Uppsala University

Bachelor's thesis in physics - 15 HP 1FA599

Deep Neural Networks as Surrogate Models for Fuel Performance Codes

Department of applied nuclear physics

Author: Wenhan Zhou
Email: wenhan.zhou.2955@student.uu.se

Project Supervisor: Gustav Robertson
Subject reviewer: Henrik Sjöstrand
Examiner: Matthias Weiszflog

August 5, 2023

Abstract

The core component of a nuclear power plant is the reactor and the fuel rods that supply it with fission fuel. Efficient and safe energy extraction is thus highly dependent on the reactor design and the conditions of the fuel rods. To anticipate high-quality operation and potential risks in advance, one must perform simulations on the fuel rods.

This is traditionally executed using fuel performance codes such as the Transuranus [1] and FRAPCON [2]. These codes offer intricate and accurate models of the underlying physical processes that govern fuel rod performance, encompassing aspects like linear heat generation rates, neutron flux and their irradiation effects, fuel rod expansion and contraction, clad corrosion, and fission gas release. However, fuel performance codes cannot be easily parallelized using modern hardware accelerators, such as graphic processing units, due to their iterative nature that follows from the complexity of coupling multiphysics-based models. Thus, they can only simulate one fuel rod at a time per CPU. The challenge comes when trying to simulate all the 10,000 to 100,000 fuel rods in a typical pressurized water reactor [3, 4], which is a relatively slow process as compared to parallelized computation for all time steps using a GPU.

To improve the speed aspect of the fuel performance modeling, a data-driven surrogate model based on neural networks is developed. The main advantage of a neural network over fuel performance codes is its ability to perform parallelized computations using one or more GPUs. The preexisting architectures that were explored include Temporal Convolutional Network [5], Fourier Neural Operator [6, 7], and a Transformer [8]. Additionally, a novel architecture is proposed, the Temporal Frequency Network. This is a heterogeneous ensemble method that is based on the Temporal Convolutional Network and the Fourier Neural Operator. The newly proposed architecture archives the lowest validation error among the preexisting architectures with a minor increase in the computations as compared to the ensemble components.

The Temporal Frequency Network is then applied to take time-dependent inputs in the form of linear heat generation rate and use it as the only information to predict various time-dependent variables of the fuel rods[1]. The predicted variables include fuel centerline temperature, central void pressure, oxidation thickness, fuel gap width, hydrogen absorption, integral fission gas release, and integral fractional gas release.

When deploying the neural network in practice, the user cannot in general trust that the model will generalize from its training, especially in fuel performance modeling where accurate predictions are important to demonstrate safe operation. To ensure that the predictions of the model are reliable, a separate neural network called *decoder* is trained to qualitatively quantify the error of the previous model that made the predictions, called *encoder*. This is done by training the decoder to reconstruct the original input to the encoder by providing it with the output, e.g., the inverse task. It is then possible to compare the original input with the reconstructed input, thus, an error can be computed that can be used to qualitatively determine the quality of the predictions.

With the Temporal Frequency Network, the average validation error was roughly 1% error. This makes it a strong candidate surrogate model for fuel performance modeling. In addition, with the encoder-decoder setup, this work provides a robust framework for error estimation that can be performed on new, potentially out-of-distribution inputs without the need for a fuel performance code. This setup is therefore suitable for applications where a low false negative rate is desired.

[1]There were several other parameters that were not included because they were fixed during the training data generation.

Sammanfattning

Huvudkomponenten i ett kärnkraftverk är reaktorn och bränslestavarna som förser den med kärnbränsle. Effektiv och säker energiextraktion är därför starkt beroende av reaktorns utformning och bränslestavarnas skick. För att förutse högkvalitativ drift och potentiella risker i förväg måste man utföra simuleringar på bränslestavarna.

Detta utförs traditionellt med bränslestavskoder som Transuranus [1] och FRAPCON [2]. Denna uppgift utförs traditionellt med bränsleprestandakoder som Transuranus [1] och FRAPCON [2]. Dessa koder tillhandahåller detaljerade och exakta modeller som modellerar bränslestavens fysikaliska beteende. Modelleringen i denna arbete omfattar aspekter som linjär längdvärmebelastning, snabbneutronflöde och deras bestrålnings effekter, expansion samt kontraktion av bränslestaven, kapslingskorrosion och utsläpp av fissionsgas. Emellertid kan dessa bränslestavskoder inte enkelt parallelliseras med moderna hårdvaruacceleratorer som grafikprocessorer på grund av deras iterativa karaktär som följer från komplexitiviteten med att modellera multiskaliga system med kopplade interaktioner mellan variablerna. Koderna kan således endast simulera ett bränslestav åt gången per CPU. Utmaningen uppstår när man försöker simulera alla 10,000 till 100,000 bränslestavar i en typisk tryckvattenreaktor [3, 4], vilket är en relativt tidskrävande process i jämnförelse till paralelliserad beräkning för alla tidssteg genom att använda en GPU.

För att reducera tiden för bränslestavsmodelleringen utvecklades en datadriven surrogatmodell som är baserad på neurala nätverk. Den viktigaste fördelen som ett neural nätverk har över en bränslestavskod är dess förmåga att utföra parallelliserade beräkningar genom att använda en eller flera GPU:er. De befintliga arkitekturerna som utforskades inkluderar Temporala faltningsnätverk [5], Fourier neurala operatorn [6, 7] och en Transformer [8]. Dessutom föreslås en ny arkitektur, Temorala frekvensnätverk. Arkitekturen är en heterogen ensemblemetod som bygger på Temporala faltningsnätverk och Fourier neurala operator. Den nyligen föreslagna arkitekturen uppnår det lägsta valideringsfelet bland de föreslagna arkitekturerna med en mindre ökning av antal beräkningar i jämnförelse med dess individa ensemblekomponenter.

Temporala frekvensnätverket användes sedan för att ta tidsberoende indata i form av linjära längdvärmebelastning och använda den som den enda informationen för att förutsäga olika tidsberoende variabler hos bränslestavarna[2]. De förutsagda variablerna inkluderar bränslets temperatur, trycket hos den fria volymen, oxidationstjocklek, bränslegapsbredd, väteabsorption hos kapslingen, ackumulerad generad fissiongas och totalt relativt fissiongasutsläpp.

När man implementerar det neurala nätverket i praktiken kan användaren i allmänhet inte lita på att modellen kommer att generalisera från sin träning, särskilt i bränslestavsmodellering där förutsägelser med hög noggrannhet är viktiga för att demonstrera säker drift. För att säkerställa att förutsägelserna är tillförlitliga tränas ett separat neuralt nätverk för att vara en *avkodare* vars uppgift är att kvantifiera kvaliteten på förutsägelser som görs av den tidigare modellen, *kodaren*. Detta uppnås genom att träna avkodaren på att lära sig funktionen som avbildar utdatan till indatan. På så vis kan den ursprungliga indatan jämnföras med den rekonstruerade indatan, och ett fel kan beräknas för att kvalitativt avgöra kvaliten hos prediktionerna av kodaren.

Med Temporala frekvensnätverket hölls det genomsnittliga valideringsfelet vid cirka 1%, vilket gör det till en stark kandidat för bränslestavsmodellering. Dessutom, med uppställningen för kodare-avkodare, erbjuder detta arbete en robust utgångspunkt för felförutsägelse som kan utföras på nya indata som potentiellt är urvald bortom fördelningen för träningsdatan utan att behöva använda en bränslestavskod. Denna uppställning är därför lämplig för tillämpningar där låg andel falska negativa fall är önskvärd.

[2]Det fanns flera andra parametrar som inte inkluderades eftersom de var fixt under generering av träningsdatan.

Contents

Nomenclature

$\mathbf{x}$	Input
$\mathbf{x}'$	Input Reconstruction
$\mathbf{y}$	Output
$\mathcal{L}(\cdot,\cdot)$	Objective Function
ε	Reconstruction Error
φ	Encoder
φ^{-1}	Decoder
Adam	Adaptive Momentum Estimation
BC	Boundary Condition
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DL	Deep Learning
FFC	Fast Fourier Convolution
FFT	Fast Fourier Transform
FN	False Negative
FNO	Fourier Neural Operator
FP	False Positive
FS	Fourier Series
GPU	Graphical Processing Unit
ITFT	Improved Temporal Fusion Transformer
LHGR	Linear Heat Generation Rate
LSTM	Long Short-Term Memory
NLP	Natural Language Processing
NRMSE	Normalized Root Mean Squared Error
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network
TCN	Temporal Convolutional Network
TFCNN	Temporal Frequency Convolutional Neural Network
TFN	Temporal Frequency Network
TFT	Temporal Fusion Transformer
TN	True Negative
TP	True Positive
TU	Transuranus

1 Introduction

Nuclear energy has become an important part of global electricity generation, providing a consistent and low-carbon source of electricity. By harnessing the power of nuclear fission, which involves splitting heavy atomic nuclei such as Uranium isotopes, a significant amount of energy is released according to Einstein's equation $E = mc^2$ where E is the energy released, m is the mass defect before and after fission, and c is the speed of light. This energy can be converted to electricity in nuclear power plants. Despite its benefits, the nuclear industry faces strict regulations due to the potential risks associated with accidents and nuclear waste management. To maximize the energy extraction from the fission process and mitigate associated risks, the design and development of new power plants and their components will play a critical role.

One of the most crucial components of a nuclear power plant is the nuclear fuel rod. This is a cylindrical component that is used in nuclear reactors to contain nuclear fuel. During operation, the fuel rods provide the reactor with fissile fuel allowing for fission to occur. It is the fuel rods that provide the fluid and sub-sequentially the steam turbines with the necessary energy that can then be converted into electrical energy. The fuel rod is also the first barrier for containing fission products.

During the operation of the nuclear power plant, various behavior of the fuel rods may occur, some of which are related to high temperature, pressure, and deformations. In extreme cases, accidents may occur. To prevent any potential accidents, one needs to perform a conservative simulation of the various time-dependent variables of the fuel rods. This is to ensure that the design choice can be applied in practice. Another safety mechanism is to collect experimental measurements during operation and use it as input such that the future states of the fuel rods can be predicted. Real-time regulations may be employed if any simulation proceeds the safety criterion.

The simulation process involving the nuclear fuel rod is known as *fuel performance modeling*. In essence, fuel performance modeling is about understanding and predicting the behavior and life cycle of nuclear fuel rods, see section 3 for more details.

From the applied nuclear physics perspective, it involves understanding the role of neutron flux and LHGR in the fission process and their impact on components, such as fuel rods and the cladding surrounding them.

Furthermore, one needs to consider the fission-related bi-products such as fission gases like xenon and krypton. This could, for instance, affect the thermal conductivity and causes changes in pressure within the rod. Another factor related to thermal conductivity is the heat transfer outside of the fuel rod. This is a crucial factor as properties of the coolant, such as type of coolant and flow rate, may influence the optimal working condition of the fuel rods by controlling the working temperature.
From a mechanical engineering perspective, fuel performance modeling addresses the mechanical stresses and strains placed on the fuel rods due to temperature changes, radiation damage, and other operational conditions. This is particularly important for maintaining the integrity of the fuel rod cladding.
Finally, fuel performance modeling must consider the impact of chemical phenomena on the cladding itself. Examples of such processes involve oxidation and hydrogen absorption which can change the thermal conductivity as well as the mechanical behavior of the cladding. By combining all these elements, fuel performance modeling provides a comprehensive tool for understanding and predicting the intricacies of nuclear fuel rod operation, informing the design, optimization, and safety protocols for nuclear reactors.

In recent times, advances in GPUs and DL have introduced a new paradigm for many scientific modeling applications. DL is a sub-field of machine learning that employs a data-driven methodology to construct models. While explicit knowledge of the interactions between variables in the system of interest is not required, the models are capable of learning the underlying relationships that link inputs to outputs. This is an optimization process that involves minimizing the error between the model prediction and the ground truth data. As a result of minimizing the error, the parameters of the model must capture the underlying principles of the system.

The most popular approach in DL is to use deep neural networks to adapt their model parameters to the data. Neural networks are especially suitable for modeling systems where data can be easily obtained and complex interactions that cannot be easily modeled by equations derived from using first principles. Additionally, computations can be efficiently parallelized by GPUs once the neural network is successfully trained. This is particularly suitable for fuel performance modeling as fuel performance codes cannot be easily parallelized with GPUs due to dynamic changes in the boundary conditions that cannot be known in advance without performing the iterative computation, see section 3 for more detail of the fuel performance code.

There are various different neural network architectures, each having its own domain of application. In fuel performance modeling, the choice of architecture must consider causal time relationships in the data, i.e. a prediction must only depend on the current and past inputs. Some common architectures that incorporate the time relationships are RNN [9], TCN [5], and Transformers [8].

In previous work, Westinghouse Electric Sweden has trained several RNNs to predict the probability of pellet-cladding interaction given a set of LHGR as input [10]. However, several different RNNs were needed to handle various special cases, which made it difficult to use in practice.

Another related work done by V.Nerlander was to use a TCN to predict the time evolution of cladding oxidation thickness that was generated using a simplified oxidation model [11]. According to the preliminary analysis, TCN demonstrated excellent long-term memory capabilities making it suitable for working with arbitrary time-dependent data. Moreover, due to its fast performance and straightforward implementation, TCN seems to be a highly promising option for fuel performance modeling.

In this work, I work with neural networks to find a surrogate model for fuel performance codes. Specifically, I take inspiration from the TCN architecture and ideas from a more recently proposed architecture FNO [6, 7] and propose a novel ensemble architecture TFN that is more powerful than its individual components. The TFN is then trained on mapping real-world LHGRs to several time-dependent outputs that are generated by TU. To ensure the predictions are reliable, I implement the autoencoder architecture to perform zero-shot outlier detection. Furthermore, I evaluate the Transformer architecture and discuss how it can be used to achieve a more informed prediction. The code is open-sourced at: https://github.com/SunAndClouds/Examensarbete

2 Nuclear Fuel Rods and Thermal-mechanical Behavior

At the core of a nuclear reactor lies the process of nuclear fission, where the fission fuel is supplied by the nuclear fuel rods. Each fuel rod comprises a series of pellets made of UO_2 stacked on top of each other. The fuel is then enclosed within a cylindrical cladding made of zirconium alloys to protect it from damage [12].

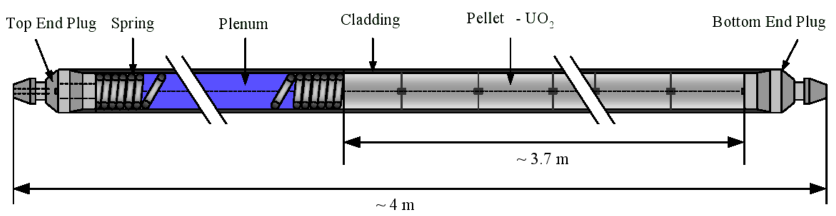


Figure 1: Illustration of a nuclear fuel rod [13] with permission from the author. Note that in reality there may be a gap between the fuel and the cladding which may change with respect to time as the mechanical properties of the fuel rod change.

When a neutron strikes the nucleus of these fissile atoms, the nucleus splits, releasing a large amount of energy in the form of kinetic energy, and various types of radiation. Simultaneously, this fission process releases additional neutrons, which causes further fission by interacting with neighboring fuel rods.

Furthermore, some UO_2 fuel rods may contain some Gadolinium, which is an element that can absorb neutrons. Hence, these types of mixed fuel rods behave differently than pure UO_2 fuel rods.

In a reactor environment, the fuel rods are not behaving statically, they are subjected to changes induced by various factors. Several factors may contribute to changes to the geometry of the rods, such as heat produced during fission, the mechanical stress caused by the pressure within the fuel rods, and the effects of fission products, etc. The complex interplay of these factors is known as the *thermo-mechanical behavior* of the fuel rod. Over time, these effects can cause the fuel rod to deform.

The deformations can mainly be divided into two types, elastic- and plastic deformations. The former describes relatively small changes in the geometry that can be reversed if the related physical quantity is reversed. The latter is a change in geometry that corresponds to an irreversible process. In more severe cases, plastic deformation can come in fuel cracking, a common undesirable behavior.

3 Fuel Performance Modeling

Modeling fuel rod dynamics is a complex task that demands an advanced methodology. In traditional practice, fuel performance codes are utilized. In this particular work, the TU code is employed. This code works by solving hundreds of equations that model the fuel rod dynamics, then advancing in time according to various input parameters. Two such critical inputs are the neutron flux, describing the number of neutrons passing through a given area per unit of time, and the LHGR, which provides a measure of the fuel rod's power production per unit length. When the solution does not converge according to some predefined threshold, the TU inserts an intermediate point in time[3] and evaluate the output variables at the intermediate point, this is repeated until convergence. By doing so, TU achieves a solution with a higher resolution than the original time-dependent input that takes the dynamic boundary conditions of the fuel rod system into consideration.

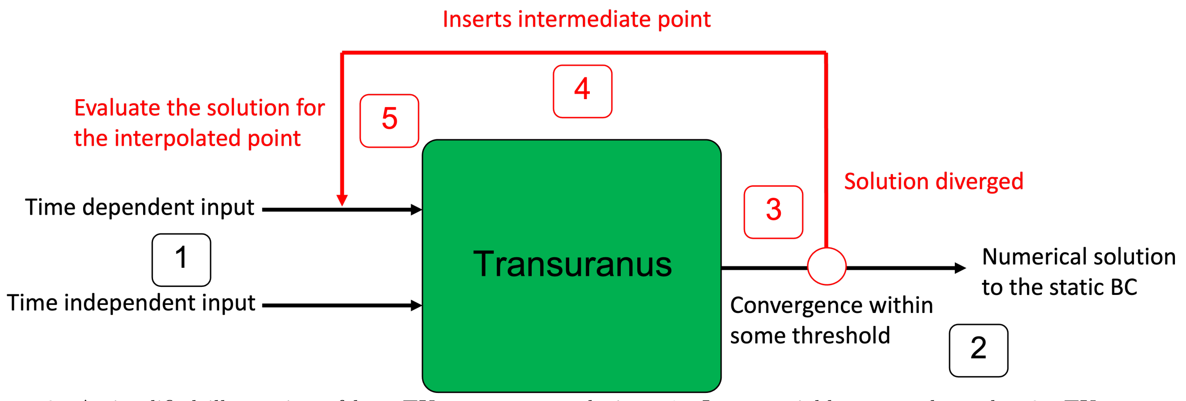

Inserts intermediate point
Evaluate the solution for the interpolated point
5
4
3
Solution diverged
Time dependent input
1
Time independent input
Transuranus
Numerical solution to the static BC
Convergence within some threshold
2

Figure 2: A simplified illustration of how TU generates a solution. 1. Input variables enter the code. 2. TU computes a numerical solution subjected to a static BC. 3. Convergence criterion. 4. Insert an intermediate point if the solution diverged. 5. Evaluate the solution at the interpolated point.

The neutron flux, while central to the operation of the reactor and crucial for estimating the LHGR, also interacts with other components of the fuel rod. Specifically, the zirconium alloy cladding is subject to a high neutron flux during reactor operation. As these neutrons interact with the cladding, they can lead to several phenomena that may potentially harm the integrity and functionality of the fuel rods:

[3]This is known as the *micro time step*. The original time points are known as the *macro time steps*.

- Neutron-induced embitterment corresponds to a phenomenon that occurs as a result of neutron-cladding interaction. The material properties of the cladding are weakened such that the probability of cracking is increased [14].
- Neutron-induced swelling is a phenomenon that occurs when the geometry properties of the cladding are changed. This corresponds to a change in boundary condition which may directly affect the mechanical behaviors of the cladding.
- The neutron cladding interaction may also affect the rate at which corrosion occurs. In fact, there is a strong correlation with the thickness of oxidation, which is further explored in the coming section.

The neutron flux and the LHGR play a critical role as inputs in the modeling of fuel rod dynamics using the TU code. Attention now turns to the results of these inputs, which will be referred to as the output variables. These outputs unveil the intricate inner workings of the fuel rod, including the creation of fission gases, changes in pressure, and impacts on the cladding, like oxidation and hydrogen absorption. These variables are explored in the following sections.

3.1 Fission Gases and Pressure

During operation, bi-products from the fission will be produced; one common type of phenomenon that is important to model for accurate prediction of the next state of the system is the fission gas release. The most important fission gases are xenon and krypton, they come in bubbles created in the grains of the fuel pellets during the fission process. The total amount of gas that is released throughout the operation is known as *integral fission gas*. Since the fuel pellets are mainly made of UO_2, the fission gases do not easily dissolve. Instead, they diffuse to the fuel gap, between the surface of the pellets and the cladding. This process changes the thermal conductivity of the gap. This creates a feedback loop that changes the thermal properties of the rod. Another gas that is produced during operation, but not as a consequence of fission, is hydrogen. One of the causes is due to the ionization of water molecules.

There are unavoidable, small manufacturing defects in the fuel rod that leads to small vacancies within the material. During operation, as fission gas is created, the gases will fill these vacancies and form bubbles. At some point, these bubbles may merge together and/or form gas tunnels which result in a fission gas release. The fuel rod may also undergo fuel cracking due to thermo-mechanical deformations. These two phenomena will both contribute to fission gas being released into the fuel gap as well as to the *plenum* as shown in the figure 1. The pressure in the plenum is known as the *central void pressure*[4].

3.2 Cladding Oxidation and Hydrogen Absorption

The cladding material is made of a zirconium-based alloy. This metal has the property to react with water (that is used as a coolant) through:

$$2H_2O + Zr \rightarrow ZrO_2 + 4H^{\cdot} \tag{1}$$

where O is the oxygen, Zr is the zirconium and $H^{\cdot}$ is the hydrogen radical [15]. There are several phases during operation at which the rate of growth changes its characteristics. During the early stages of operation, the thickness of the oxidation is very thin. The main contributing factor to the oxidation thickness growth will thus correlate to the reaction rate of (1).

As the thickness grows, the reaction-driven oxidation transforms into diffusion-driven oxidation growth. This is because oxidation shields the chemical reaction and the main contributing factor to oxidation is caused by reactant (oxygen) diffusing through the oxidation.

[4]Since the fuel gap and the plenum is connected, the rod internal pressure and the central void pressure are equivalent.

The diffusion process has a parabolic characteristic rather than the linear characteristic of reaction-driven oxidation. Then a more complex process, called *post-transition regime* takes over, which has a pseudo-linear growth [14].

Although the oxidation grows rather slowly, it will consume the cladding converting it to zirconium oxide. The hydrogen radicals of the reaction will for the most part combine and form hydrogen molecules which are then dissolved in the water. A limited amount of hydrogen will penetrate the oxide to reach the metallic matrix where it is resolved. Another possibility is that it interacts with zirconium to form hydrides [15]. In conclusion, the cladding will slowly undergo corrosion, this process will simultaneously increase the oxidation thickness of the cladding which changes the physical behavior of the cladding. The most important one is the heat conductivity of the cladding.

4 Data-Driven Science

For a long time, the scientific method has been proposing a hypothesis based on first principles that aim to capture the underlying behavior of a system. The hypothesis is then challenged by repeated experiments. A hypothesis will be accepted as a theory if it provides accurate predictions that generalize under various independent scenarios. This particular method is said to be theory-driven. This approach works well when the underlying phenomenon is relatively unaffected by other (hidden) variables and their interactions. However, this means that an accurate model will be increasingly difficult when non-linear interactions occur between various variables. Due to the growing accessibility of enhanced computational power, data-driven methodology has surfaced as an evolving paradigm in scientific research [16].

In contrast to theory-driven methodology, the data-driven methodology focuses on first collecting data that accurately represent the dynamics of the system, then through prior (in this case, physical) knowledge, designing a statistical model that is able to model the desired dynamics of the system. For instance, in a curve-fitting problem with a given dataset, there may be several functions that offer a good fit. However, with physics knowledge of the system, some architectural designs are more appropriate than others. This prior knowledge, known as inductive biases, helps the model to effectively generalize when it comes across unfamiliar data [16].

In practice, one constructs an over-parameterized model with physical constraints considered, such as geometrical symmetries, causal relationships in the data, and conservation laws, and applies an optimization procedure to let the model parameters adapt to the data. The optimization process is often referred to as fitting or training. In the context of machine learning, which is a data-driven discipline, the data that is used to achieve this optimization procedure is often referred to as the *training data*. The model essentially finds a set of parameters on its own that best represents the training data. This is particularly useful when trying to model a complex system with many interacting variables, such as fuel performance modeling [17].

However, the success of data-driven methodology largely depends on the quality and representativeness of the available data. As a general rule of thumb, if the model is fed with bad data, the predictions will be at best, as bad as the training data. In the same way, the upper limit of the predictive accuracy and precision of the model is limited by the noise present in the training data. The data-driven approach serves as a surrogate model, rather than a substitute for the original source responsible for generating the data, which in this work, is the TU code.

Neural networks are examples of powerful data-driven models. They are constructed by a large number of parameters, known as *weights* and *biases*. A subset of the parameters forms *layers*, which corresponds to a transformation of the data. To learn non-linear transformations, one needs to apply a non-linear activation function to the output of each layer[5]. Together, they form a network structure that has the ability to perform non-linear computations that resembles the activity of neurons in the brain, hence the name, neural networks [17].

The design choice of a neural network architecture involves using prior knowledge of the system by adding constraints to a generic neural network. In this work, the time causality of the data points will be considered as well as energy conservation when performing normalization techniques on the data. Additionally, the predicted result must follow fundamental conservation laws, in this case, the energy must be conserved.

During training, the weights will be updated to minimize the error. The error of the model on the training data is known as the *training loss*. When a neural network is trained, it acquires the ability to accurately represent the data it was trained on. However, there is no guarantee that the network will perform well on new, unseen data. If the input is slightly perturbed, the output may change significantly. This phenomenon is known as *over-fitting* and is a common unwanted bi-product of the training process[6]. To reduce the effects of over-fitting, a subset of the dataset will not be used to train the model, but rather use it as an evaluation metric. This subset of the data is referred to as validation data. If the network simultaneously performs well on the training data and the validation data, then there is a statistical significance that the network has learned to generalize patterns during its training [17].

5 Training Data Generation with Transuranus

This section describes the characteristics of the data that is passed to the TU as well as the training- and validation data. There were several input data types that were used for the TU, both static and dynamic variables. However, the most important one was the LHGR, the remaining input variables were fixed. This simplification was made to reduce noise in the training data and speed up the development process.

5.1 Input-files to Transuranus

1. A dataset consisting of 1416 samples of realistic LHGRs and neutron flux of each 32 fuel-pellet was compressed to 1416 samples with each sample consisting of 10 LHGRs. This step was done using linear interpolation in the vertical axial direction.

2. The 10 interpolated LHGRs were then normalized by their average values:

$$\text{LHGR}_{relative} = \frac{\text{LHGR}_{pellet}}{\langle \text{LHGR}_{pellet} \rangle}. \tag{2}$$

3. A weighted normalization procedure was then performed on the neutron flux Φ_{pellet}:

$$\Phi_{relative} = \frac{\Phi_{pellet}\text{LHGR}_{pellet}}{\langle \Phi_{pellet}\text{LHGR}_{pellet} \rangle}, \tag{3}$$

   this quantity can be shown to be equivalent to the relative LHGR.

4. The relative LHGRs, neutron flux, coolant- temperature, and pressure were used as inputs to the TU from which the corresponding outputs were generated.

[5] A neural network that does not have non-linear functions can be considered as a linear regression model.

[6] Over-fitting is not an exclusive phenomenon that occurs when training neural networks. It is a general phenomenon that may occur in regression tasks where the model is over-parameterized. One example is using a high-order polynomial to fit a line with some noise.

5.2 Data Specifications

The training data $\mathbf{x}$ is used as input to the neural network. The rows represent the time steps and the columns represent the different LHGRs. The structure of the data is given by:

$$\mathbf{x} = (\text{batch size}, \text{time steps}, \text{LHGR}_1, ..., \text{LHGR}_n).$$

The specific values used during training are:

- batch size $= 1200$ for training and 216 for validation.
- time steps $= 80$ for all data.
- $1 \leq n \leq 10$.

The target data $\mathbf{y}$ generated by TU have the following dimensions:

1. Time , $t \in \mathbb{R}^{\text{time steps} \times 1}$ h.
2. Linear heat generation rate , $Q \in \mathbb{R}^{\text{time steps} \times 10}$ kW/m.
3. Fuel centerline temperature, $T \in \mathbb{R}^{\text{time steps} \times 10}$ C.
4. Central void pressure $P \in \mathbb{R}^{\text{time steps} \times 1}$ MPa.
5. Oxidation thickness, $S \in \mathbb{R}^{\text{time steps} \times 10}$ μm.
6. Gap width, $L \in \mathbb{R}^{\text{time steps} \times 1}$ μm.
7. Hydrogen absorption, $H \in \mathbb{R}^{\text{time steps} \times 1}$ ppm.
8. Integral fission gas release, $G_1 \in \mathbb{R}^{\text{time steps} \times 1}$ moles.
9. Integral fractional gas release, $G_2 \in \mathbb{R}^{\text{time steps} \times 1}$.

They are structured in the following way

$$\mathbf{y} = (t, T, P, S, L, H, G_1, G_2).$$

Each of the quantities below is scaled by dividing each variable with the maximum value of the respective interval before it was passed into the neural networks.

1. Time $[0, 40375.19921875]$.
2. Linear heat generation rate $[0.0739375, 36.72453]$
3. Fuel centerline temperature $[295.2045, 1629.64913.61367]$.
4. Central void pressure $[1.019701, 13.61367]$.
5. Oxidation thickness $[3.560042 \cdot 10^{-6}, 8.526746 \cdot 10^{-3}]$.
6. Gap width $[0, 72.08205]$.
7. Hydrogen absorption $[0.06679402, 111.0238]$.
8. Integral fission gas release $[0, 168248.1]$.
9. Integral fractional gas release $[0, 0.1630313]$.

6 Data Preprocessing

The simulation output of TU may differ by the number of time steps for each run. This is a consequence of the TU inserting micro-time steps to achieve the necessary convergence criterion[7]. It can be as small as having 7 time steps and the largest has 91250 time steps. Therefore, it is not feasible to simply pad the data to the largest sequence. This is resolved by using a down-sampling technique to first reduce the length of the data, then re-inserting important data points that were filtered by the down-sampling algorithm, and finally, the processed data samples are interpolated such that the dimensions are consistent across all samples.

6.1 Down Sampling Alternatives

The LHGR in this dataset has the nature of being piece-wise linear. Additionally, it is common to observe discontinuous changes in the LGHR because of discontinuous positional between cycles. The discontinuous changes in the LHGR are highly correlated to many quantities. The most informative part of the data, therefore, lies close to the points of steep gradients of the LHGR. However, this constraint is not enough to filter away most of the 91250 points in the longest data sequence. I observed that TU tends to insert most of the micro time steps between a certain pair of points.

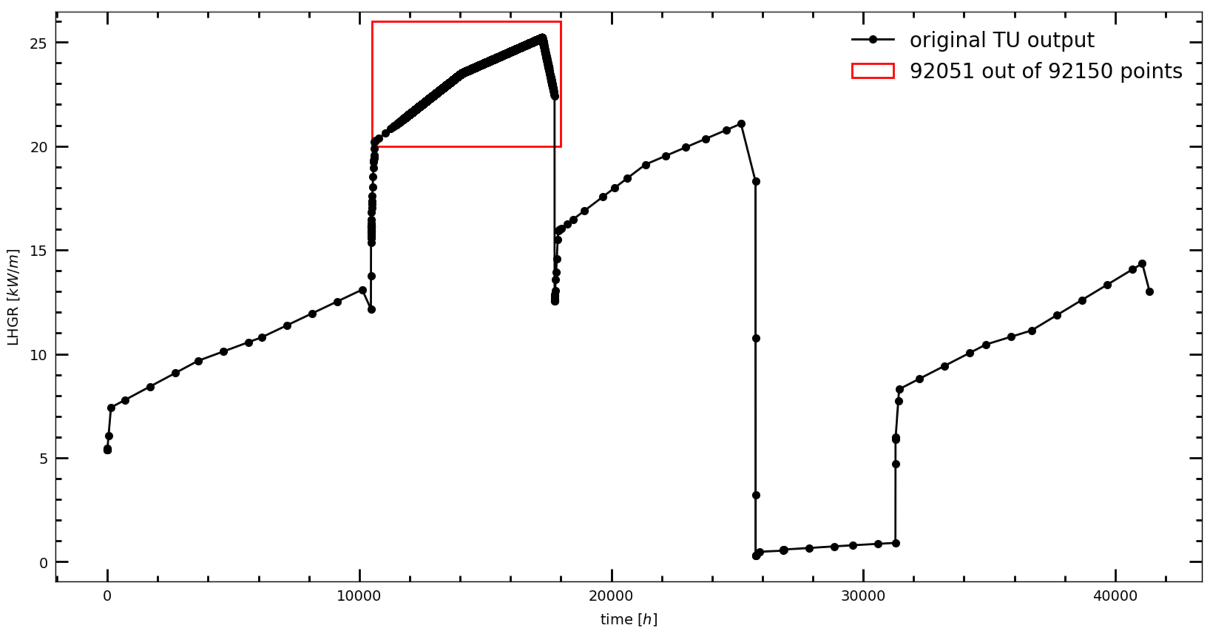


Figure 3: Large number of inserted micro time steps within a relatively small time interval.

To filter these densely inserted points, an additional constraint is proposed such that two neighboring points must have a sufficiently large change in the absolute magnitude of LHGR to be considered. The following equation defines a set of LHGR samples that fulfill specific conditions:

$$Q_{sampled} = \{Q|\Delta(\nabla Q) > 3 \times 10^{-4}, \Delta Q > 10^{-4}\}. \quad (4)$$

In this equation, $Q_{sampled}$ represents the set of LHGR samples that meet the two criteria. The set is composed of all LHGR samples Q for which the change in LHGR gradient, represented by $\Delta(\nabla Q)$, is greater than 3×10^{-4}, and the change in LHGR, represented by ΔQ, is greater than 10^{-4}.

However, these two constraints proved to be insufficient as too many points were discarded in a large time interval since the gradients might be close to constant with no significant absolute changes in the magnitude. While it is still representative of the LHGR, it is not sufficient for other physical quantities such as the fuel centerline temperature as it can behave very non-linear even when the LHGR is linear within that time interval. To reduce this effect, an additional functionality built on top of the gradient-based sampling algorithm is proposed, which works as follows:

[7]See Fuel Performance Codes

1. Compare the down-sampled sequence to the original sequence. Check if the down-sampled sequence has a shorter sequence length than the desired length.
2. If yes, compute the largest time interval in the down-sampled sequence and record the respective time points.
3. Check if there are any time points between the recorded ones in the original sequence.
4. If yes, pick a random element within the specified time interval, continue otherwise.
5. If 2. is no, pick the first desired time points from the down-sampled data.

After this part of the algorithm, the data will be more uniformly sampled but may have different lengths. To ensure consistency in the dimension of the data. Uniform linear interpolation was performed where the start- and final time steps are shifted by 1h to avoid overlapping.

In conclusion, the augmented training data consist of:

- Gradient-based sampling algorithm with absolute changes in LHGR as an additional constraint.
- Random insertion of micro time steps to the down-sampled data drawn from the original data.
- Linear interpolation based on the previously sampled points.

The figure below shows the sampled points for all physical quantities with the described strategy.

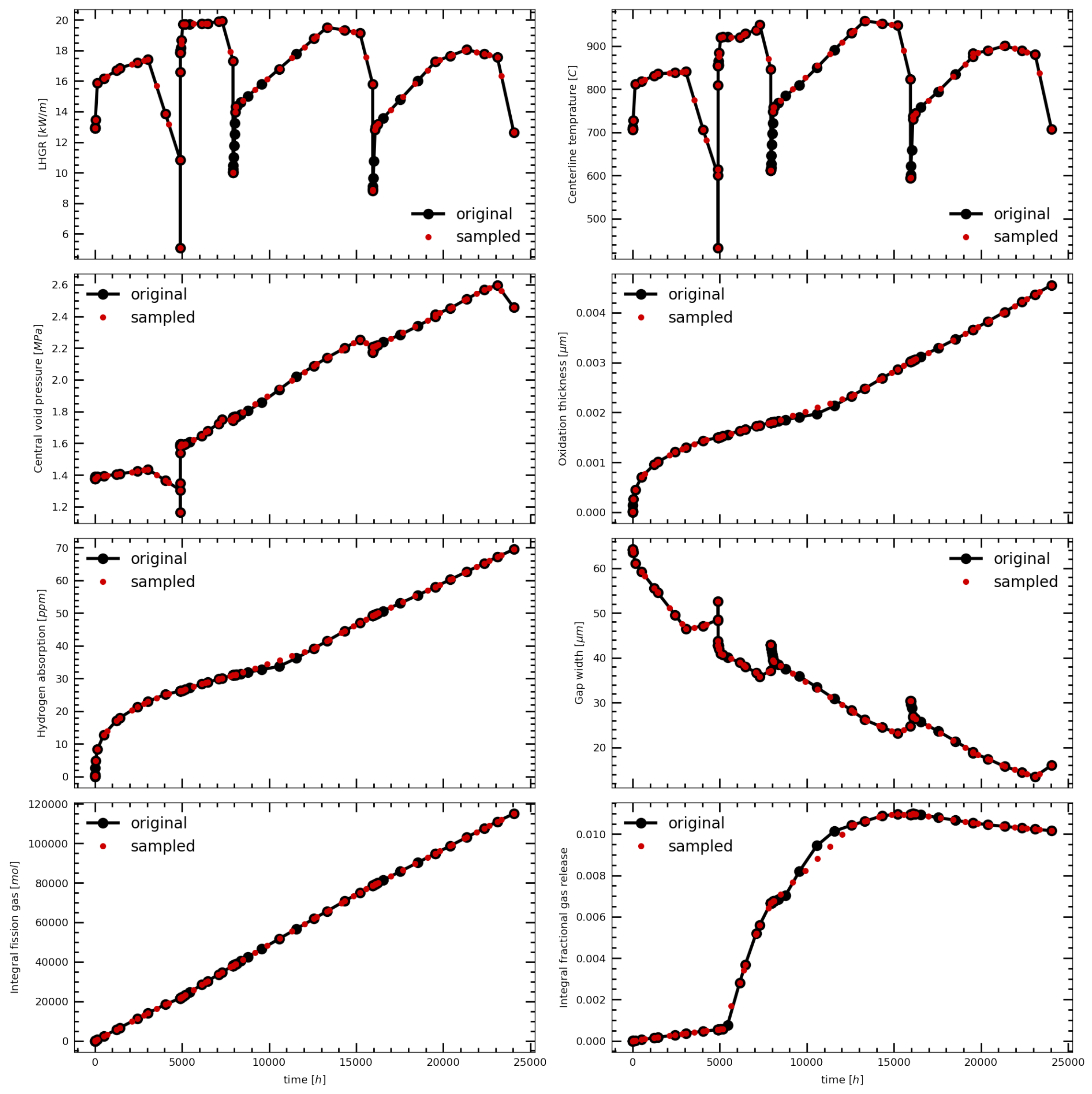


Figure 4: Black: Original output from TU. Red: Actual training data that is passed to the neural network. The sampled data largely overlaps with the original data in black, with some additional interpolated points included. All processed data have 80 points in total for training and validation data.

7 Neural Network Representations

The mathematical definition of a generic neural network is a function φ that maps an element $\mathbf{x}$ in the input space I to an element $\mathbf{y}$ in the output space O. The function φ is represented by recursion of affine transformations and element-wise non-linear transformation f. In other words, given some input $\mathbf{x}_1$, φ satisfies $\mathbf{y}_i = \varphi(\mathbf{x}_1)$ such that:

$$\begin{cases} \mathbf{x}_{i+1} = f(\mathbf{x}_i \mathbf{W}_i + \mathbf{b}_i) \\ \mathbf{y}_i = \mathbf{x}_{i+1} \end{cases}$$

for $1 \leq i \leq l$. For example, the simplest case of one layer ($l = 1$) neural network can be written as the linear regression with hyperbolic tangent as the non-linear activation function

$$\mathbf{y}_1 = \varphi(\mathbf{x}_1) = \tanh(\mathbf{x}_1 \mathbf{W}_1 + \mathbf{b}_1).$$

One can, in general, construct neural networks with an arbitrary number of layers that have been empirically shown to improve the quality of the predictions [17].

7.1 Training Process Description

The training objective for a generic neural network, φ, is to minimize the error between the predicted output versus the TU output. The *objective function*[8] $\mathcal{L}(\cdot,\cdot)$ is a function that measures the performance of φ with the dataset as reference. A low value of $\mathcal{L}$ refers to a low error between the prediction and the data. One common type of objective function that is used in regression tasks is the *root mean squared error* (RMSE):

$$\text{RMSE}(\varphi(\mathbf{x}), \text{TU}(\mathbf{x})) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\varphi_i(\mathbf{x}) - \text{TU}_i(\mathbf{x}))^2} \tag{5}$$

where the summation operates over the indices i of the samples of batches with size N. In practice, I found the NRMSE:

$$\text{NRMSE}(\varphi(\mathbf{x}), \text{TU}(\mathbf{x})) = \frac{100}{\frac{1}{N}\sum_{i=1}^{N}\text{TU}_\text{i}(\mathbf{x})} \cdot \text{RMSE}(\varphi(\mathbf{x}), \text{TU}(\mathbf{x})) \tag{6}$$

to be slightly more interpretable as it shows a percentage error, thus invariant under linear scaling of the data that was done in the pre-processing step. The objective function is a function of the parameterized neural network that contains weights and non-linear functions. As a result, the objective function is therefore differentiable. The gradient of the objective function with respect to the weights $\mathbf{W}$ at each layer l:

$$\nabla\mathcal{L} = \frac{\partial\mathcal{L}}{\partial\mathbf{W}_l} \tag{7}$$

provides the neural network the information on updating the weights to minimize the training objective. This is commonly done with the gradient descent algorithm:

$$\mathbf{W}_l^{n+1} = \mathbf{W}_l^{n} - \eta\nabla\mathcal{L} \tag{8}$$

where η is called the *learning rate*. In practice, a more sophisticated version of the gradient descent is used, namely the Adam algorithm [17, 18]. Training a neural network thus involves solving variants of equation (8) with an empirical number of iterations n and stops before the model overfits.

[8]Also known as the loss-function or cost-function

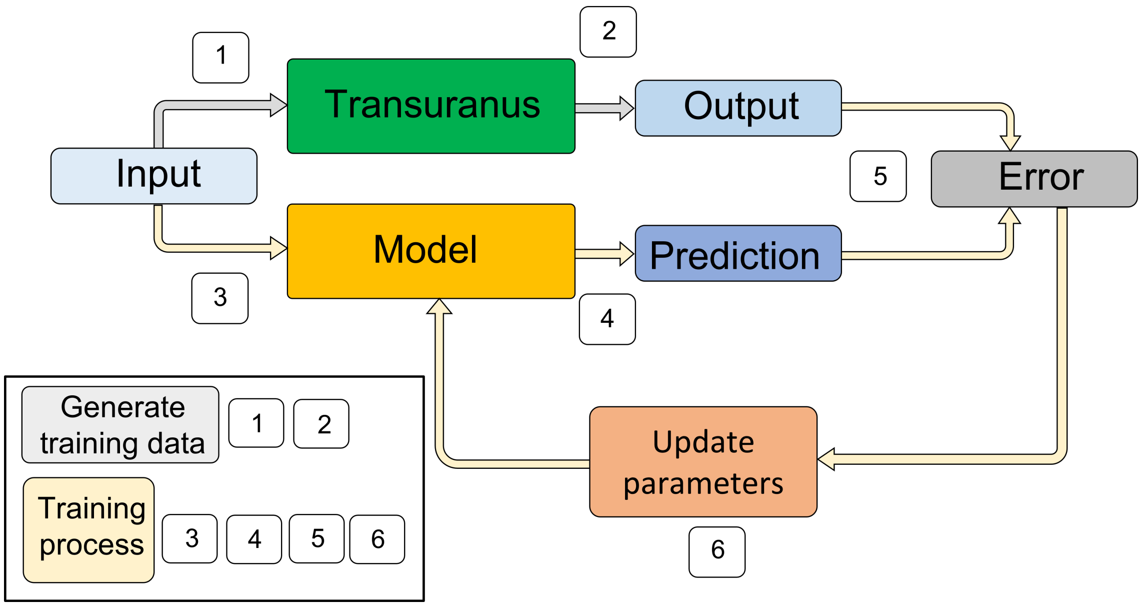


Figure 5: Illustration of how the neural network model is trained in conjunction with the Transuranus.

7.2 Temporal Convolution Network

The key aspect of accurately modeling the behavior of the fuel rods as a function of time-dependent LHGRs is to consider the time causality of the data. The prediction at time t_i may appropriately depend on the corresponding input value at t_i and the historical values. Constraining the model to consider the time causality will help the model to capture the relevant relationships between the input and the output. To model this aspect, the convolution operator is used. The convolution operator Conv is a linear transformation:

$$\text{Conv} : I \longrightarrow O \tag{9}$$

that satisfies

$$(\text{Conv} * \mathbf{x})(t) = \sum_{\tau=0}^{L-1} \text{Conv}[\tau - t]\mathbf{x}[\tau] \tag{10}$$

for $\mathbf{x} \in I$ and sequence length L. This can be interpreted as a special case of matrix-vector multiplication where the matrix is band diagonal. With appropriate padding, it is possible to handle the causal relationships in the data. This is demonstrated in figure 6.

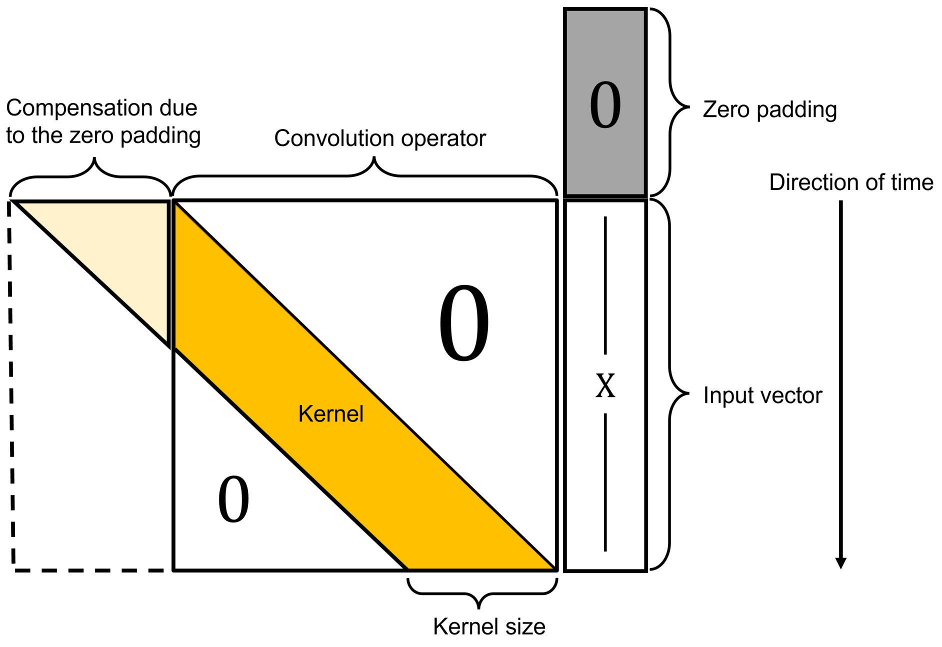


Figure 6: Illustration of the convolution operator that operates over a vector with time-correlated entries. The kernel size determines the "bandwidth". The matrix is extended if padding is used, this is to ensure consistency of the dimensionality of the output.

It is possible to put more constraints on the convolution operator to ensure that it captures the desired properties of the system of interest, introducing the *dilated convolution*. This operation can be visualized in figure 7.

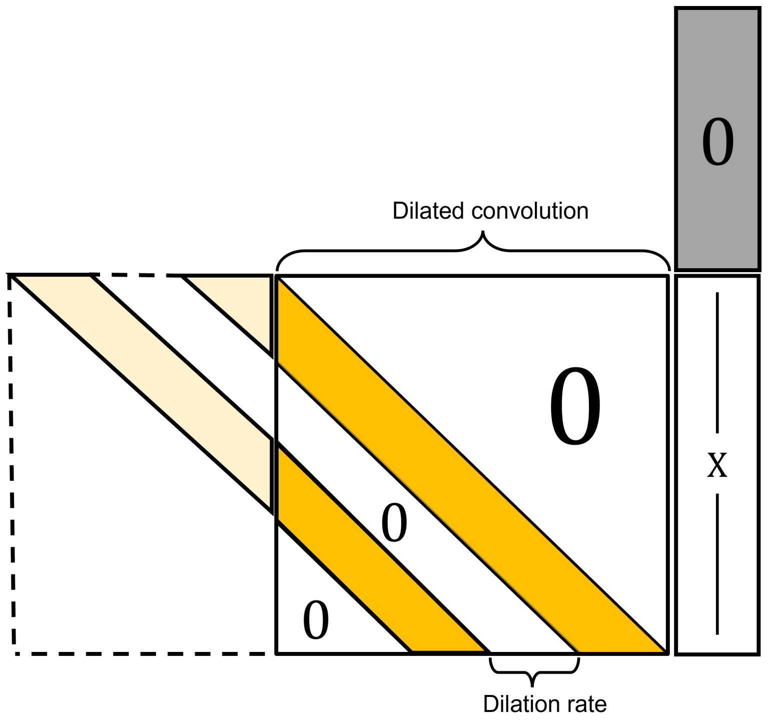


Figure 7: Illustration of the dilated convolution operator. The dilation rate demonstrates the spacing between the separated kernel. In the regular convolution 6 the dilation rate is equal to one. The larger the dilation rate, the band diagonal structure will get separated. As a result, more padding is needed to ensure that the dimensionality of the convoluted input is preserved.

Convolutions can be used to construct a more sophisticated function, e.g., a neural network, this is done by the repeated composition of convolutions followed by non-linear functions. The result is known as the TCN [5, 19, 17], and the architecture is defined as:

$$\mathrm{TCN}(\mathbf{x}) = \mathrm{Conv}_l \circ f \circ ... \circ f \circ \mathrm{Conv}_1(\mathbf{x}). \tag{11}$$

The motivation for dilated convolution is that by sequentially composing convolutions with exponentially increased dilation rate:

$$d_i = b^i \tag{12}$$

for some integer dilation base b and layers $0 \leq i \leq l$. This construction allows for a greatly increased perception field of the composed operation as compared to the individual convolution layer. As a result, the TCN has a superior memory mechanism as compared to a traditional convolutional network that uses a dilation rate equal to one throughout the whole architecture.

In order to get full history coverage of the data, one needs to stack sufficiently many convolution layers. The relationship between the number of layers l for full history coverage depends on the kernel size k, sequence length L, and dilation base b [19]:

$$l = \left\lceil \log_b \left(1 + \frac{(L-1)(b-1)}{2(k-1)} \right) \right\rceil. \tag{13}$$

One also needs to adjust the number of padding p accordingly [19]:

$$p_i = d_i \cdot (k-1). \tag{14}$$

The implementation of the TCN is inspired by [20].

7.3 Fourier Neural Operator

Previously, the convolution operation has been treated as a linear map that maps some input data $\mathbf{x}(t)$ in the time domain to the convoluted output. However, there exist alternative coordinate systems where the convolution can be equally well performed, for instance. In the context of this work, the alternative coordinate system is the frequency domain. The bridge that connects the convolution in the time- and the frequency domain is the *convolution theorem*:

$$(\mathrm{Conv} * \mathbf{x})(t) = \mathcal{F}^{-1}\left[\mathcal{F}(\mathrm{Conv}) \cdot \mathcal{F}(\mathbf{x}(t))\right] \tag{15}$$

where $\mathcal{F}$ denotes the Fourier transform. The key step now is to realize that the convolution operator is a third-order tensor[9] $\mathbf{W}_\omega$, and so does its FFT counterpart:

$$\mathbf{W}_\omega = \mathcal{F}(\mathrm{Conv}). \tag{16}$$

It is in general a nontrivial task for a neural network to learn the discrete Fourier transform matrix. However, with the prior knowledge of the convolution theorem and FFT algorithm, it is possible to guide the neural network to learn in an alternative, but equally meaningful coordinate system by skipping through the FFT of the kernel weights and directly representing them in the frequency domain. The resulting operation that follows is known as the FFC operation [6]:

$$(\mathrm{Conv} * \mathbf{x})(t) = \mathcal{F}^{-1}\left[\mathbf{W}_\omega \cdot \mathcal{F}(\mathbf{x}(t))\right]. \tag{17}$$

The key idea of FFC is that the weight tensor $\mathbf{W}_\omega$ has the ability to capture the frequency features in the data, as opposed to the temporal features in the convolution kernels in TCN. The convolution theorem then guarantees that the result is equivalent to the convolution that is performed in the time domain. Since the weights now represent the frequency components of periodic functions, the FFC operation has the ability to effectively represent the global features of the data. Finally, one can use a series of stacked FFCs and combine them with several standard techniques to form the complete neural network architecture, namely, the FNO [7].

8 Deep Ensembles

A neural network can be considered to be a complex system with many parameters involved. Despite its complexity, its performance is commonly evaluated with a few numbers that represent the error between the predictions and the data. The error is a macroscopic property of the network that the data scientist care about as compared to the specific arrangements of the weights and their values which correspond to a microscopic property. This is analogous to how physicists care about macroscopic properties of physical systems such as the pressure of a set of colliding gas molecules. There are a huge number of different ways the molecules could collide that correspond to the same pressure.

In the context of the optimization procedure of a neural network, the error landscape tends to have many local minima that share the same value of error, but the arrangement of the weights is drastically different. One can capitalize on this property and train multiple neural networks that have similar errors, but different weights arrangements [21]. Each neural network will thus represent similar, but different functions. If validation data is passed as input to the neural networks, the predictions made by each neural network would thus be similar, but slightly different. These predictions can then be combined (for example by computing the average) to obtain a more robust prediction. This method is known as an *ensemble method* which has been traditionally applied to classical machine learning algorithms. This explains the terminology *deep ensembles* as the components of the ensemble consist of **deep** neural networks.

8.1 Homogeneous Ensembles

One of the ways to do ensemble methods is to use the same architecture for every member of the ensemble. These architectures could be trained on different parts of the training data to ensure that they do not learn the same function. Another approach is to initialize the weights of the members differently. Since there are many local minima in the error landscape, different initialization tends to lead to different local minima. Homogeneous ensembles have the advantage of being easily parallelizable.

[9]A matrix with one more dimension.

8.2 Heterogeneous Ensembles

Each of the ensemble members must not in general consist of the same architecture. The ensemble is known as *heterogeneous* when there is a mix of different architectures. The advantage of a heterogeneous ensemble is that other architectures may have different strengths and weaknesses, and combining the predictions of each architecture may contribute to a more complete ensemble architecture, this method is known as *stacked generalization*[10] [22]. The key to success in constructing a heterogeneous ensemble is therefore to choose architectures that learn the desired map in a complementary fashion.

9 Temporal Frequency Network

The TCN architecture can be interpreted as a neural network architecture that learns the temporal relationships in the data with the casual dilated convolutions. The FNO architecture can be interpreted as an architecture that learns the frequency relationships in the data through the linear map that occurs in the frequency domain. The two architectures are mathematically closely related from the perspective of the convolution theorem. However, these architectures differ as the weights lie in different coordinate systems, the TCN has the weights in the time domain while FNO has them in the frequency domain.

There is a fundamental relationship between the variance of a function in time and frequency that is described by the *uncertainty theorem*:

$$\mathrm{Var}(\Psi(t)) \cdot \mathrm{Var}(\hat{\Psi}(\omega)) \geq \frac{1}{4} \tag{18}$$

that holds for any function Ψ in Hilbert space $\mathcal{H}$ such that:

$$1 = \int_{\mathbb{R}} |\Psi|^2 dt. \tag{19}$$

This theorem reveals that there is a fundamental trade-off between information that can be gained for Ψ[11]. In other words, some coordinate systems will almost always be a better choice than their transformed counterpart. To understand why this is the case, consider the following example with the function $\Psi(t) = \frac{\sin t}{t}$ in the time domain. To represent this function, one must (in principle) evaluate $\Psi(t)$ on every real-valued t to capture the complete characteristic of the function. However, its Fourier transformed counterpart is $\hat{\Psi}(\omega) = \sqrt{\frac{\pi}{2}}\,(H(\omega+1) - H(\omega-1))$ where $H(\cdot)$ is the Heaviside function. The observation is that $\hat{\Psi}(\omega)$ can be represented in fewer points in the frequency domain since it needs to be evaluated within a much more narrow interval $\omega \in [-1, 1]$, to capture the complete characteristics of the function. This is visualized in figure 8.

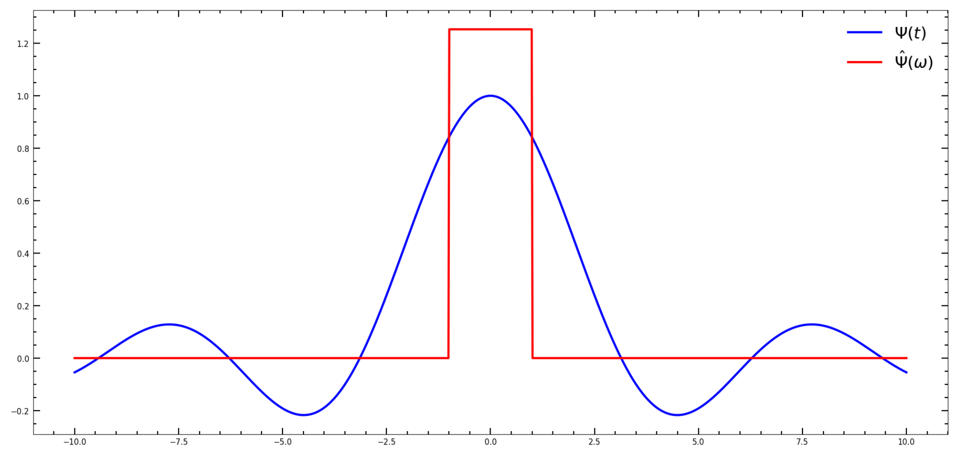


Figure 8: Comparing $\Psi(t) = \frac{\sin t}{t}$ against its Fourier transformed counterpart $\hat{\Psi}(\omega) = \sqrt{\frac{\pi}{2}}\,(H(\omega+1) - H(\omega-1))$. The latter is better localized in the frequency domain.

[10] In the original paper the author trained the models with different parts of the training data and then combines the individual components. However, in this work, I have taken a slightly different approach by training the ensemble on the complete training data with distinct architectural differences to ensure heterogeneity.

[11] There is an exception, that is if Ψ is a Gaussian. In that case, there may not be a preference over either of the coordinate systems.

In other words, the Fourier transformed version $\hat{\Psi}(\omega)$ yields (in this example) a superior compression as compared to $\Psi(t)$.
The uncertainty theorem thus sets up a promising mathematical foundation for constructing complementary architectures that can be used in designing a heterogeneous ensemble. The result of which is the TFN. This is a novel proposed neural network architecture that is based on the principles of heterogeneous ensembles by combining two architectures whose weights lie in complementary coordinate systems. This results in maximizing the complementary effects of the individual ensemble components.

The general definition of TFN is given by:

$$\text{TFN}(\mathbf{x}) = \alpha\hat{\varphi}(\mathbf{x}) + (1-\alpha)\varphi(\mathbf{x}) \tag{20}$$

where $0 \leq \alpha \leq 1$ is a trainable parameter that is initialized to be 0.5. The α-parameter plays the role of shifting the optimization to either coordinate system depending on the nature of the dataset. The architectures that were used to represent the time- and frequency architectures are $\varphi = \text{TCN}$ and $\hat{\varphi} = \text{FNO}$ respectively:

$$\text{TFN}(\mathbf{x}) = \alpha\text{FNO}(\mathbf{x}) + (1-\alpha)\text{TCN}(\mathbf{x}) \tag{21}$$

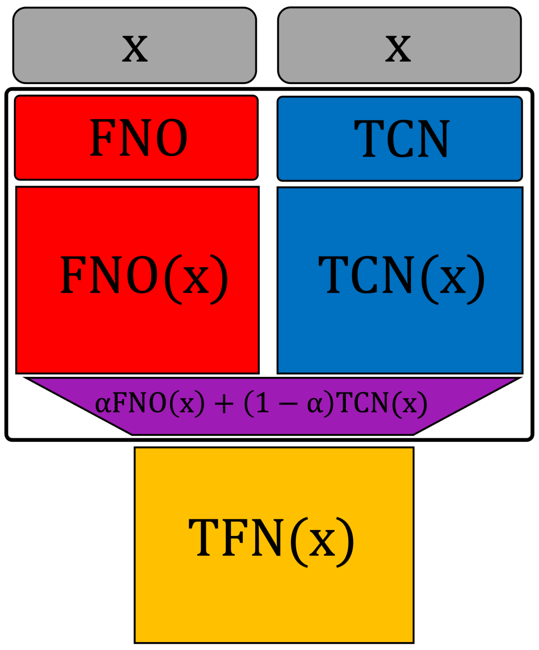


Figure 9: Illustration of the TFN architecture.

Despite the simplicity of TFN, it is shown in this work to be significantly better than its individual components, this is shown in the result section 12. The implementation can be found at: https://github.com/SunAndClouds/Examensarbete

10 Interpretable Transformer

The TCN is a reasonable architecture to capture explicit time-dependent patterns. However, the weights of the convolution kernels are difficult to interpret. The case for FNO is similar. This is a general drawback of neural networks since more layers bring significantly better performances but also fundamentally make it more challenging to interpret. Therefore, another neural network, a *Transformer* [8, 23], is trained to get more insight into the dataset. This part should be treated as an independent extension of the TFN.

10.1 Self-attention Mechanism

This is the core component of Transformer architecture. Consider the following example, given a sequence of the data $\mathbf{x} = (Q_1, Q_2, Q_3)$ and one wish to know how the point Q_3 depends on the past points Q_1 and Q_2. The relationship can be represented by the following diagram[12].

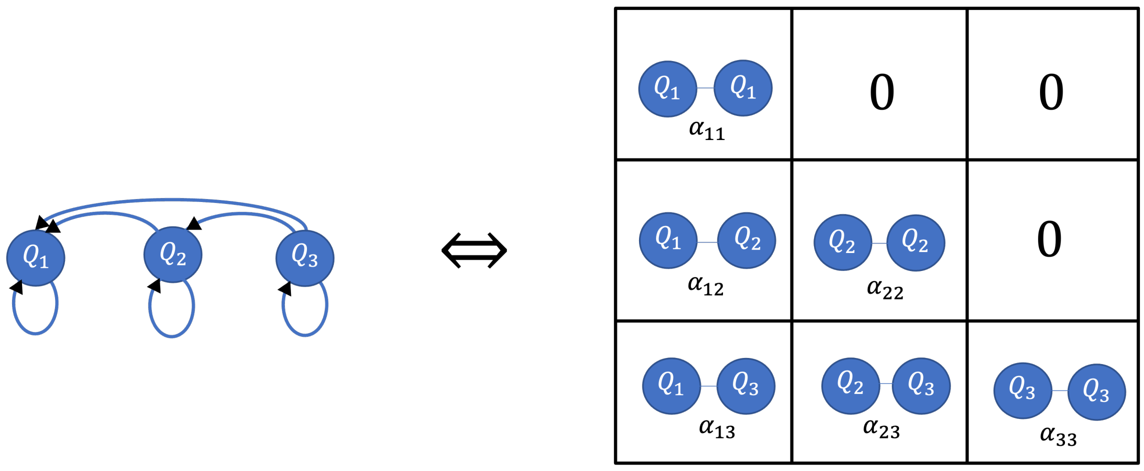

Q_1
Q_2
Q_3
$\Leftrightarrow$
Q_1 Q_1 α_{11}
0
0
Q_1 Q_2 α_{12}
Q_2 Q_2 α_{22}
0
Q_1 Q_3 α_{13}
Q_2 Q_3 α_{23}
Q_3 Q_3 α_{33}

Figure 10: Left, correlations between the points, including correlation with itself. Right, the same representation is in matrix form where each entry is given by the attention weights.

Note how the upper triangle is filled with zeros to prevent past points to get influenced by future points. The entries of the matrix α_{ij} indicate how related two pairs of points are. One way to compute the similarity is by taking the dot-product between the row vector $\mathbf{x}$ and the column vector $\mathbf{x^T}$. Note how this step also works if $\mathbf{x}$ is a matrix. This step is then generalized as follows:

- Perform a linear map for $\mathbf{x}$ and adding a vector $\mathbf{b}_Q$

$$\mathbf{Q} = \mathbf{x}\mathbf{W}_Q + \mathbf{b}_Q. \tag{22}$$

- Perform another separate linear map for $\mathbf{x}$ and adding a vector $\mathbf{b}_K$

$$\mathbf{K} = \mathbf{x}\mathbf{W}_K + \mathbf{b}_K. \tag{23}$$

- Compute the similarities of the matrices by taking the dot-product

$$\mathbf{Q}\mathbf{K^T}. \tag{24}$$

It is now convenient to output the similarity measure of each pair of points as a probability distribution. This is ensured by the softmax function (the mathematical term for Boltzmann distribution)

$$\mathbf{P}(\mathbf{x}) = \mathrm{Softmax}(\mathbf{Q}\mathbf{K^T}) = \frac{e^{\mathbf{Q}\mathbf{K^T}}}{\sum_{i,j}^{n} e^{\mathbf{Q}\mathbf{K^T}}}. \tag{25}$$

Note that the matrix exponent is in this case performed element-wise. The softmax function has a slight numerical downside, large negative inputs tend to all be close to 0, which may

[12]Note that this is a simplified diagram, the actual attention matrix considers the correlation between the query $\mathbf{Q}$ and the key $\mathbf{K}$ matrices. The diagram assumes that they are equal which in the general case is not.

cause a problem with the training. Therefore, the argument of the softmax is normalized by the square root of the dimension of $\mathbf{K}$

$$\mathbf{P}(\mathbf{x}) = \text{Softmax}\left(\mathbf{Q}\mathbf{K}^{\mathbf{T}}/\sqrt{\dim(\mathbf{K})}\right). \tag{26}$$

This matrix is called the *attention weight matrix*, and its elements consist of the α_{ij} in the matrix in figure 10. This matrix can be interpreted as a *right stochastic matrix* due to the row-wise softmax operation, that is, a matrix whose row-sum is equal to 1. The coefficients are called *attention weights*. These attention weights can now be used to extract information from the input $\mathbf{x}$. Similarly as equation (22) and (23), this steps is generalized by first computing

$$\mathbf{V} = \mathbf{x}\mathbf{W}_V + \mathbf{b}_V. \tag{27}$$

The information is then extracted from the matrix $\mathbf{V}$

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathbf{P}(\mathbf{x})\mathbf{V} \tag{28}$$

which completes the self-attention mechanism. The result is a matrix of the same dimension as the input $\mathbf{x}$. The weights in the matrices $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V$ and vectors $\mathbf{b}_Q, \mathbf{b}_K, \mathbf{b}_V$ together with the softmax function ensures that Attention$(\mathbf{Q}, \mathbf{K}, \mathbf{V})$ have extracted the temporal relationships between the points.

A simplified transformer may consist of one self-attention layer. However, in this work, the self-attention layers were stacked in parallel to form the *multi-head attention* layer [8]. This is based on the same principle as the self-attention mechanism, but each head (self-attention layer) can extract different features from the data. With only one multi-head attention layer, it is possible to construct a simple, yet powerful Transformer that is also interpretable.

After a successful training process, the attention weight matrix $\mathbf{P}(\mathbf{x})$ will faithfully represent the causal relationships between data points that are temporally correlated. This informs how the next prediction of the transformer will depend on past observations.

11 Outlier Detection with Autoencoder

In fuel performance modeling, reliability is a crucial property of the fuel performance code, there is no exception for the surrogate model. To test the model's capabilities, it is challenged with unseen data. However, there is a distinction between **precision** and **accuracy**. The former refers to a specialized model that may perform well on a narrow subset of all possible LHGRs, and the latter refers to the model's capability to generalize to a wide variety of possible LHGRs. The network was trained on a relatively small dataset, which makes it precise as shown in the result section 12, but there is no guarantee that it is accurate. When the trained network is deployed in practice, the user may not be aware of the generalization capabilities of the model. This may cause harm when the prediction quality does not match the results in this report. Therefore, a robust methodology for determining the quality of the predictions is essential for real-world deployment. To gain the trust of the model predictions, an *autoencoder* setup is used for performing outlier detection.

An autoencoder consists of two parts, an *encoder* and a *decoder*. The encoder is a model, φ, that is trained to map the input $\mathbf{x}$ to the desired result $\mathbf{y}$:

$$\mathbf{y} = \varphi(\mathbf{x}). \tag{29}$$

So far, all the previously mentioned architectures can perform the task of an encoder. The decoder, φ^{-1}, is a model that is trained to perform the inverse task, i.e., mapping the outputs to the reconstructed input $\mathbf{x}'$:

$$\mathbf{x}' = \varphi^{-1}(\mathbf{y}). \tag{30}$$

In essence, the composition of the encoder and the decoder is the autoencoder, which is defined as:

$$\mathbf{x}' = \varphi^{-1} \circ \varphi(\mathbf{x}) \tag{31}$$

With a successfully trained autoencoder, $\mathbf{x}$ and $\mathbf{x}'$ should be similar but not strictly the same due to the non-trivial architectural design of φ and φ^{-1}. The autoencoder architecture is thus capable of producing a reconstruction that is dependent on the output, either by TU or the neural network, this opens up the possibility of computing a *reconstruction error* ε between the true input $\mathbf{x}$ and the reconstructed input $\mathbf{x}'$ with an objective function that minimizes the difference between the original input and the reconstructed input:

$$\varepsilon = \mathcal{L}(\mathbf{x}, \mathbf{x}'). \tag{32}$$

The error of the decoder can be evaluated in two different ways:

1. The reconstruction is based on the validation output data from TU that the decoder was not trained on, $\mathbf{y}_{TU,valid}$. This produces the validation reconstruction error:

   $$\varepsilon_{TU} = \mathcal{L}(\mathbf{x}_{TU,valid}, \varphi^{-1}(\mathbf{y}_{TU,valid})). \tag{33}$$

   This evaluation metric is used to evaluate the generalization capabilities of the autoencoder during the training when the ground truth outputs of TU are available.

2. The reconstruction is based on the validation input data that the encoder was not trained on, $\mathbf{x}_{TU,valid}$. This produces a different set of inputs $\mathbf{y}_{\varphi,valid} = \varphi(\mathbf{x}_{TU,valid})$ that is then passed to the decoder. The test reconstruction error is thus given by:

   $$\varepsilon_{\varphi} = \mathcal{L}(\mathbf{x}_{TU,valid}, \varphi^{-1}(\mathbf{y}_{\varphi,valid})). \tag{34}$$

   This evaluation metric is used to evaluate a more realistic generalization error when the ground truth TU outputs are not available.

There are several possible outcomes for different values of ε_{φ} and ε_{TU}:

1. Precise prediction by φ and low reconstruction error. This is the idealized outcome where both the encoder and decoder perform well. As a result, the prediction of the encoder will be accepted as a TP case.

2. Poor prediction by φ and high reconstruction error. This is also a desirable behavior that shows the decoder's capability to successfully identify the TN cases.

3. Precise prediction by φ and high reconstruction error. This is a less desirable case that corresponds to a precise encoder but a poor decoder. As a result, the FN will be treated as a TN case, which corresponds to a conservative decision.

4. Poor prediction by φ and low reconstruction error. This corresponds to an FP case which is a catastrophic outcome that will likely misinform the user. However, this is a very rare event as shown in figure 23.

The first three aspects can be improved with a better-trained encoder and decoder. However, considering the field of application of the neural networks, it is recommended to consider the worst-case scenario, i.e., the FP cases.

To further reduce the probability of obtaining an FP case, a homogeneous ensemble setup was used. The ensemble that was used consisted of three TCNs due to the low inference time[13]. Each of the TCN was trained using different weight initialization, figure 11 shows that each member does a good job of reconstructing in-distribution validation data. In deployment, each of the TCNs may make its own reconstruction based on the encoder prediction, the prediction may then be rejected if at least one of the TCNs does not predict low reconstruction error. A hypothetical FP case must thus "fool" all three TCNs to be found in the final set of TP predictions. This setup can be found in the source code.

[13]It is possible to train a smaller TFN that also has the advantage of being fast. The smaller design choice was not explored due to time constraints.

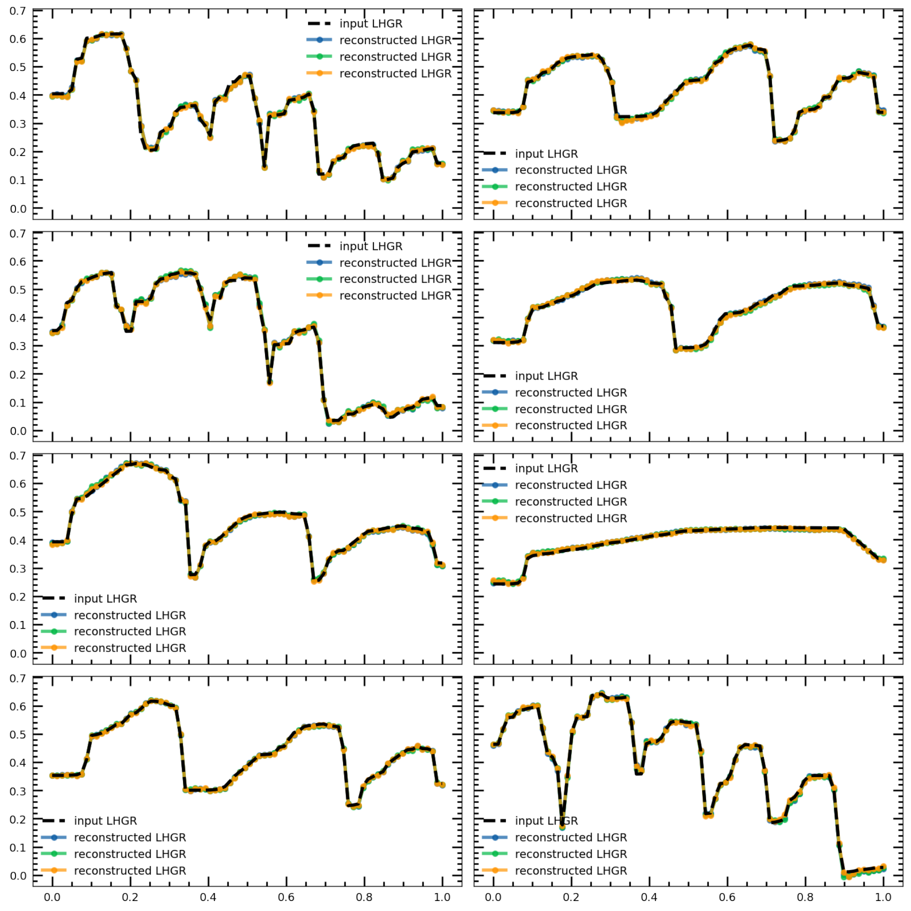


Figure 11: Evaluating three different TCN decoders on TFN encoder predictions that are based on in-distribution validation data. Although the trained weights are completely different, the validation reconstruction is similar for each decoder.

However, if an out-of-distribution input is passed to the encoder, a poor reconstruction is expected, this is demonstrated in the figure below.

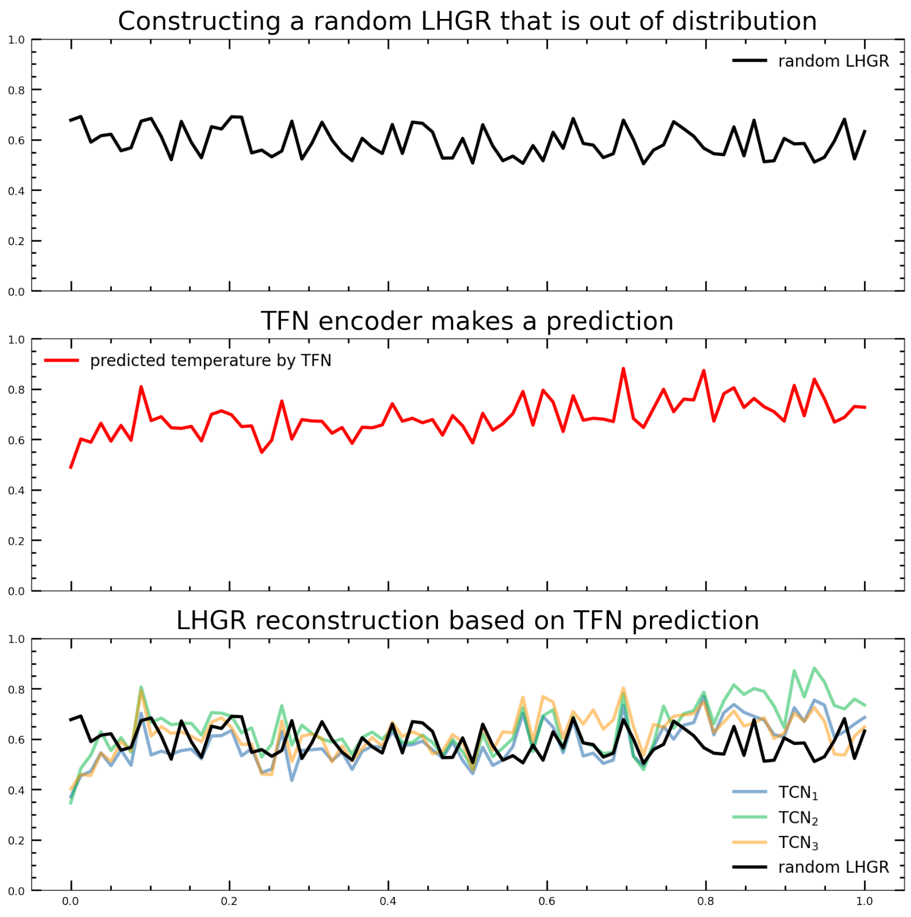


Figure 12: Evaluating three TCN decoders on TFN encoder prediction based on a random out-of-distribution LHGR. Each decoder produces qualitatively different reconstructions although they were similar in figure 11. Note that although the true temperature prediction is not known, it should be qualitatively similar to the input LHGR.

12 Results

The results show that even with a relatively small dataset, it is possible to achieve promising precision. This is observed for all architectures that were evaluated in this work, however, the TFN shows significantly lower validation error than any other architectures at little computational overhead.

12.1 Baselines

After the training process, each encoder architecture is evaluated on the validation dataset which consists of 216 samples that were not directly present in the training set[14]. Based on the validation predictions, the test (same as validation) error can then be computed sample-wise, resulting in 216 different errors. The distribution of these errors is then visualized with a histogram in figure 13.

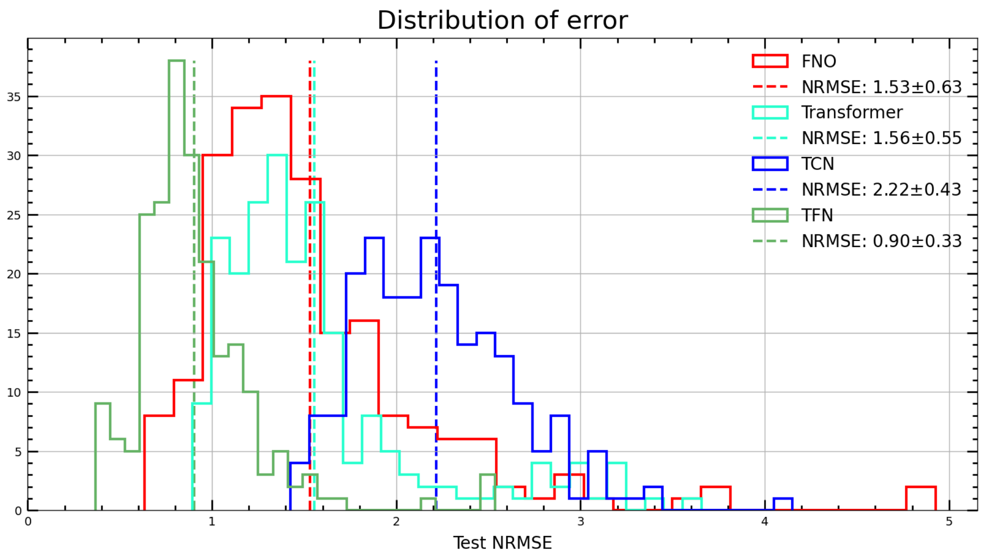


Figure 13: Distribution of the NRMSE for different encoder architectures evaluated on unseen data using normalized LHGR as input and NRMSE as the objective function. Note that the test NRMSE is in this case the same as the validation NRMSE.

[14]The validation data is sampled from the same distribution as the training set.

12.2 Network Predictions

The input validation dataset is passed through the TFN as LHGR to obtain the predictions. The validation error is then computed for the corresponding prediction. Note that there are 10 fuel pellets in each fuel rod, the result below 14 thus only displays the qualitative behavior of the predictions, specifically the first fuel pellet. The NRMSE is also averaged over all output variables.

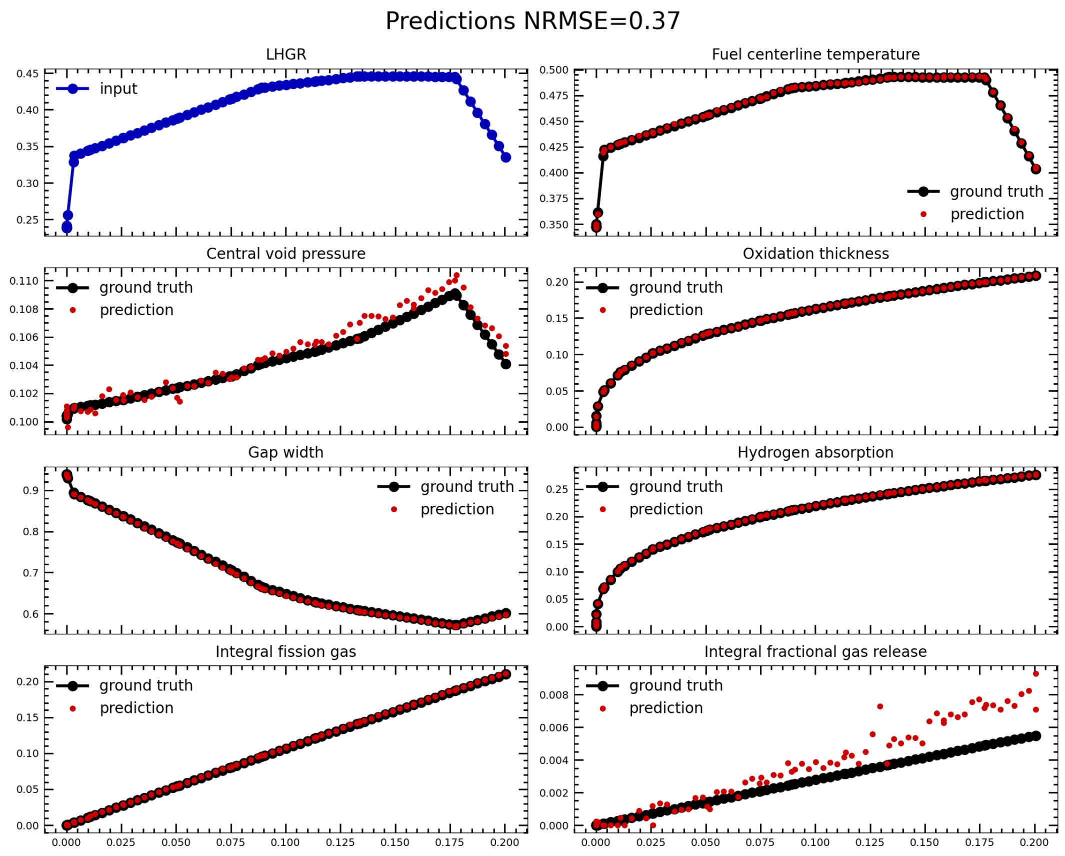


Figure 14: Prediction made by TFN for one cycle LHGR operation. This set of predictions has the lowest NRMSE among all architectures. Upon initial inspection, the predictions for central void pressure and integral fractional gas release may appear to be significantly worse. However, by looking more closely at the scale of the y-axis it is apparent that the NRMSE is indeed small, for all displayed predictions.

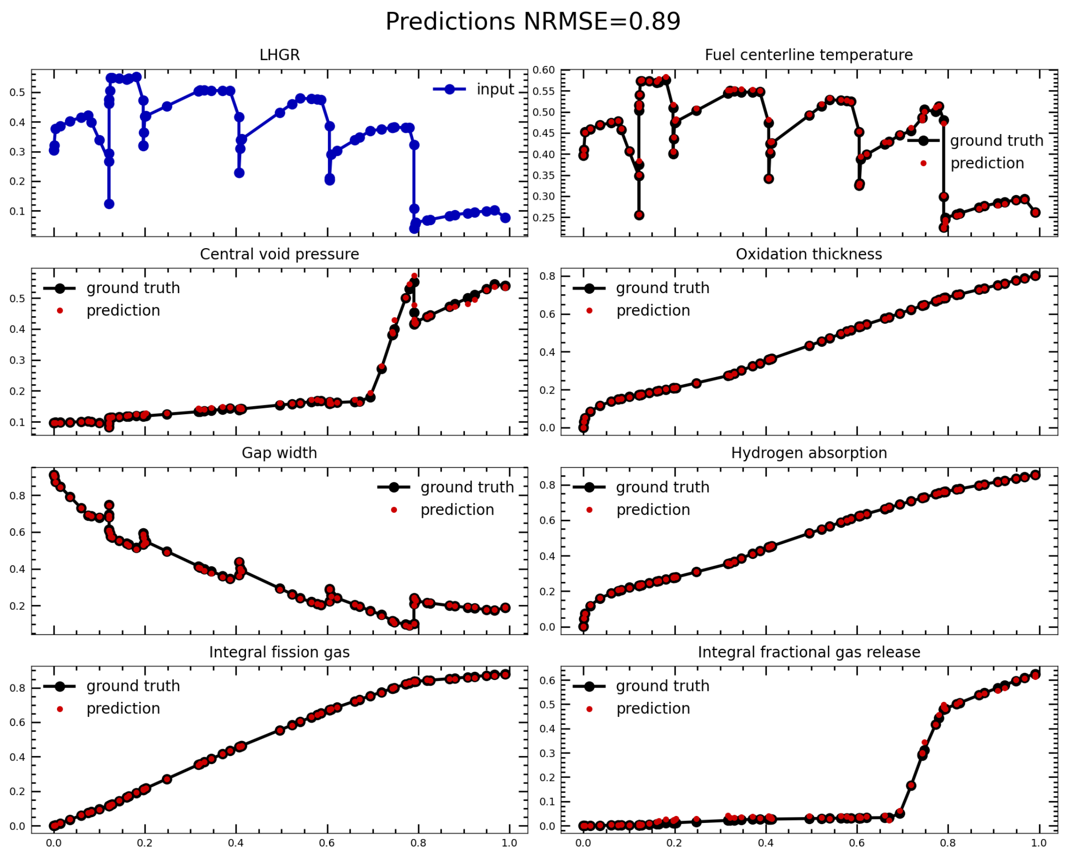


Figure 15: Prediction made by TFN for five cycles of LHGR operations. The scale of the y-axis is in the same order of magnitude, therefore, despite the higher validation error, the predictions are more visually pleasing than figure 14. Note that the value of the NRMSE is very close to 0.90, which is the average NRMSE of TFN.

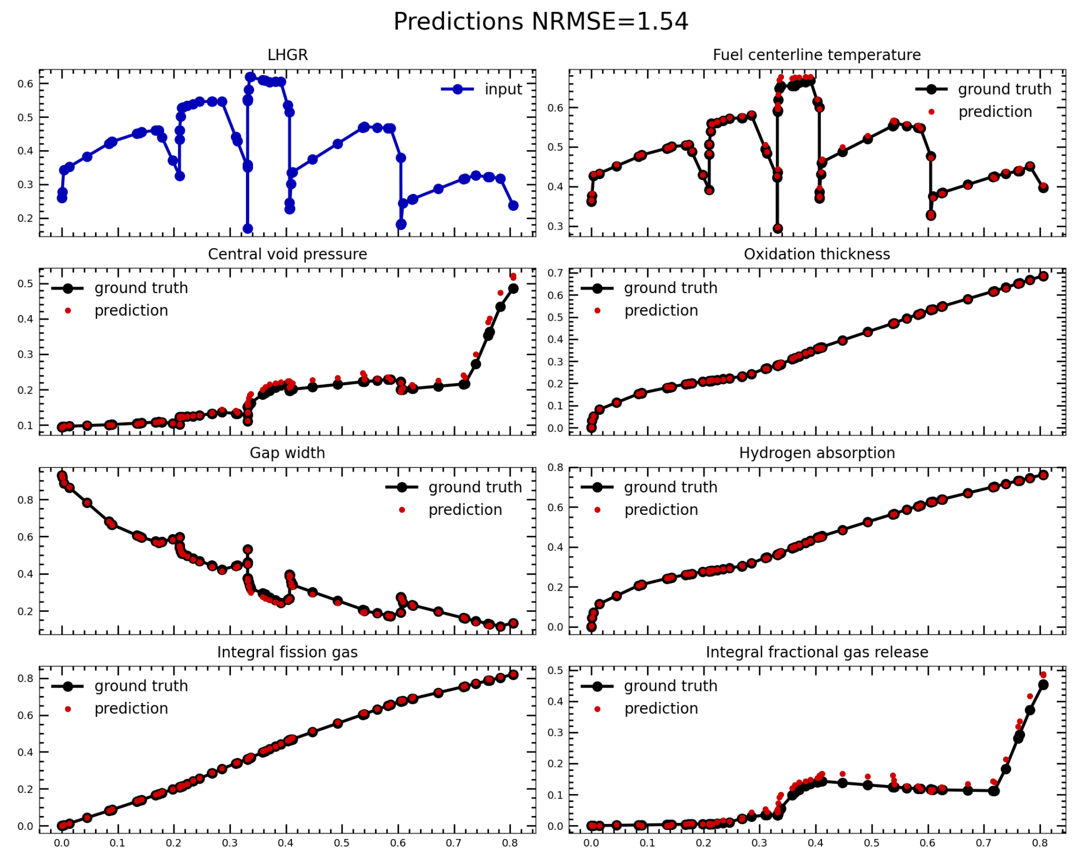


Figure 16: Yet another five-cycle LHGR prediction made by TFN. Note that the NRMSE of this set of predictions is close to the average NRMSE of FNO (1.53) and Transformer (1.56) as shown in 13.

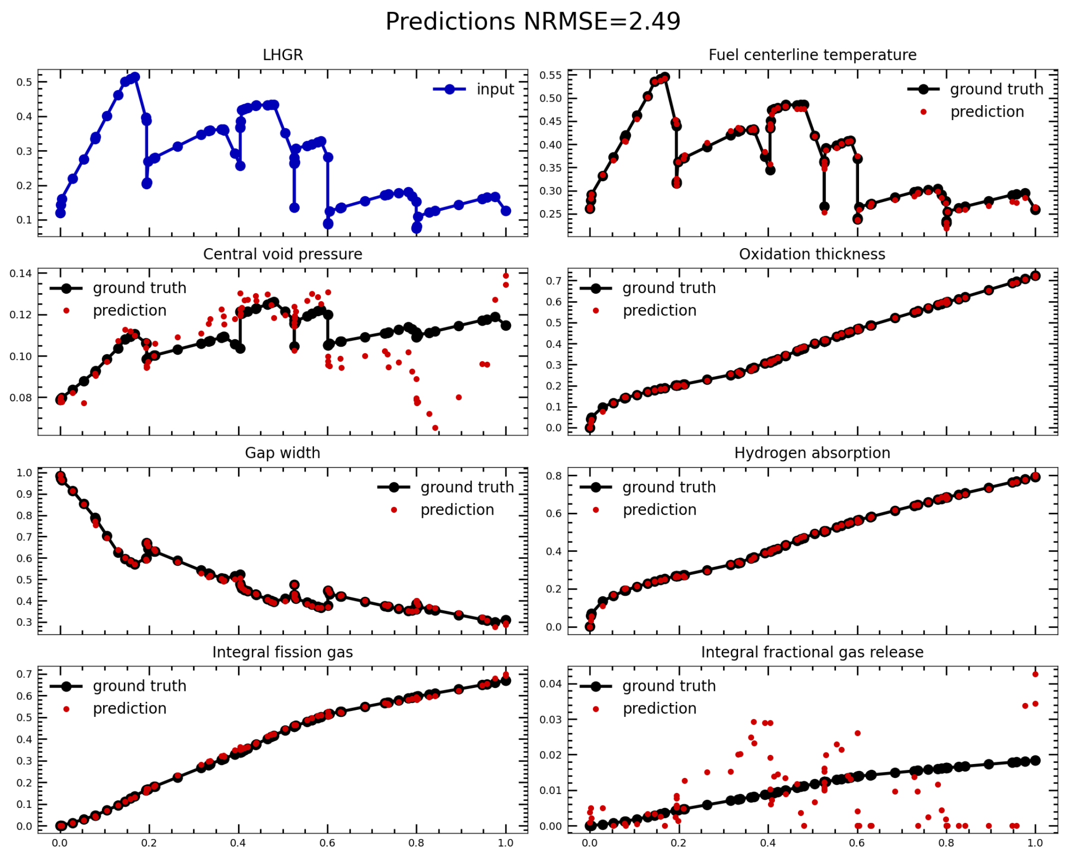


Figure 17: This set of predictions has the largest validation error for TFN and may be considered to be an outlier case. The central void pressure and integral fractional gas release are not well predicted. A further discussion about why the TFN performed badly on this particular LHGR is provided in section 13.3.

12.3 Attention Plots of LHGR

The attention matrix can be interpreted as a right stochastic matrix. The value of the attention weight of a row thus corresponds to how important the information at each time step of the LHGR contributes to the corresponding predictions. In general, attention weights that lie close in time in relation to the prediction time-step tend to have a high value. However, there may be implicit changes in the boundary condition of the system that may have a significant influence on prediction far into the future. Such events may be identified as an "spike". A more in-depth discussion is provided in the section 13.5.

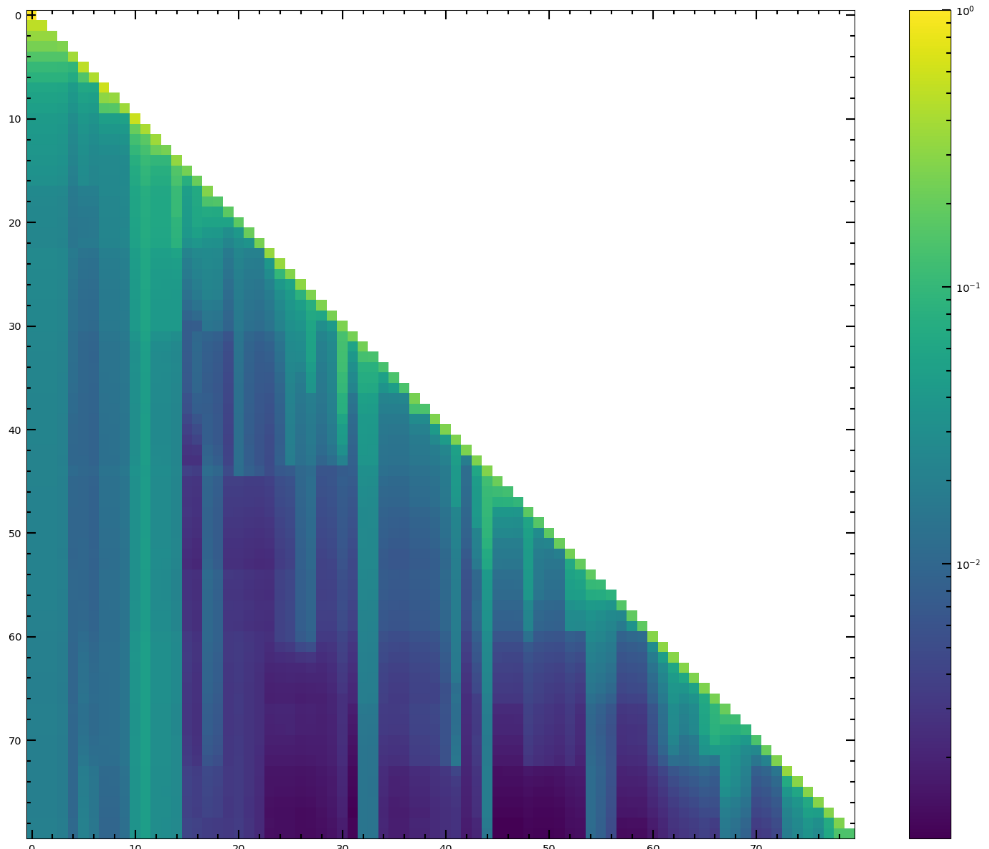


Figure 18: This plot shows the attention weight matrix defined in equation (26). Note how the matrix is lower triangular to ensure that only the past attention weights can contribute to the prediction.

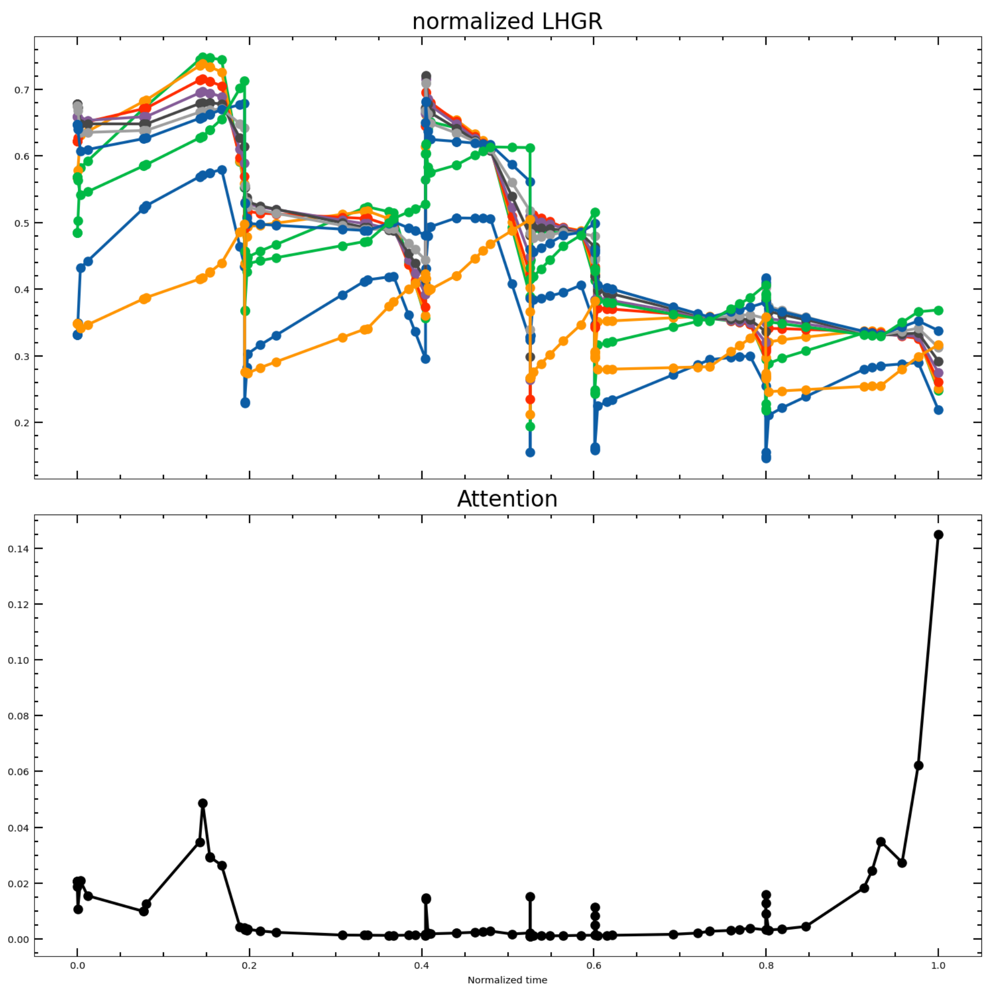


Figure 19: The attention weights in black corresponds to the last row of figure 18. The value of the right-most attention weight is the largest because it directly influences the prediction at the normalized time-step 1.

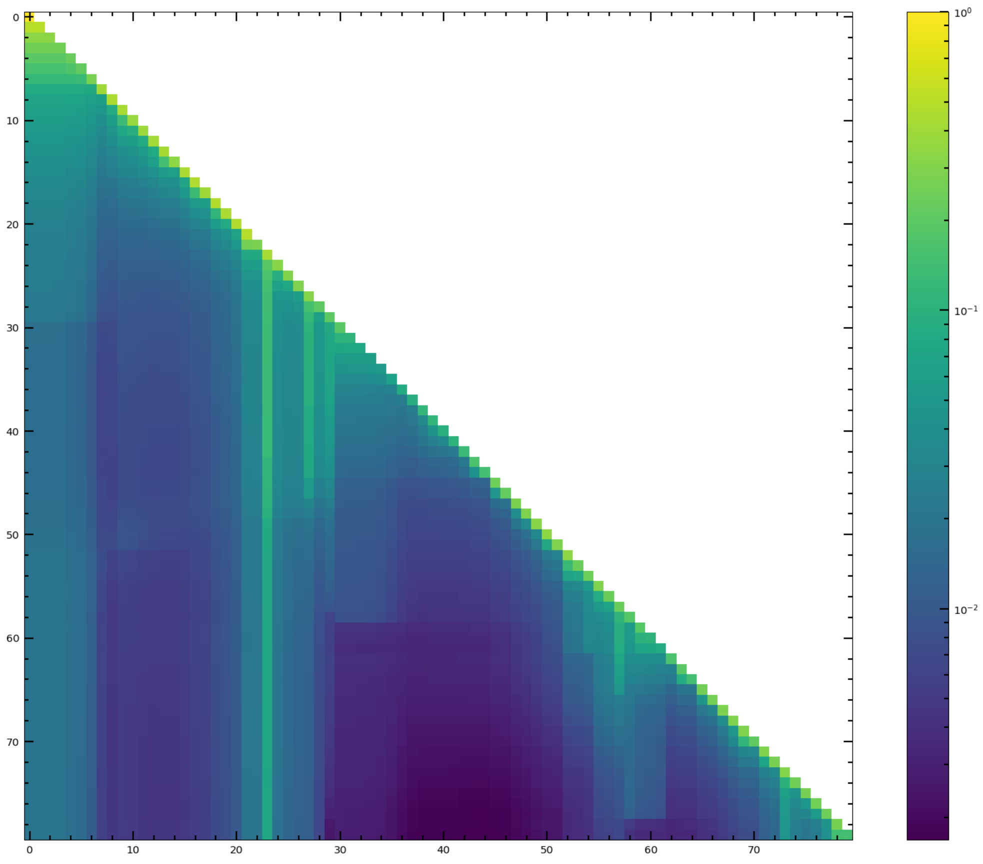


Figure 20: This plot shows the attention weight matrix for another set of LHGR.

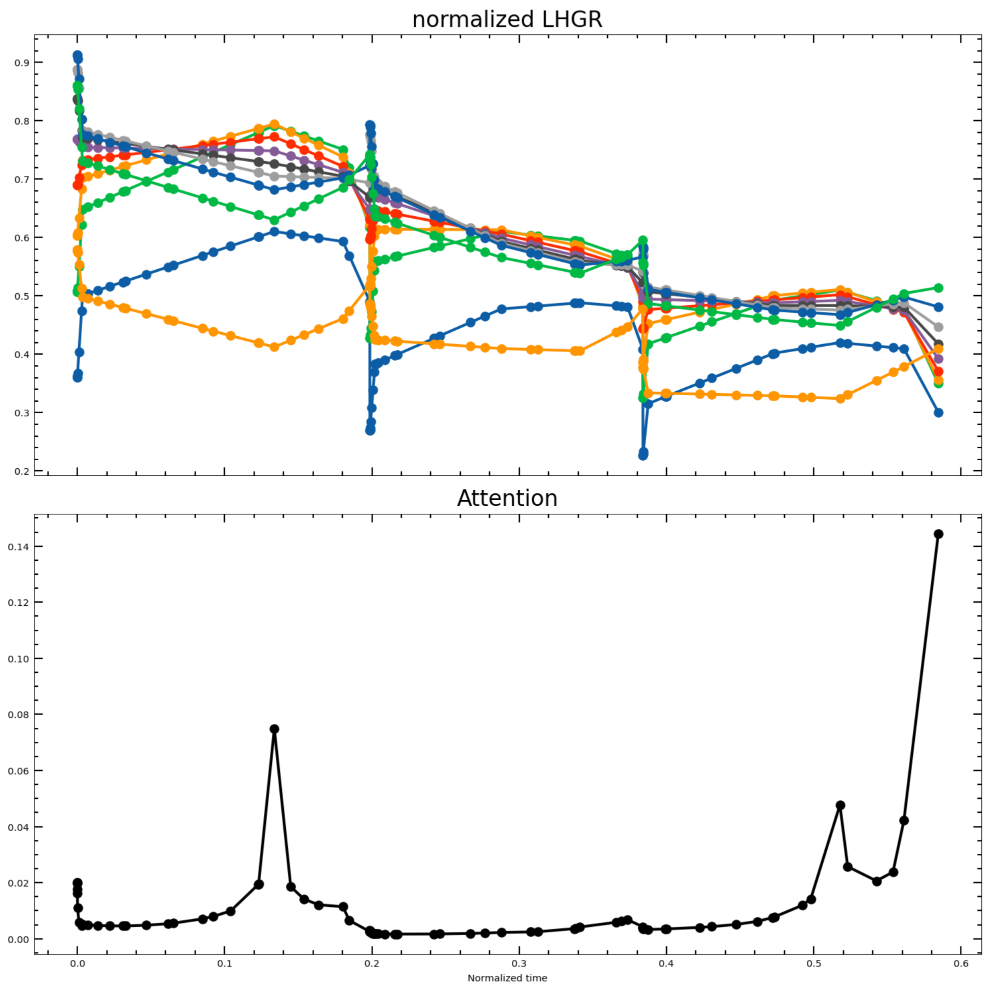


Figure 21: The attention weights has a clear spike between the normalized time 0.1 to 0.2. This spike is likely explained by a change in boundary conditions.

12.4 Decoder Performances

Similar to how the baseline performance is evaluated. After the training process, each decoder architecture is evaluated on the validation dataset which consists of 216 samples that were not directly present in the training set. The distribution of each sample-wise error is visualized in figure 22.

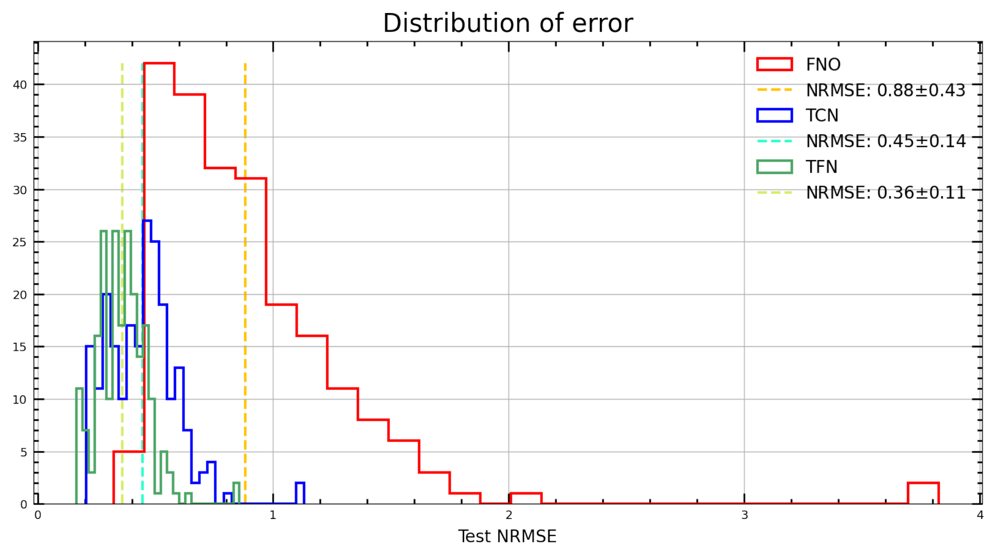


Figure 22: Distribution of the NRMSE for different decoder architectures using the validation data $\mathbf{y}_{TU,valid}$ as input. The decoders were trained on NRMSE.

As discussed in section 11, there exist two alternative ways of evaluating the validation error of the decoder. These two errors are in general, not the same. Figure 23 shows the correlation between the two types of reconstruction error. A more in-depth discussion is provided in the section 13.6.

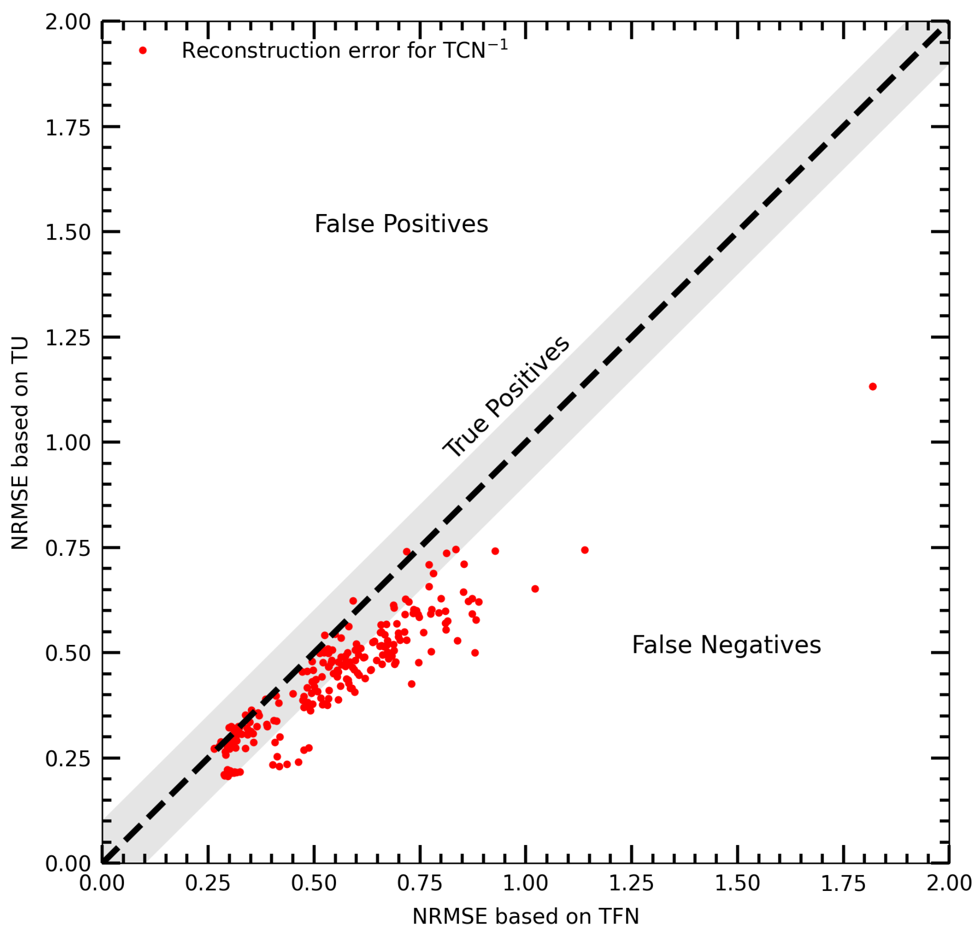


Figure 23: This plot compares the error of the TCN decoder with default weight initialization given the predictions that are based on $\mathbf{y}_{TU,valid}$, in the y-axis and, $\mathbf{y}_{TFN,valid}$ in the x-axis. The values of x- and y-axis are thus given by (34) and (33) respectively. The left upper corner corresponds to hypothetical FP cases. The band in the diagonal is an arbitrary threshold that the user defines to be classified to be TP. The right bottom corner corresponds to the FN cases. The right upper corner represent the TN cases.

13 Discussions

13.1 Related Work

It is worth noting that during the implementation process of FNO, G. Robertson hypothesized that replacing the residual layers in FNO with TCNs may improve the FNO architecture.

13.1.1 TFN vs TFCNN

The idea of simultaneously extracting time- and frequency features from the data is not new, one example is the TFCNN [24][25]. It works by constructing a spectrogram from the time-dependent data and using that as a 2D input to a convolutional neural network allowing TFCNN to extract temporal and frequency features from the data simultaneously. However, there are some distinct differences:

- TFCNN transforms the time-dependent data into a spectrogram and employs a 2D convolutional network to process the spectrogram and extract features. On the other hand, TFN does not explicitly create a spectrogram representation of the input data.
- TFCNN uses an attention mechanism that focuses on specific frequency bands and time frames on the spectrogram, aiming to reduce the influence of background noise and irrelevant frequency bands. TFN, however, does not utilize an attention mechanism in this manner.
- TFN combines two distinct architectures, TCN and FNO, which operate in the time and frequency domains, respectively. This combination is achieved through a learnable parameter α, which allows the architecture to shift optimization towards the domain that best represents the nature of the dataset. This kind of dynamic balancing is not present in TFCNN.

13.1.2 TFN vs Inception

The idea of combining different hierarchical features from the data using combined operations is not new either. The Inception architecture [26] can adapt to patterns of different scales within the data due to the use of parallel convolutional layers with different kernel sizes, which is similar to how temporal and frequency features are combined in TFN. However, there are significant differences that distinguish the two architectures:

- Inception uses concatenated feature maps from different convolutional layers to capture multi-scale information in the spatial domain only, while TFN integrates features from both time and frequency domains.
- The dynamic optimization process with α is not present in the Inception network.

13.2 General Observations During Training

The Transformer and FNO outperform the TCN with a lower expected value of the error and the standard deviation of the error distribution is also smaller. This is likely explained by the larger number of parameters used. This is a common phenomenon that is observed in NLP [27]. Although the task is not NLP, increasing the size of the dataset and parameters may still contribute to improved performance.

In training the different neural networks, the loss curve for TCN converged slowest among the architectures, this is likely due to the low number of parameters, ~ 5000 as compared to $\sim 900,000$ of FNO and TFN. After training for 30000 epochs, the NRMSE for TCN dropped from 2.22 to as low as 1.75. The observation is that smaller models need to train for longer to reach competitive performance as larger models such as FNO (1.53) and Transformer (1.56). However, the validation error does not asymptotically converge to the same value. This phenomenon has been empirically observed on other training occasions [28]. The loss curves are provided in the Appendix.

The 1200 training samples and 216 validation samples that the neural networks were trained on is a rather small datasets for DL standards. However, upon increasing the size of the training set to 10,000 samples and 2491 validation samples the validation error did not change significantly. A possible explanation for this observation is that the generalization ability of the model has been improved because of the increased diversity of the training data. To decrease the validation error one needs more data that is semantically similar, especially for those predictions with a high validation error.

13.3 Analysis of the Baseline Performances

Two visual anomalies can be observed in figure 14, despite being the prediction with the lowest NRMSE the predictions for the central void pressure and integral fractional gas release look to be inaccurate. This visual bias is caused by the tiny scaling of the y-axis. For the pressure, the difference between the lowest value and the largest is about 10^{-3}. This is a significantly smaller scale than the other predictions where the order of the scale is 10^{-1}. If all plots were represented with the same scaling in the y-axis, then pressure and gas release would be close to a flat line.

Figure 17 shows the worst-case performance of TFN, a hypothesis that may explain the poor performance is that the LHGR may be produced by the Gadolinium rod. There are significantly fewer Gadolinium rods in a reactor as compared to the pure UO_2 fuel rod. This explains why TFN is able to predict some of the more regular behaving variables such as fuel centerline temperature, oxidation thickness, and hydrogen absorption, which behave similarly for all types of rods.

13.4 Data Transformations

There are some constraints that must be considered when transforming the data to ensure that the necessary physical properties of the data are preserved. The relevant aspects include:

- Uniqueness of the transformations.
- Total energy conservation.
- Temporal causal relationship.

The first two conditions are satisfied under the class of linear transformations. That is why linear scaling is performed in the pre-processing step. A common scaling technique that does not work is logarithm scaling, which is a unique transformation but does not preserve the total energy (the time integral of the LHGR) before and after the inverse transform due to the non-linearity. However, other variables may be scaled using a unique and non-linear transformation. The third condition is satisfied by padding the dilated convolutions as well as guaranteed by the convolution theorem.

It is also worth noting that throughout the work, the data is assumed to be functions in Hilbert space. One trivial assumption that is made is that the data can be represented by an FS, but an FS only converges in the l_2 metric, which is why variants of the mean squared error are used. In other words, there is no guarantee that the loss will decrease during the training if the objective function is chosen otherwise.

13.5 Implications of the Attention Matrix

The attention graphs should indicate that points in close proximity in time typically have a significant impact on the subsequent state of the system. This is indeed observed for some of the points. However, there seem to be some exceptions.

Figure 20 and 21 display a clear spike in attention around 1.35 that is aligned with a sudden change in the LHGR. This shows that an important event in the past had a significant impact on the prediction of the subsequent state of the system. Another spike in attention can be observed at around 0.52, which also occurs at a sudden change in LHGR. This is likely caused by an irreversible change in the boundary condition that has been triggered by the change in the LHGR, although the exact cause is unknown. On the counterpart, the rapid changes in the LHGR that come after at around 0.2 and 0.38 did not have a significant effect.

13.6 Analysis of the Decoder Performance

Although there is a significant performance difference among the decoder architectures, they mostly differ by the errors of the outliers. However, since the inference time is a crucial factor one may instead use a homogeneous ensemble of TCN as the decoder.

Figure 23 shows that the error of the TCN decoder is somewhat skewed towards the FN side. An ideal case would be that all the red points lie on the black reference line. However, this is expected because the predictions that the TFN produces are slightly different than the ideal validation data which TU produced. This input error then gets amplified by the imperfect training process of the TCN decoders.

There are a few points that are slightly above the reference line. These points may be classified by the decoder as FP cases depending on the chosen threshold. Choosing a permissive threshold is usually not a big problem in practice. In the case of figure 23, the threshold is 0.1 which would approximately give predictions where the error is lower than the average for TFN (NRMSE=1.10).

In figure 11, the individual TCN encoder are highly consistent with the reconstruction LHGR although they were trained with different weight initialization, thus different function approximations. This is further confirmed by evaluating the individual decoders on the same encoder prediction that is based on an out-of-distribution input in figure 12. None of them performs particularly well, as expected. This is a sign that the encoder has produced an inaccurate prediction.

13.7 Recommendations

If high-quality predictions are desired, one may change the threshold for declining the TN cases to be a low value of the NRMSE. The trade-off is that there will be relatively many potential good predictions from the encoder that needs to be checked by the TU. One may use figure 24 to select a desired error threshold.

In practice, it is always a good idea to run the TU for the full core evaluation before deploying the final design choice, even when a strict threshold is chosen. One may therefore use a threshold that rejects the "obvious" outliers to reduce the usage of TU in the earlier phases of the development cycle.

There is a small trade-off between the validation error and the inference time. One may thus decrease the number of parameters of FNO to increase the inference time of TFN. Although TCN ensembles were used for faster inference of the decoder, it is, in general, a good idea to link small-sized FNOs to construct a homogeneous TFN ensemble.

It is recommended that the user performs the same scaling and interpolation as described. The performance might still be good for other time steps, especially for FNO as it learns a resolution invariant representation of the data [7]. However, TCN is dependent on the length on which it was trained. Furthermore, since TFN has a TCN component, it might lose some of the resolution invariant properties of the FNO. Therefore, it is a good idea to interpolate the same number of time steps as the training data.

It is also a good idea to have relatively uniformly distributed input in time. This is because the FFT in FNO assumes that the points are uniformly distributed. Another reason is that the system will likely behave non-linearly during any time period. To resolve the non-linearity one need to avoid having big time gaps in the input.

13.8 Limitations and Potential Risks

The outputs given by the Transuranus are good enough for practical purposes, however, there are some deviations compared to some of the measured quantities. Therefore, the predictions of the trained networks cannot recover the real data that is being measured at the reactor.

Another approximation that has been made is that the training data is an approximation of the raw output of the TU. For instance, there are still some minor non-linearity in the data that the linear interpolation cannot recover. There is also no guarantee that the preprocessing technique truly captures all the essential points from the TU. Any error that occurs at the preprocessing step will propagate to the network prediction as a systematic error. The results are thus slightly biased since the networks are not evaluated on all the original data points, but something that closely resembles it. Smaller deviations such as the integral fractional gas release plot in figure 4 between the time interval 10000 to 13000 may occur.

The interpretability of the attention matrix assumes that the model is well-trained, only then will the attention weights accurately represent the relations between the inputs and outputs.

The training dataset is relatively small, which hinders the network's capacity to extrapolate to fundamentally dissimilar input LHGRs. Furthermore, there is a notable chance of the networks overfitting the data due to the limited variety of LHGRs and corresponding outputs. However, this was not apparent during training, it may be a probable outcome of the restricted diversity.

There was a unit-conversion error that occurred when input files to TU were created. As a consequence, the magnitude of the neutron flux was approximately ten orders of magnitude lower than what it was supposed to be. As a result, the network predictions are more or less independent of the neutron flux. The current versions of the models do not have the ability to predict neutron-induced embrittlement, neutron-induced swelling, neutron-cladding interaction, and related indirect consequences involving neutrons.

14 Conclusions

In this work, a novel ensemble neural network architecture, TFN, has been designed to learn the mapping between a set of LHGRs and corresponding time-dependent variables of UO_2 fuel rods. The relevant physical quantities include:

- Fuel centerline temperature.
- Central void pressure.
- Oxidation thickness.
- Gap width.
- Hydrogen absorption.
- Integral fission gas release.
- Integral fractional gas release.

The TFN utilizes fundamental trade-offs between the time- and frequency domains to gain a more complete parameter representation than the time or frequency domain only. This allows the TFN to significantly outperform its individual components, TCN and FNO, in terms of reaching the lowest validation error, which is approximately 1% as compared to the second-best model, FNO, with approximately 1.5%. The result suggests that TFN may be considered to be a strong candidate as a surrogate model for fuel performance modeling as well as it provides a simple yet powerful idea in DL that enhances the generalization capabilities of models.

To ensure that the predictions of the TFN are reliable, an ensemble consisting of three TCNs trained with different weight initialization was constructed. This TCN ensemble is then concatenated with the TFN to form an autoencoder architecture. This allows for robust zero-shot outlier detection, which makes the TFN suitable for applications where a low false positive rate is desired.

Furthermore, a one-layered Transformer architecture has been trained to gain insights into the data. It is thus possible to further stack the Transformer on top of TFN to simultaneously achieve interpretable results while maintaining high-quality predictions.

15 Outlook

15.1 Increasing the Diversity of the Dataset

The versatility of the network is rather small, only limited to 1200 training examples. One may run more TU for more diversified data, thereby, increasing the generality. Another possible improvement is to train the network on both time-dependent and time-independent variables.

15.2 Generalizations of Temporal Frequency Network

The TFN architecture can be treated as a general recipe for building a heterogeneous ensemble that capitalizes on the fundamental trade-off in different coordinate systems. There is in general, no constraint that the architectures must be FNO or TCN nor the specific combinations of the hyper-parameters. For instance, one may use the Transformer architecture as a drop-in replacement of TCN[15]. It is also possible to construct a Transformer where the attention mechanism operates in the frequency domain. The possibilities for diverse variants of the TFN are endless.

15.3 Alternative Coordinate Systems

The TFN is currently only learning in two different coordinate systems, but it is, in general, possible to transform the data in more diverse ways. One may try out different wavelets in wavelet transform and/or Laplace transform where efficient implementations are available. Another interesting generalization would be to use fractional Fourier/wavelet transform.

15.4 Backpropagation on Encoder Prediction

After performing a forward pass with the autoencoder we may obtain poor predictions as suggested by the reconstruction error. To improve the predictions, one may backpropagate through the decoder to find the gradients of the reconstruction error with respect to the encoder output. With this information, it is thus possible to perform a gradient descent step (with an appropriate optimizer) to adjust the predicted quantities $\varphi(\mathbf{x})$. This optimization procedure does not require generalization properties since it is essentially "overfitting" the encoder output. Therefore, it will almost always[16] give better results for an appropriately

[15] This would likely further improve the capabilities of TFN considering that the specific Transformer that was used in this work performed better than TCN.

[16] The errors will drop like the typical training error, which is almost always a monotonically decreasing function.

chosen step size. Furthermore, it is possible to iteratively perform this procedure to get arbitrary accuracy. However, this is rather expensive to perform since one iteration requires a forward pass of the decoder and backpropagation.

15.5 Better Transformers

Ever since the introduction of transformer architecture [8], it had been the state-of-the-art architecture for sequence-to-sequence tasks. The main mechanism, self-attention, provides global similarities between the data in time, enabling it to learn the full-time dependencies in the data. Since then, there has been work done on fusing different architectures with transformers. The first attempt was the TFT that used LSTM as encoders and then a single multi-head-attention-layer for learning the latent space representations of the data [23].

Another improved version ITFT incorporates temporal convolutions that replace the LSTM layers in the TFT which established a new state-of-the-art [29]. A potential research direction is to combine the TFN with the interpretable multi-head attention, this could potentially result in a reasonably fast, accurate neural network architecture that is simultaneously interpretable.

References

[1] European commission, Joint Research Centre Directorate G - Nuclear Safety Security. TRANSURANUS HANDBOOK. 2019 01.

[2] Geelhood K, Luscher W, Raynaud P, Porter I. FRAPCON-4.0: A Computer Code for the Calculation of Steady-State, Thermal-Mechanical Behavior of Oxide Fuel Rods for High Burnup. 2015 09.

[3] Administration USEI. *Nuclear explained The nuclear fuel cycle*;. Last updated: July 12, 2022. https://www.eia.gov/energyexplained/nuclear/the-nuclear-fuel-cycle.php.

[4] Nuclear Power. Fuel Pellets;. Accessed February 25, 2023. https://www.nuclear-power.com/nuclear-power-plant/nuclear-fuel/fuel-assembly/fuel-pellets/.

[5] van den Oord A, Dieleman S, Zen H, Simonyan K, Vinyals O, Graves A, et al. WaveNet: A Generative Model for Raw Audio. CoRR. 2016;abs/1609.03499. Available from: http://arxiv.org/abs/1609.03499.

[6] Chi L, Jiang B, Mu Y. Fast Fourier Convolution. In: Larochelle H, Ranzato M, Hadsell R, Balcan MF, Lin H, editors. Advances in Neural Information Processing Systems. vol. 33. Curran Associates, Inc.; 2020. p. 4479-88. Available from: https://proceedings.neurips.cc/paper/2020/file/2fd5d41ec6cfab47e32164d5624269b1-Paper.pdf.

[7] Li Z, Kovachki N, Azizzadenesheli K, Liu B, Bhattacharya K, Stuart A, et al.. Fourier Neural Operator for Parametric Partial Differential Equations. arXiv; 2020. Available from: https://arxiv.org/abs/2010.08895.

[8] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention Is All You Need. CoRR. 2017;abs/1706.03762. Available from: http://arxiv.org/abs/1706.03762.

[9] Rumelhart DE, McClelland JL. In: Learning Internal Representations by Error Propagation; 1987. p. 318-62.

[10] Gärdin O. “Development of a Clad Stress Predictor for PCI Surveillance using Neural Networks”. 2016:75. Available from: https://publications.lib.chalmers.se/records/fulltext/249359/249359.pdf.

[11] Nerlander V. “Temporal Convolutional Networks in Lieu of Fuel Performance Codes: Conceptual Study Using a Cladding Oxidation Model”, Advanced Project Work in Energy Systems Engineering. 2021:26. Available from: http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-455904.

[12] Van Uffelen P, Hales J, Li W, Rossiter G, Williamson R. A review of fuel performance modelling. Journal of Nuclear Materials. 2019. Available from: https://www.sciencedirect.com/science/article/pii/S0022311518310298.

[13] Blair P. Modelling of fission gas behaviour in high burnup nuclear fuel; 2008. .

[14] Tupin M. 7 - Understanding the corrosion processes of fuel cladding in pressurized water reactors. In: Ritter S, editor. Nuclear Corrosion. European Federation of Corrosion (EFC) Series. Woodhead Publishing; 2020. p. 251-99. Available from: https://www.sciencedirect.com/science/article/pii/B978012823719900007X.

[15] Lemaignan C. 2.07-Zirconium alloys: properties and characteristics. Comprehensive nuclear materials. 2012;2:217-32.

[16] Brunton SL, Kutz JN. In: Frontmatter. Cambridge University Press; 2019. p. i-iv.

[17] Géron A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. 3rd ed. O'Reilly Media, Inc.; 2022.

[18] Kingma DP, Ba J. Adam: A Method for Stochastic Optimization; 2017.

[19] Lässig F. *Temporal Convolutional Networks and Forecasting* ; Oct 2020. https://medium.com/unit8-machine-learning-publication/temporal-convolutional-networks-and-forecasting-5ce1b6e97ce4.

[20] Bai S, Kolter JZ, Koltun V. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. CoRR. 2018;abs/1803.01271. Available from: http://arxiv.org/abs/1803.01271.

[21] Fort S, Hu H, Lakshminarayanan B. Deep Ensembles: A Loss Landscape Perspective. arXiv; 2019. Available from: https://arxiv.org/abs/1912.02757.

[22] Wolpert D. Stacked Generalization. Neural Networks. 1992 12;5:241-59.

[23] Lim B, Arik SO, Loeff N, Pfister T. Temporal Fusion Transformers for Interpretable Multi-horizon Time Series Forecasting. arXiv; 2019. Available from: https://arxiv.org/abs/1912.09363.

[24] Liu J, Song Y, Wang L, Dang J, Yu R. Time-Frequency Representation Learning with Graph Convolutional Network for Dialogue-Level Speech Emotion Recognition. In: Proc. Interspeech 2021; 2021. p. 4523-7.

[25] Piczak KJ. Environmental sound classification with convolutional neural networks. In: 2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP); 2015. p. 1-6.

[26] Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, et al.. Going Deeper with Convolutions; 2014.

[27] Hoffmann J, Borgeaud S, Mensch A, Buchatskaya E, Cai T, Rutherford E, et al.. Training Compute-Optimal Large Language Models; 2022.

[28] Touvron H, Lavril T, Izacard G, Martinet X, Lachaux MA, Lacroix T, et al. LLaMA: Open and Efficient Foundation Language Models. arXiv preprint arXiv:230213971. 2023.

[29] Feng G, Zhang L, Ai F, Zhang Y, Hou Y. An Improved Temporal Fusion Transformers Model for Predicting Supply Air Temperature in High-Speed Railway Carriages. Entropy. 2022;24(8). Available from: https://www.mdpi.com/1099-4300/24/8/1111.

[30] Hendrycks D, Gimpel K. Bridging Nonlinearities and Stochastic Regularizers with Gaussian Error Linear Units. CoRR. 2016;abs/1606.08415. Available from: http://arxiv.org/abs/1606.08415.

16 Appendix

16.1 Training Specifications

The TCN, FNO, TFN, and Transformer were trained for 10000 epochs with a learning rate of 10^{-3} with an Adam-optimizer. The inference time is measured by averaging the times of 100 runs, the computation was performed on a batch with size 1200 on an Intel Xeon CPU 2.20GHz. Note that the inference time may be decreased by an order of magnitude using an NVIDIA TESLA P100 GPU (free tier GPU in Kaggle).

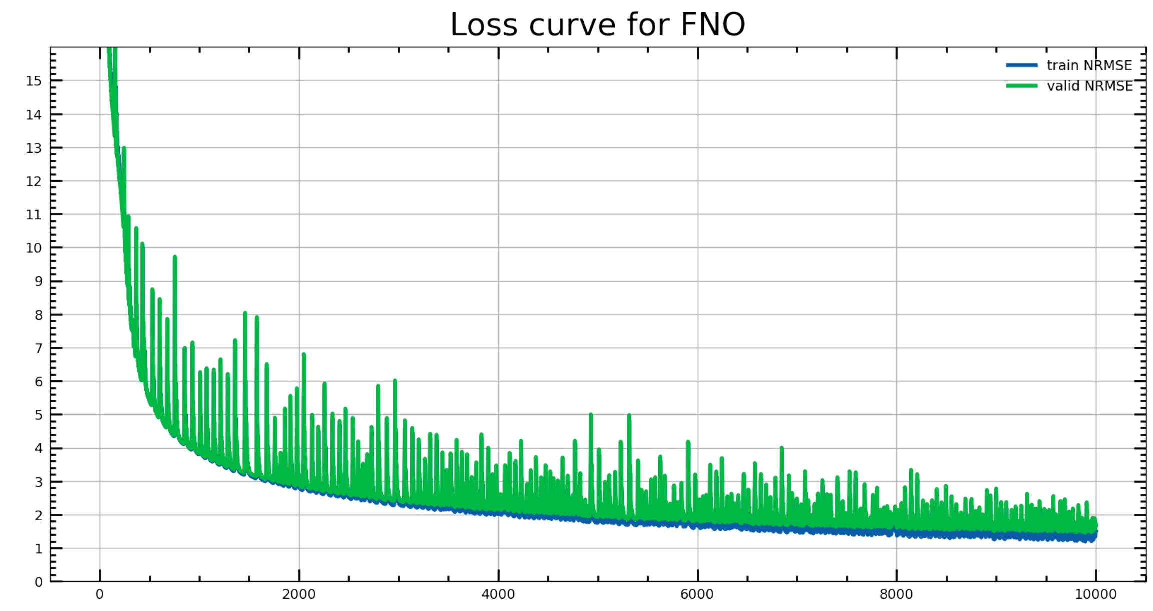
Architecture	Channels	frequency modes	layers
FNO	100	16	4

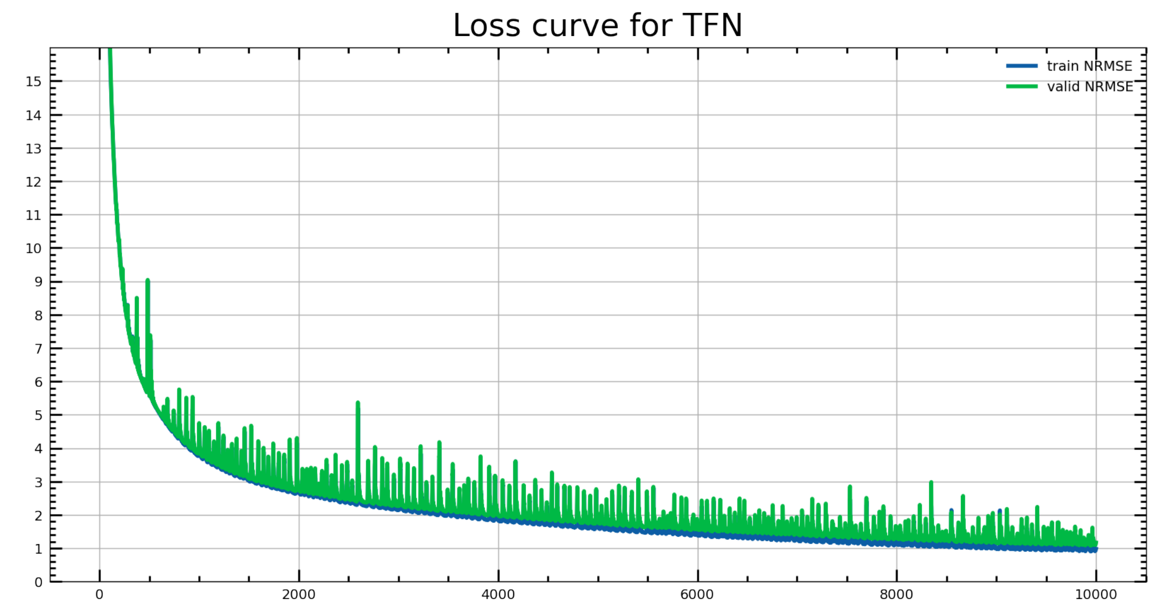
Table 1: Choice of hyperparameters for FNO and TFN.

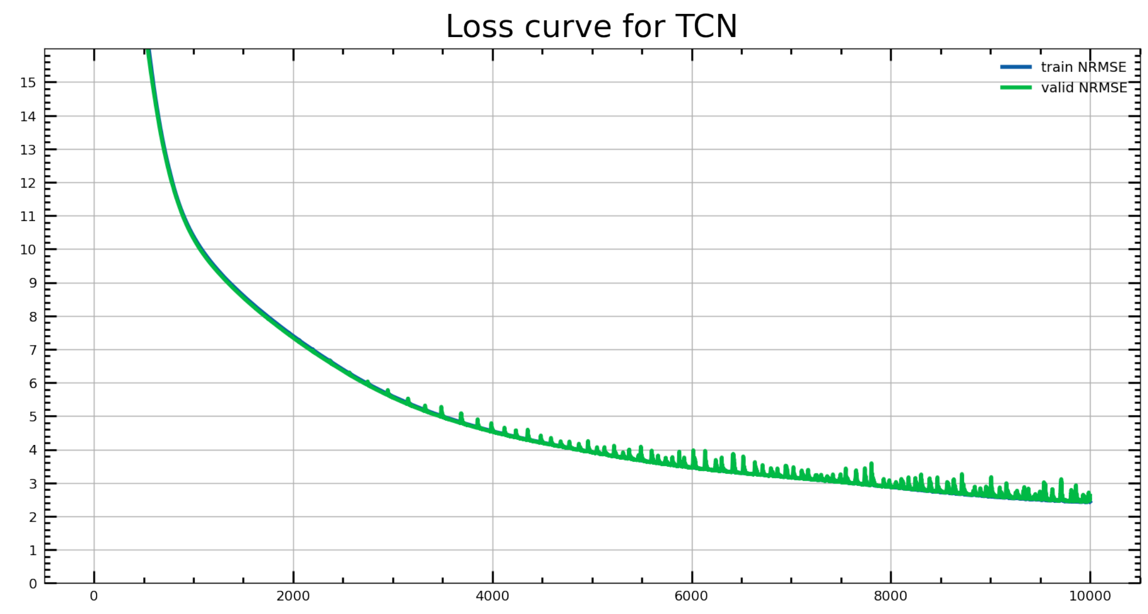
Architecture	Weights	Inference time [ms]
TCN	5334	127 ± 7
FNO	872941	3130 ± 39
TFN	876691	2880 ± 28
Transformer	129104	7680 ± 190

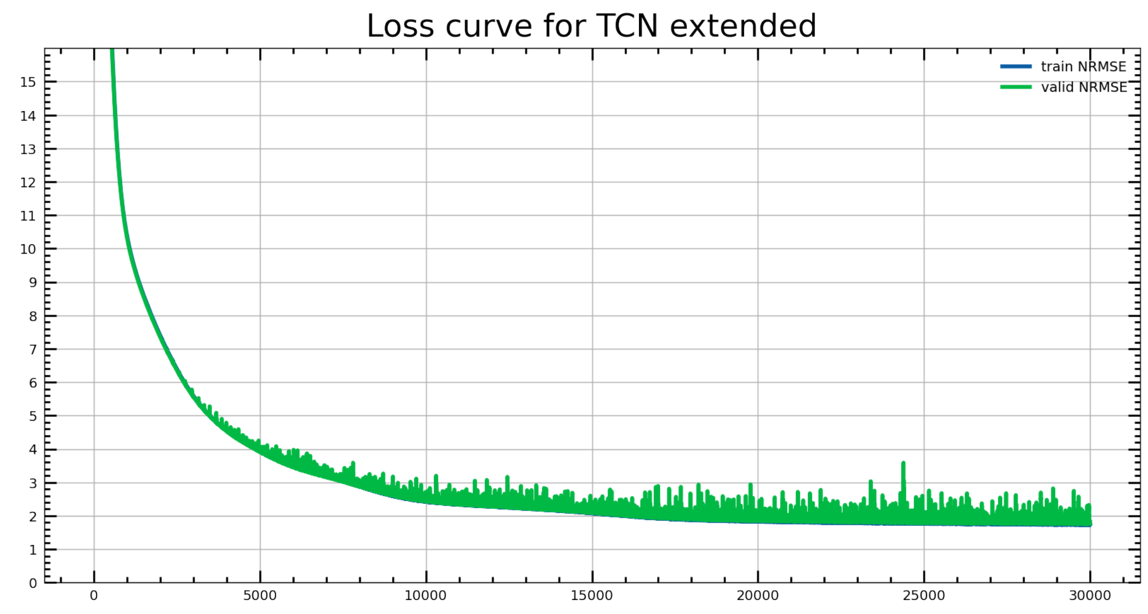
Table 2: Weights and inference time for different encoder architectures.

The figures below show the loss curves of the different architectures. The y-axis represents NRMSE and the x-axis represents the number of epochs.









16.2 Performance for TCN Decoder Ensembles

The TCN were trained separately with the following weight initialization:

1. Kaiming uniform (PyTorch default).
2. He normal.
3. Xavier normal.

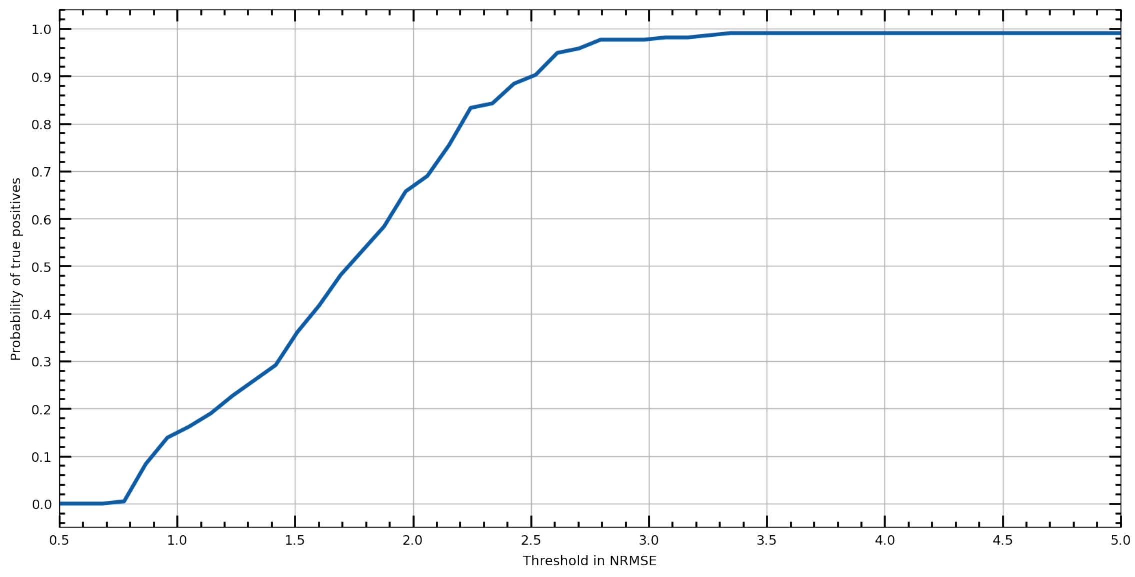


Figure 24: Three TCNs with different weight initialization have been used to reject potential negative cases from a TFN encoder. From this, the probability of getting true positive cases as a function of the error threshold is obtained.

16.3 Implementing the Fourier Neural Operator

The implementation of the FFC-block is very much based on the FNO implementation [7] with some details changed[17], the overall structure is described as follows:

1. Permute the time and feature dimension of the input data to prepare for the PyTorch convolution operation.
2. Perform a point-wise convolution operation that transforms the feature dimension to the hidden dimension:
$$\mathbf{x}_2 = \mathrm{Conv}(\mathbf{x}_1). \tag{35}$$
3. Perform a point-wise convolution that preserves the hidden dimension:
$$\mathbf{x}_3 = \mathrm{Conv}(\mathbf{x}_2). \tag{36}$$
4. Perform FFT on the data in the time domain.
$$\boldsymbol{\omega}_2 = \mathcal{F}(\mathbf{x}_2) \tag{37}$$
5. Truncating the high-frequency modes, the first 16 modes are kept.
$$\boldsymbol{\omega}_2 \rightarrow \tilde{\boldsymbol{\omega}}_2 \tag{38}$$
6. Perform a batched matrix multiplication over the hidden feature dimension followed up by an element-wise multiplication for the frequency modes. This operation is denoted with Einstein notation and the result can be interpreted as batched matrix multiplication weighted by the frequency modes:
$$\tilde{\boldsymbol{\omega}}_{3,box} = \tilde{\boldsymbol{\omega}}_{2,bix}\mathbf{W}_{\omega,biox}. \tag{39}$$
7. Inverse transforms the truncated frequencies back to the time domain.
$$\tilde{\mathbf{x}}_3 = \mathcal{F}^{-1}(\tilde{\boldsymbol{\omega}}_3) \tag{40}$$
8. Compute a linear combination of the results from the convolution and the output from the inverse FFT
$$\mathbf{x}_4 = 2\left[(1-\alpha)\mathbf{x}_3 + \alpha\tilde{\mathbf{x}}_3\right] \tag{41}$$
where $0 \leq \alpha \leq 1$ is a learnable parameter.

[17]In the original implementation, 7. $\alpha = 0.5$, 8. $\alpha_1 = \alpha_3 = 1$, $\alpha_2 = 0$.

9. Perform a composition of convolutions with kernel size 1 and activation function Gaussian error linear unit (GELU) [30] in the time domain.

$$\mathbf{x}_4' = \mathrm{GELU} \circ \mathrm{Conv}(\mathbf{x}_4)$$
$$\mathbf{x}_4'' = \mathrm{GELU} \circ \mathrm{Conv}(\mathbf{x}_4')$$

   Weight the results obtained after each operation for some weights $0 \leq \alpha_i \leq 1$.

$$\mathbf{x}_5 = \alpha_1 \mathbf{x}_4 + \alpha_2 \mathbf{x}_4' + \alpha_3 \mathbf{x}_4'' \tag{42}$$

The FNO architecture consists of four FFC blocks with a point-wise convolution and dimensionality permutation in the end:

$$\mathrm{FNO}(\mathbf{x}) = (\mathrm{Conv} \circ \mathrm{FFC}_4 \circ \mathrm{FFC}_3 \circ \mathrm{FFC}_2 \circ \mathrm{FFC}_1)(\mathbf{x}) \tag{43}$$