically faster than a TCN that requires more depth to reach full history coverance ($L = 5$ with a kernel size of $k = 3$). Therefore, we propose a more complex FNO-v3 with approximately the same inference time as TCN-v2. Finally, we propose an ensemble model, TFN-v2, where the intermediate number of channels for both the FNO and the TCN is selected to have an equal computational cost between the two components. The candidates are listed below in Table 1.

Table 1
Showcasing the performances of variants of the same model architecture of different sizes.

Architecture	h_s:h_t	Parameters
TCN-v1	29	32 616
TCN-v2	42	66 871
FNO-v1	29	32 187
FNO-v2	42	66 247
FNO-v3	165	990 715
TFN-v1	29:29	66 500
TFN-v2	80:16	252 813

5. Results

We evaluate the models in the previous section using five-fold cross-validation of the processed dataset. This ensures that the randomness that arises when splitting the dataset is captured. We train all models

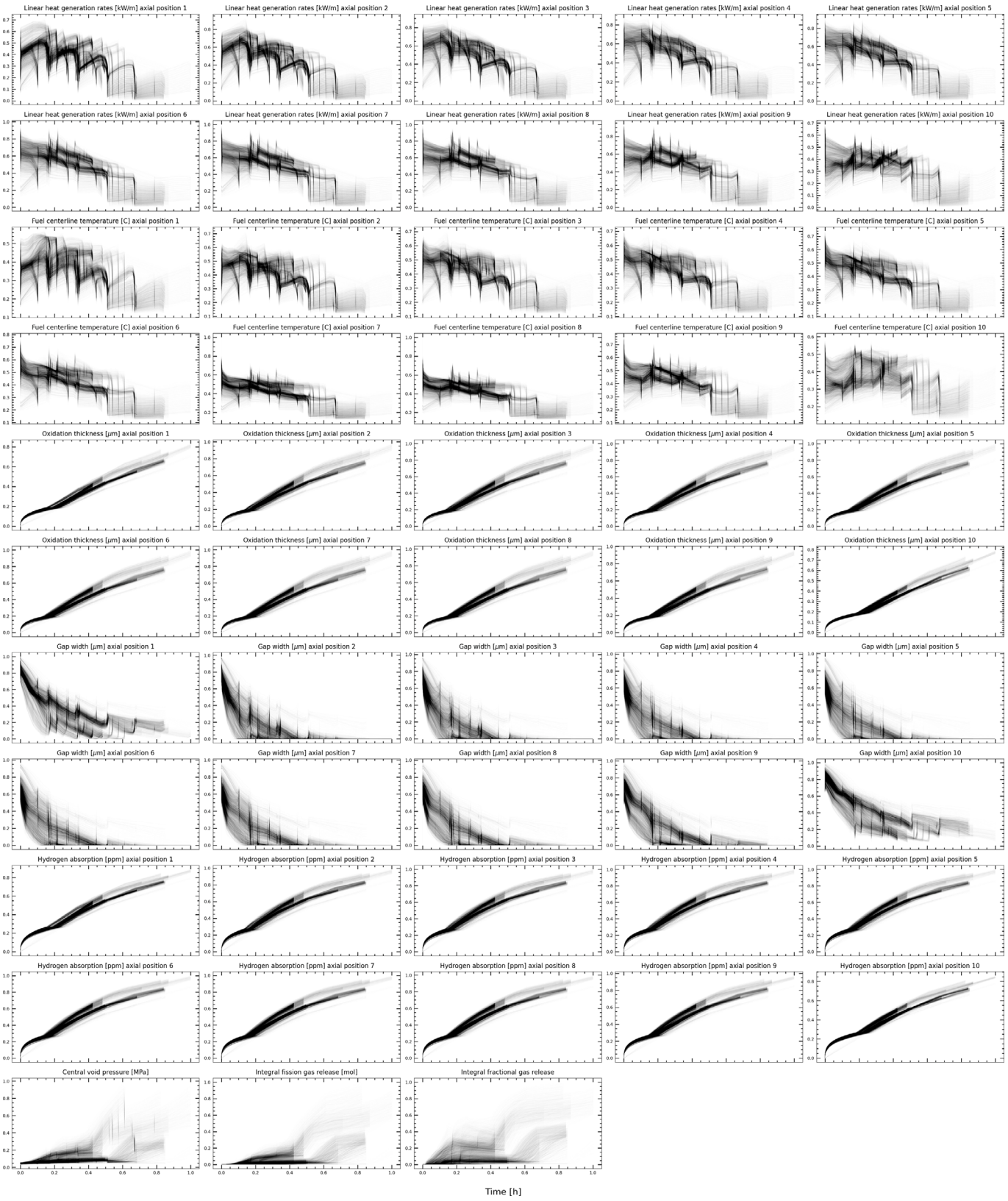


Fig. 5. The complete dataset used for training and testing. The first ten figures represent the input LHGRs **X** for each axial position of a rod. The 43 remaining figures together represent the target **Y**. The trained model will likely perform well on samples in the regions with high data density, see Fig. 10. Conversely, the model's performance is expected to degrade for regions with low data density, see Fig. 11. Model outputs that significantly deviate from those shown in this figure cannot be trusted.

using the AdamW (Loshchilov and Hutter, 2017) optimizer with default settings for 500 epochs using a mini-batch size $|B_i| = 200$. The learning rate is set up to decay with 10% when there are no improvements in the training loss for the latest 100 epochs. The average losses are calculated fold-wise for the test data using Eq. (17) averaged over all samples, variables, and channels. For an inference time analysis, we employ the

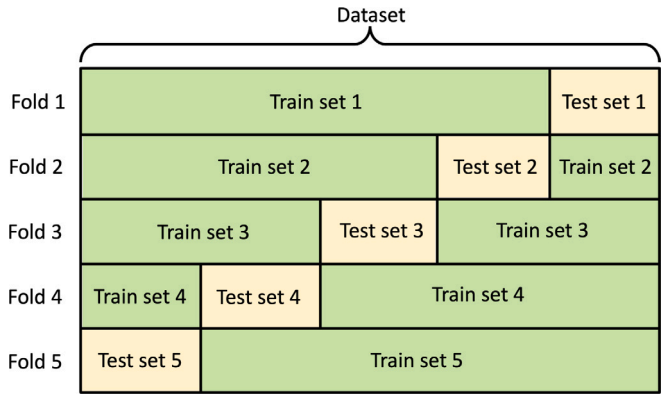


Fig. 6. Demonstrating 5-fold cross-validation for a shuffled dataset. The same model architecture is trained on each validation fold. Hence, all parts of the dataset will be used for testing once, which maximizes the data utilization for model evaluation.

Table 2
Showcasing the performances of variants of the same model architecture of different sizes. The model is chosen at a checkpoint with the smallest test loss within the 500 epochs of each validation fold. The error shown in the table is the average of those five smallest values.

Architecture	Error	Inference time
TCN-v1	6.70 ± 0.40	9.7 ms
TCN-v2	5.92 ± 0.26	13.0 ms
FNO-v1	27.53 ± 7.84	2.2 ms
FNO-v2	9.03 ± 0.50	3.3 ms
FNO-v3	5.86 ± 0.84	12.6 ms
TFN-v1	4.20 ± 0.05	12.4 ms
TFN-v2	3.78 ± 0.21	12.7 ms

best-performing model to process 1000 input samples repeated 1000 times using an NVIDIA TESLA P100 GPU. The average and the standard deviation of the errors from each fold are reported for each model along with the average inference time in Table 2.

In addition to the errors and the inference times presented above, loss curves for the different models are provided in Fig. 7. Here, the loss is averaged for the five cross-validation runs shown for the training and test data sets.

5.1. Final surrogate model selection

As seen in the previous section, Table 2 shows that TFN-v2 is the best-performing model with a reasonable inference time. From Table 2, it is evident that FNO-v3 performs comparably to TFN-v1 and TFN-v2 regarding error metrics. However, we selected TFN-v2 for extensive training because it offers the optimal balance between minimal error and a manageable number of parameters, as seen in Table 1. Additionally, Fig. 7 indicates no significant signs of over-fitting for all models. Furthermore, TFNs tend to converge faster while being more robust to changes in the data, as shown by the smaller uncertainties from the cross-validations. Therefore, TFN-v2 is selected as the final surrogate model for extended training, where we extend the number of training epochs from 500 to 5000. The training is done for the first validation fold only. The loss curve for this extensive training is presented in Fig. 8. As seen from the figure, extensive training leads to increased performance in that the loss decreases without obtaining any significant tendency to overfit. Compared with previous loss curves, we can also see that the loss curve from the extended training keeps decreasing without converging, indicating that further training may be beneficial.

We have calculated each variable's mean and median relative errors to provide an overview of how the final model performs. The results are presented in Table 3, indicating that the model performs well on the test data.

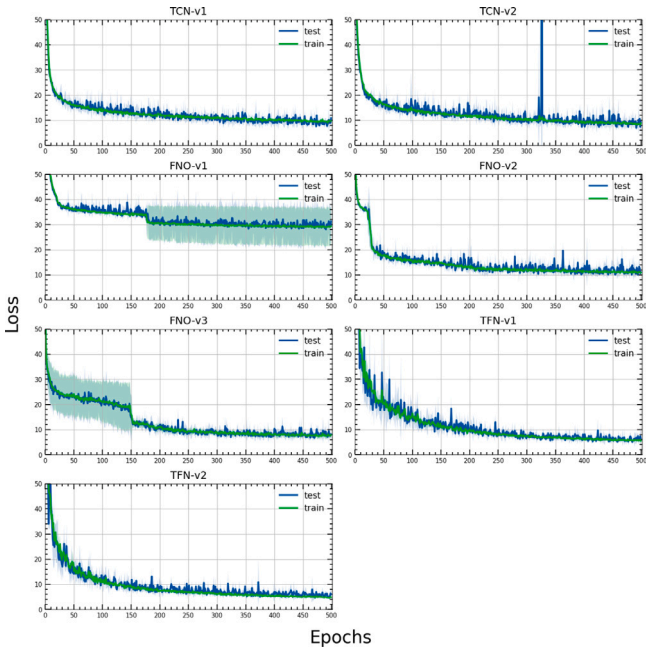


Fig. 7. Average loss curves over the cross-validations are plotted in the y-axis per epoch for the models. Each loss curve corresponding to individual validation folds is computed using (20). The uncertainty arises from the cross-validation, shown in one unbiased standard deviation. Models with small uncertainty are more robust to changes in data.

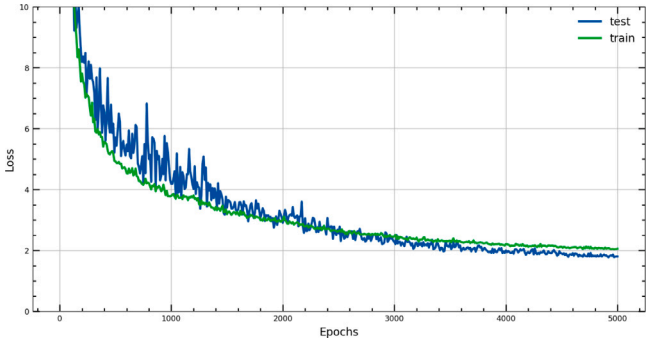


Fig. 8. Loss curves for TFN-v2, trained for 5000 epochs on the first cross-validation fold. For visualization purposes, the loss is averaged for every ten epochs.

Table 3
Mean and median errors over the samples calculated using Eq. (17) for the different output variables based on the IID test data. The most difficult variables are the measures for fission gas release.

Variable	Relative error [%]		
	Mean	Median	Std Dev
Centerline temperature	0.79	0.71	0.38
Rod internal pressure	1.20	0.97	1.04
Oxide layer thickness	0.74	0.64	0.51
Gap width	2.16	1.72	2.22
Hydrogen pick-up	0.67	0.58	0.46
Fission gas release	3.33	2.55	3.55
Fractional gas release	3.03	2.42	3.10
Average	1.70	1.37	–

5.2. Final surrogate performance

To showcase the performance of the obtained final surrogate model, we present how it estimates some typical cases in the test data. Since the training and test data sets include different cycles, we have selected

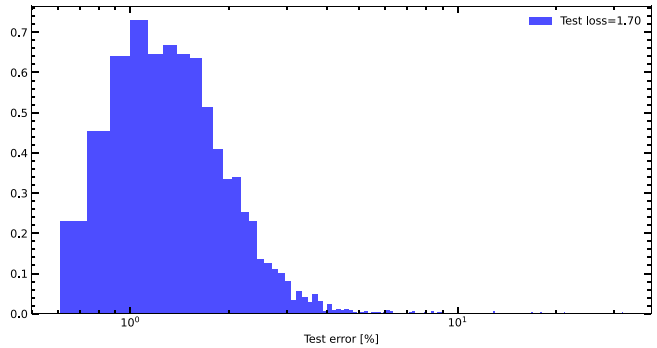


Fig. 9. Test-loss distribution. Most independent test predictions agree with data with relevant low errors, less than 5% according to Eq. (18).

a typical two and five-cycle rod for the demonstration. Those results are presented in Fig. 10. As can be seen from the figure, there is typically a significant resemblance between estimations and independent test data calculated with Transuranus which aligns with the results in terms of low errors presented in Table 3.

Despite that, we obtained low errors on average; certain test cases experienced high losses. This is especially seen when looking at the distribution of test losses as presented in Fig. 9. A more thorough examination of these cases reveals that they either involve power histories affected by burnable absorbers – all rods are modeled in Transuranus as standard UO2 rods without burnable absorbers, but some power histories represent rods with burnable absorbers – or cases where the data exhibit uneven temporal spacing between data points. The latter is an artifact coming from that Transuranus tends to insert points more closely when convergence is not achieved. As the interpolation method used to resample the data does not ensure consistent time steps in the final result, this issue is further amplified, leading to clusterlike clouds of data points that are very close together, as shown in Figs. 11(a) and 11(b). To conclude, we have two types of cases for which the model does not perform well. This data is, however, underrepresented, and there are only about 0.6% such cases in the whole dataset.

6. Discussion

This work proposes a data-driven surrogate model that uses seq2seq neural networks to replace the fuel performance code Transuranus in specific time-critical applications. For example, if all the fuel rods in the core need to be evaluated within less than a second. The proposed surrogate model is a joint architecture that uses a TCN and an FNO to form a TFN. That is a joint architectural model consisting of TCN and FNO that can simultaneously learn complementary sequence characteristics.

This type of multi-scale modeling is not new in the machine-learning community, and the idea of multi-scale learning has been deployed in many different forms. For example, the inception architecture (Szegedy et al., 2014) uses convolutional kernels of various sizes at each layer, and the TS-TFC (Liu et al., 2023) uses the concept of temporal-frequency co-training to perform semi-supervised learning for time series data. Our TFN differs, however, from TS-TFC in the temporal and spectral models. Furthermore, the TFN combines two models through a linear map to perform a supervised regression task. Another model that uses both the temporal- and frequency domains is BTSF (Yang and Hong, 2022), which performs unsupervised time series representation.

Through our experiments, we have shown that the TFN performs well on tests that are IID concerning the training data. However, we want to emphasize that our model is trained to map the inputs to the targets, and it does not learn how to generalize beyond the train and test distributions without a significant increase in error. The model is designed to operate within the data distribution presented in Fig. 5 with lower errors in regions with high data density and vice versa.

The samples with the largest errors shown in Fig. 11 are typically underrepresented in the dataset as these are either rods with power histories affected by burnable absorbers (with different power levels during the first cycle) or rods with very inconsistent lengths between time steps. We argue that the inconsistent length between time steps is less problematic because, in a real-life application, a consistent time-step series would be employed. To reduce the error, it is thus advised that in future work, we try to include a more even selection of input power histories, where all types of operations are represented equally.

In this work, we interpolate the data to a consistent sequence length of 80, but the model can process larger input sequence lengths than it was trained on. The maximum perceptive field under the current setup is, however, 125. That means that for longer sequence lengths, the TFN may not accurately account for the earlier behavior of the input variables. In addition, the minimal sequence length is 65, as limited by the number of frequency components. We recommend interpolating to the same number of points on which the model was trained and evaluated.

7. Conclusions

In the presented work, we have tested whether a seq2seq neural network surrogate model can approximate the fuel performance code Transuranus under certain conditions. We have tested both a model in the time domain (TCN) as well as in the frequency domain (FNO) and a heterogeneous joint model architecture (TFN). We have demonstrated that the suggested architectures are suitable for mapping multiple power history sequences to corresponding multivariate output sequences in fuel performance modeling. Given a fixed computational cost, we conclude that the TFN outperforms both the TCN and the FNO. More specifically, the TFN demonstrates faster convergence and good robustness to data variations during cross-validation.

The TFN was then able to estimate critical fuel performance variables such that the test error for most variables when using the best-performing model, is below a few percent. Consequently, in scenarios where surrogate models for correlating power histories with predictions of fuel performance phenomena are lacking, employing a TFN emerges as a promising solution.

8. Outlook

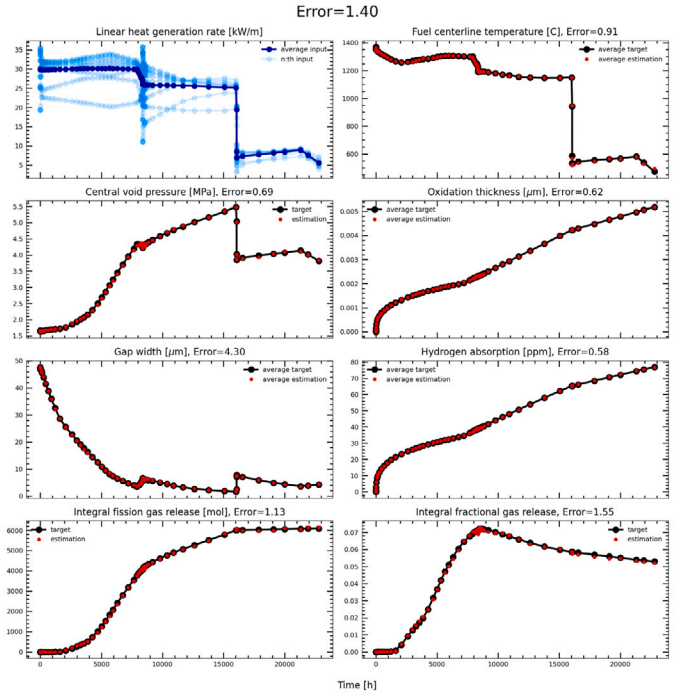
If computational resources are available, we recommend increasing the training time further as the loss curves in Fig. 8 did not fully converge. Furthermore, we recommend reusing the models evaluated for cross-validation by constructing a bagging-style ensemble, which could further decrease the estimation error.

This study found significant errors in a limited number of fuel rods modeled as standard rods despite their power histories influenced by burnable absorbers. It is recommended to include a broader range of fuel rod designs with varying burnable absorber levels to improve the model's generalizability. In future studies, the initial composition should be also explicitly accounted for rather than being implicitly included in the power history.

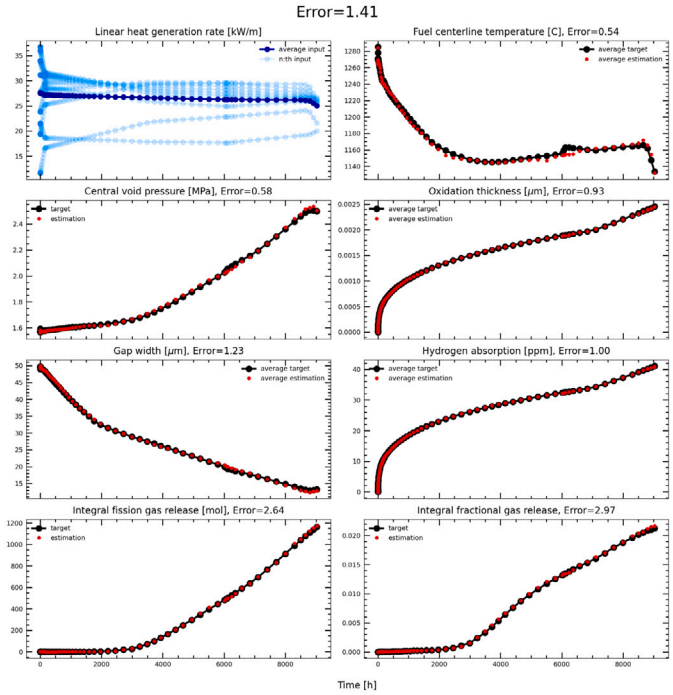
Finally, a future recommendation is to incorporate uncertainty estimation, for example, using a Bayesian framework, so a model prediction uncertainty can accompany that model prediction.

Declaration of competing interest

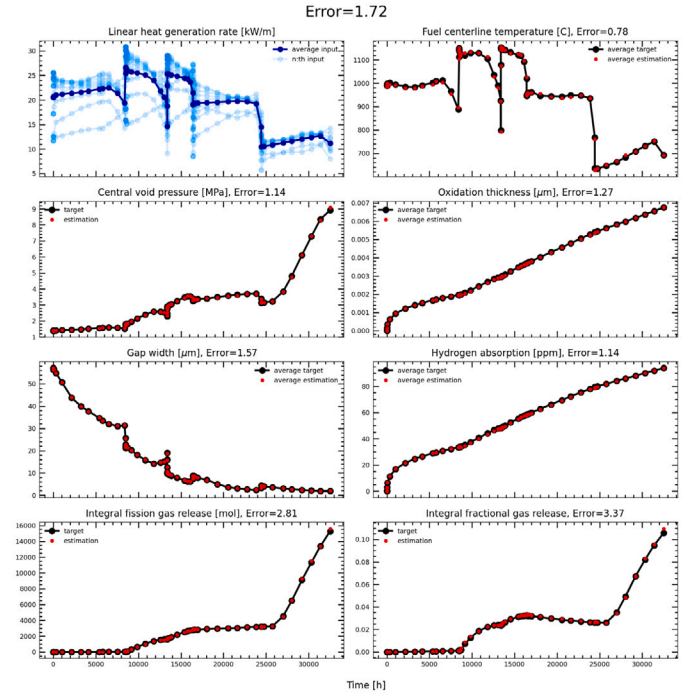
The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Henrik Sjostrand reports financial support was provided by Swedish Centre for Nuclear Technology (SKC). Gustav Robertson reports a relationship with Westinghouse Electric Sweden that includes: non-financial support. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.



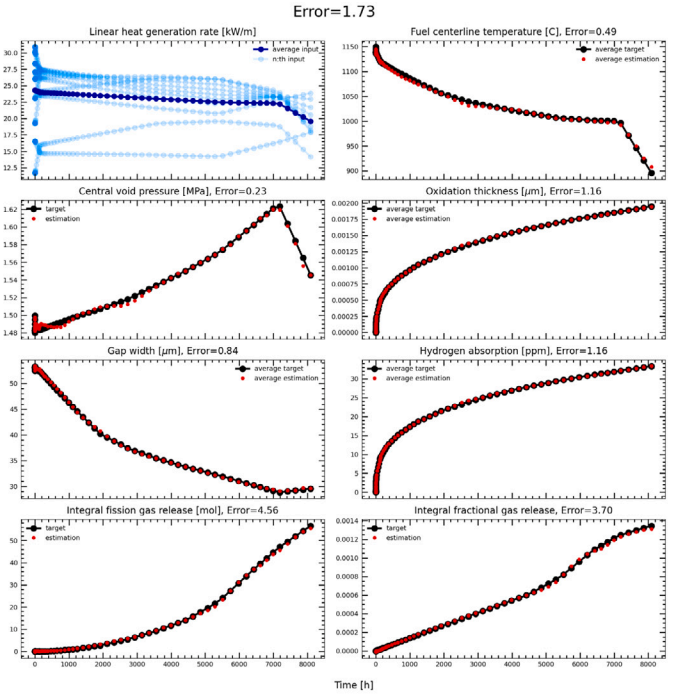
(a) A typical three-cycle LHGR operation with a test error close to the median.



(b) A typical one-cycle LHGR operation with a test error close to the median.

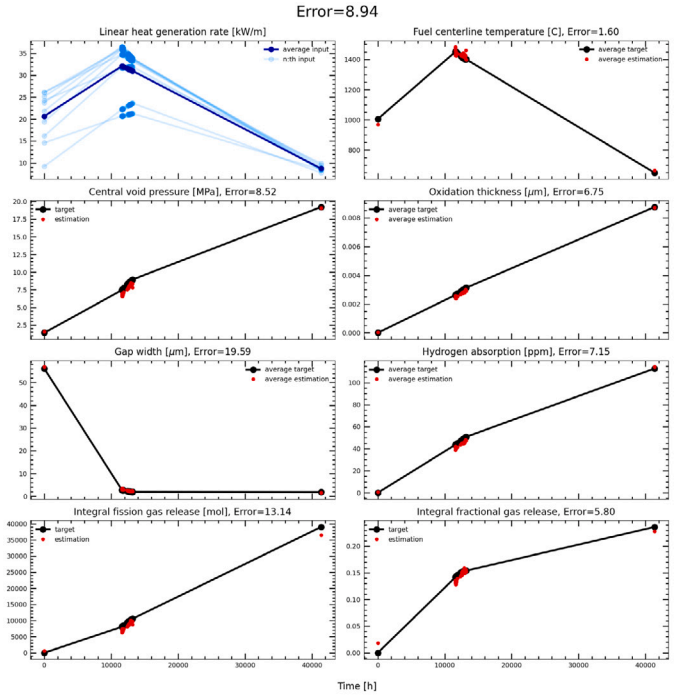


(c) A typical five-cycle LHGR operation with a test error close to the average.

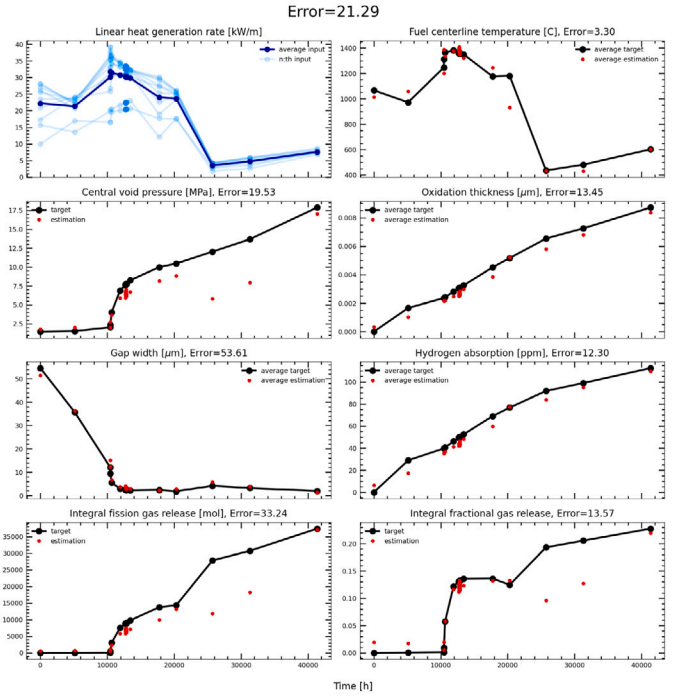


(d) A typical one-cycle LHGR operation with a test error close to the average.

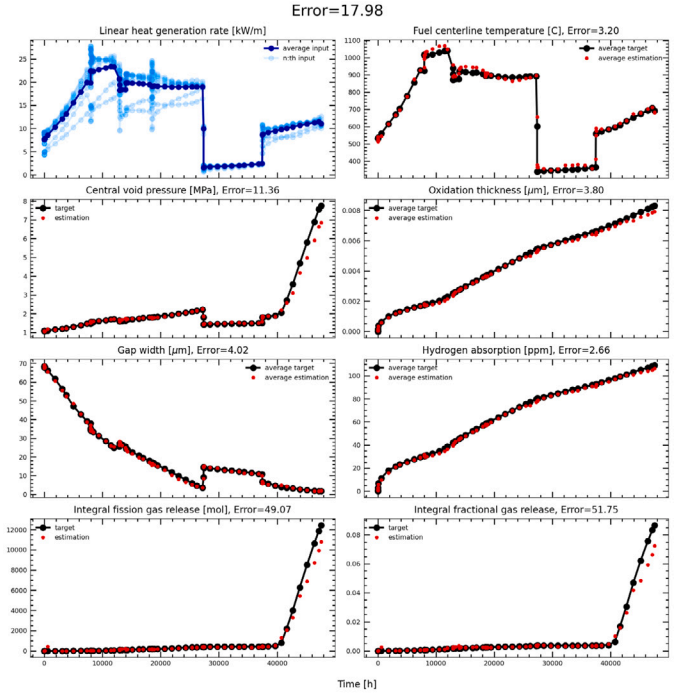
Fig. 10. The figure demonstrates inputs regarding LHGR and outputs from Transuranus and the TFN-v2 model. These examples represent the model's average performance as shown in the test error distribution in Fig. 9. The outputs are fuel center line temperature, rod internal pressure, cladding oxide layer thickness, gap width, hydrogen pick-up, and the absolute and relative amount of fission gas release. We have only shown the average outputs for visualization purposes.



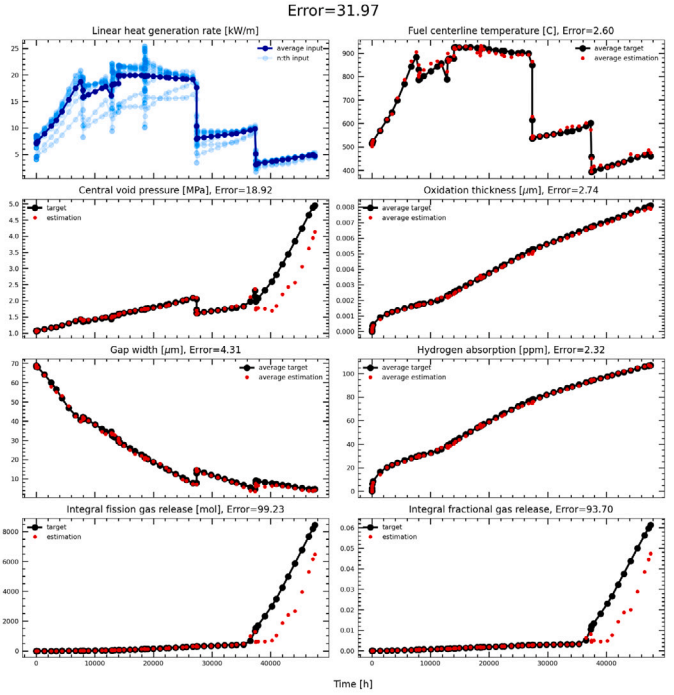
(a) A typical unrealistic case.



(b) A typical unrealistic case.



(c) A typical rod with burnable absorber with large error but has a visually good fit.



(d) The is the test sample with the largest error, which represents a typical rod with a burnable absorber.

Fig. 11. The figure demonstrates inputs regarding LHGR and outputs from Transuranus and the TFN-v2 model, where the obtained test losses were high. The outputs are fuel center line temperature, rod internal pressure, cladding oxide layer thickness, gap width, hydrogen pick-up, and the absolute and relative amount of fission gas release. We have only shown the average outputs for visualization purposes.

Data availability

Data will be made available on request.

Acknowledgments

This research acknowledges funding from the Swedish Centre for Nuclear Technology, SKC, an organization to which the authors are thankful. In addition, we would like to thank Westinghouse Electric Sweden AB for providing support and the power histories used for generating the training data. In addition, we express our gratitude to the Joint Research Centre (JRC) at the European Commission in Karlsruhe for delivering the fuel performance code Transuranus.

References

Bai, S., Kolter, J.Z., Koltun, V., 2018. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. URL arXiv:1803.01271 [cs].

Chi, L., Jiang, B., Mu, Y., 2020. Fast Fourier convolution. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M.F., Lin, H. (Eds.), In: Advances in Neural Information Processing Systems, vol. 33, Curran Associates, Inc., pp. 4479–4488, URL https://proceedings.neurips.cc/paper_files/paper/2020/file/2fd5d41ec6cfab47e32164d5624269b1-Paper.pdf.

Fort, S., Hu, H., Lakshminarayanan, B., 2020. Deep ensembles: A loss landscape perspective. URL arXiv:1912.02757 [cs, stat].

Lassmann, K., 1992. TRANSURANUS: a fuel rod analysis code ready for use. J. Nucl. Mater. 188, 295–302. http://dx.doi.org/10.1016/0022-3115(92)90487-6, URL https://www.sciencedirect.com/science/article/pii/0022311592904876.

Lassmann, K., Blank, H., 1988. Modelling of fuel rod behaviour and recent advances of the transuranus code. Nucl. Eng. Des. 106 (3), 291–313. http://dx.doi.org/10.1016/0029-5493(88)90292-0, URL https://linkinghub.elsevier.com/retrieve/pii/0029549388902920.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., Anandkumar, A., 2021. Fourier neural operator for parametric partial differential equations. arXiv:2010.08895 [cs, math].

Liu, Z., Ma, Q., Ma, P., Wang, L., 2023. Temporal-frequency co-training for time series semi-supervised learning. Proc. AAAI Conf. Artif. Intell. 37 (7), 8923–8931. http://dx.doi.org/10.1609/aaai.v37i7.26072, URL https://ojs.aaai.org/index.php/AAAI/article/view/26072.

Loshchilov, I., Hutter, F., 2017. Fixing weight decay regularization in Adam. CoRR URL arXiv:1711.05101.

Magni, A., Del Nevo, A., Luzzi, L., Rozzia, D., Adorni, M., Schubert, A., Van Uffelen, P., 2021. The TRANSURANUS fuel performance code. pp. 161–205. http://dx.doi.org/10.1016/B978-0-12-818190-4.00008-5.

Oord, A.v.d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., Kavukcuoglu, K., 2016. WaveNet: A generative model for raw audio. URL arXiv:1609.03499 [cs].

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S., 2023. torch.nn.Conv1d - PyTorch documentation2024-05-01. URL https://pytorch.org/docs/stable/generated/torch.nn.Conv1d.html.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S., 2024. torch.nn.functional.interpolate - PyTorch documentation2023-04-03. URL https://pytorch.org/docs/stable/generated/torch.nn.functional.interpolate.html.

Rippel, O., Snoek, J., Adams, R.P., 2015. Spectral representations for convolutional neural networks. Adv. Neural Inf. Process. Syst. 28, URL https://proceedings.neurips.cc/paper/2015/file/536a76f94cf7535158f66cfbd4b113b6-Paper.pdf.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., 2014. Going deeper with convolutions. URL arXiv:1409.4842 [cs].

Van Uffelen, P., Hales, J., Li, W., Rossiter, G., Williamson, R., 2019. A review of fuel performance modelling. J. Nucl. Mater. 516, 373–412. http://dx.doi.org/10.1016/j.jnucmat.2018.12.037, URL https://www.sciencedirect.com/science/article/pii/S0022311518310298.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I., 2023. Attention is all you need. URL arXiv:1706.03762 [cs].

Wang, Z., Liang, Z., Zeng, R., Yuan, H., Srinivasan, R.S., 2023. Identifying the optimal heterogeneous ensemble learning model for building energy prediction using the exhaustive search method. Energy Build. 281, 112763. http://dx.doi.org/10.1016/j.enbuild.2022.112763, URL https://www.sciencedirect.com/science/article/pii/S0378778822009343.

William Beckner, 1975. Inequalities in Fourier analysis. Ann. of Math. 102, 159, URL https://api.semanticscholar.org/CorpusID:123174610.

Yang, L., Hong, S., 2022. Unsupervised time-series representation learning with iterative bilinear temporal-spectral fusion. URL arXiv:2202.04770 [cs].

Yi, K., Zhang, Q., Cao, L., Wang, S., Long, G., Hu, L., He, H., Niu, Z., Fan, W., Xiong, H., 2023. A survey on deep learning based time series analysis with frequency transformation. URL arXiv:2302.02173 [cs].