Using a surrogate model for the detection of defective PWR fuel rods

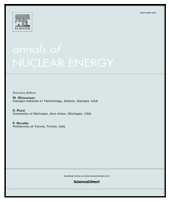
Karine Chevalier-Jabet, Lokesh Verma, François Kremer

Annals of Nuclear Energy 209 (2024) 110779

Contents lists available at ScienceDirect

Annals of Nuclear Energy

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



Using a surrogate model for the detection of defective PWR fuel rods

Karine Chevalier-Jabet *, Lokesh Verma, Francois Kremer

PSN-RES/SAM/LETR, Institut de Radioprotection et de Surete Nucleaire, Cadarache, 13115, St. Paul les Durance, France

ARTICLE INFO

Keywords:
Fission product
Defective fuel
Surrogate model
Artificial neural network
Autoencoder

ABSTRACT

Timely and accurate detection of defective fuel rods is critical as the release of radioactive fission products from defective fuels can lead to primary circuit contamination and radiation exposure. Due to the complexity of the physical phenomena, models for fault diagnosis can be difficult to construct and recently data driven surrogate models have being increasingly used to detect and characterize defective fuel rods: they make use of a computational database to learn from and make predictions about new unknown data. In this paper, we present a method for the elaboration of an anomaly detector based on neural networks, taking into account the fact that physical computation can be CPU intensive and thus overcome this issue. A physical model for fission products release and coolant activity calculation was built and used to generate a surrogate activity model that enables the generation of a bigger database in small amount of CPU times. Then using this bigger computational database, a recurrent autoencoder was trained for anomaly detection. The network classifies the defect status with 100% accuracy and a good time precision. A sensitivity analysis with lower activity increase at defect onset and addition of noise was conducted in order to better understand the limits of this method. Such methods can be useful for operators of the existing as well as future reactors to make timely predictions of defective fuel rods and avoid operational and economic setbacks for power plants. The work described in this paper was carried out within the R2CA (Reduction of Radiological Consequences of design basis and extension Accidents) project, funded in HORIZON 2020 and coordinated by IRSN (France).

1. Introduction

In a Pressurized Water Reactor (PWR), a defective cladding can cause an increase in the coolant activity and can lead to increasing primary circuit contamination and radiation exposure. Although the probability of fuel failures has been reduced to low levels (~ 0.004% for PWRs (IAEA (2019))), it is still of utter importance to timely detect any fuel failure that may occur during operations since operating a reactor under defective conditions may lead to not only operational and economic setbacks for power plants due to early discharge of the involved fuel assemblies, but also radiological and environmental hazards.

In recent years, with the progress in machine learning and deep learning approaches, data-driven methods have gained momentum for fuel defect detection. Neural networks can be very efficient tools to make predictions about defective fuels. A general review of the state-of-the-art can be found in Abiodun et al. (2018). Likhanskii et al. (2006) used Artificial Neural Networks (ANN) for fuel failure detection in VVER reactors. Wallace et al. (2020) recently demonstrated the application of neural network models for defect defection in the CANDU fuels. The approach was found to be faster than the existing processes. Dong et al. (2020) used an ANN method for fuel failure detection with the input of the ANN being the specific activities of FP in the primary coolant and the output being the degree of failure of the fuel cladding.

One of the drawbacks of simple ANN used as classifiers for defect detection is their lack of generality : the database has to be reasonably balanced, with well represented defective situations ; would the method be applied on real life data, defective situation would be less numerous, and missclassification could emerge. Moreover, it would be better if time, an important information was used, so that defect detection rely also on some “change” in the observations.

On the other hand, novelty detection techniques are often used when the quantity of available normal data is much higher than the quantity of abnormal ones. These methods are used in various industrial fields such as electronic IT security, medical diagnostics, industrial monitoring and damage detection or video surveillance.

Pimentel et al. (2014) have grouped novelty detection techniques in four categories : probabilistic, distance assessment based (Liu et al. (2008)), domain based (see Cortes C. (1995) for example), information-theoretic and reconstruction techniques. On all these techniques, reconstruction has the advantage of versatility : a model of the normal data is built, the novelty being based on the reconstruction error. This kind

* Corresponding author.
E-mail addresses: karine.chevalier-jabet@irsn.fr (K. Chevalier-Jabet), lokesh.verma@irsn.fr (L. Verma), francois.kremer@irsn.fr (F. Kremer).

https://doi.org/10.1016/j.anucene.2024.110779
Received 9 January 2024; Received in revised form 20 May 2024; Accepted 9 July 2024
Available online 5 August 2024

of tools is independent of the distribution but assumes that normality has been fully sampled before learning, and they may perform poorly when outside the learning range, leading to false positive. Example tools of that class are gradient boosting techniques (Friedman (2001)), or dedicated artificial neural networks . Some of them are detailed hereafter, as they have been subject of numerous developments in the two last decades.

One class neural networks (Chalapathy et al. (2018)) only learn the properties of one class and treat any data point that deviates significantly from this as an anomaly.

Deep belief neural networks, convolutional neural networks are especially effective for anomaly detection in images, where they can learn to identify normal patterns from training data and then recognize abnormalities in new images. They are also effective at representing time sequences of variables and for example have been used by Microsoft (see Ren et al. (2019)) to monitor the metrics of their applications and services and detect anomalies.

Restricted bolzman machines (Hinton (2012), Hinton et al. (2015), Fiore et al. (2013)) consist of two layers of neurons — visible units and hidden units. The visible units represent the input data, while the hidden units capture the underlying patterns and features in the data. Each visible unit is connected to every hidden unit, and vice versa. RBMs are based on function that assigns an 'energy' value to each possible configuration of the visible and hidden units. The goal is to find the configuration with the lowest energy, which corresponds to the most probable state of the RBM. The training is performed using a process called Contrastive Divergence (a method related to MCMC methods), in order to reconstruct the input data by adjusting the connection weights between the visible and hidden units.

Generative adversarial networks (GAN) (Goodfellow et al. (2014)) can be used for anomaly detection by training them to generate normal data. Any data that the GAN has difficulty generating could potentially be viewed as an anomaly. This is specifically useful in scenarios like detecting anomalies in images or in any data that requires generative models to make a normality distinction. GAN use two networks, a generator, and a discriminator. The generator generates new data instances, while the discriminator evaluates them for authenticity. By comparing the generated instances to the original ones, it can learn to differentiate anomalies from normal instances; an example of fault detection can be found in the field of building and indoor environment surveillance by Yan et al. (2020).

Autoencoders are a type of neural network used for learning efficient coding of input data. They work by encoding the input data into a compressed representation, and then decoding this representation back into the original format. Kieu et al. (2019) have used autoencoders based on recurrent neural networks (RNN) in order to perform time related fault detection and compared them to a set of traditional methods. The study they conducted show that the proposed autoencoder ensembles are effective and outperform all baselines methods. The same conclusion prevailed for Alfeo et al. (2020).

As RNN allow time representation and can easily be fed to autoencoders, it may be useful to check other common related types of RNN : Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber (1997)) neural networks, and Gated Recurrent Units (GRU) networks (Cho et al. (2014)), which are popular because known to be effective.

In this paper, they will be fed to autoencoders and tested for defect detection based on timely primary coolant measurements.

In order to produce activity sequences, a large database was needed ; a fast model was used so that the generation could be done in a reasonable amount of time : an ANN model of activity computation due to defect onset was built, based on a simple physical model that was developed previously by Verma et al. (2023). The physical model is briefly presented in a first section of this paper, but the reader can refer to Verma et al. (2023) for detailed information. The ANN model elaboration and validation is described in Sections 3.2 and 3.3. The whole database generation methodology is presented in Section 3.

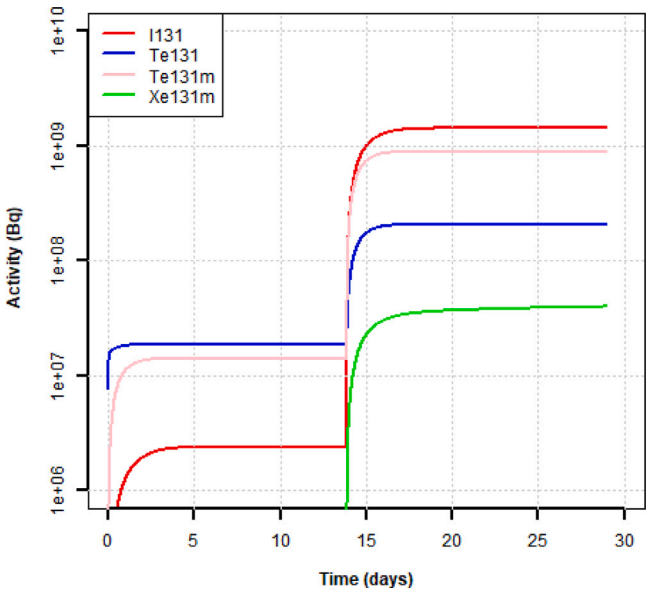


Fig. 1. Activity of various isotopes in the primary coolant after defect onset.

Section 4 presents the architectures that were used for default detection, and the performance associated to each of them. A sensitivity analysis was performed to simulate small defects and noisy measurements. Finally the conclusions of the analyses done in this work, some limitations and the prospects for future work are presented in Section 5.

2. Physical model and activity calculation

The fission products (FP) release estimation from a defective fuel rod has be subdivided into three steps: (i) transport of FP in the fuel pellet, (ii) transport in the fuel-cladding gap, (iii) transport in the coolant. The transport mechanisms in the three regions and the corresponding mass balance equations have been presented in detail in an earlier paper by the authors (Verma et al. (2023)), as well as the verification from literature. Although the model can take in 8 decay chains comprising a total of 30 radioisotopes, for this analysis, we considered the I-131 decay chain whose properties are presented in Table 1 and extracted from JEFF 3.3 (2017).

$$^{131m}Te \longrightarrow {}^{131}Te \longrightarrow {}^{131}I \longrightarrow {}^{131m}Xe$$

The parameters used in the release model for activity calculation and their values are presented in Table 2. We suppose that we are in stationary nominal operation conditions, so operational parameters are supposed to be constant during all the calculation. The typical operating values of a PWR are used for the fuel and gap parameters. The remaining parameters (especially concerning the coolant region) were adopted from Iqbal et al. (2008).

The model produces the results which would be deemed necessary for generating the computational database, and an example is shown at Fig. 1; the total simulated time was set at approximately 28.9 days. The time step for the time integration was taken as 40.0 s. The activities in the primary coolant before defect onset are due to the residual tramp uranium contribution. As soon as the defect occurs, the activity values rise for most of the isotopes and reach a saturation state for all the isotopes within the duration of the calculation as the source of radioisotopes gets balanced by their decay or by removal through the CVCS purification system in the primary coolant.

Table 1
Properties of the radioisotopes considered in the analysis.

isotope	location in the chain	decay constant (s^{-1})	fission yield	$frac_{12}$	$frac_{13}$	$frac_{23}$	$frac_{24}$	$frac_{34}$	$frac_{35}$	$frac_{45}$
Te-131m	1	6.42e−06	0.00504	0.21	0.79	0.0	0.0	0.0	–	–
Te-131	2	4.62e−04	0.0252	0.0	0.0	1.0	0.0	0.0	–	–
I-131	3	1.0e−06	0.02921	0.0	0.0	0.0	0.0	0.019	–	–
Xe-131m	4	6.71e−07	0.000317	0.0	0.0	0.0	0.0	0.0	–	–

Table 2
The nominal values of model parameters used in the analysis.

Parameter	symbol	Name used	value	units
fuel stack length	l	–	0.168	m
average linear power	P_{lin}	power	22.77e+03	Wm^{-1}
width of fuel-to-sheath gap	d_{gap}	gap_width	24.32e−06	m
radius of pellet	R_{pel}	radius_pel	4.15e−03	m
gap internal pressure	P_{int}	Pressure_int	15.5e+06	Pa
fission rate	F	–	3.03e+10	fissions $W^{-1}s^{-1}$
location of cladding defect	l_d	defect_loc	0.084	m
resin efficiency for Te	η_{Te}	resin_eff_Te	0.9	–
resin efficiency for I	η_I	resin_eff_I	0.99	–
resin efficiency for Xe	η_{Xe}	resin_eff_Xe	0.9	–
Boltzmann constant	k	–	1.38e−23	JK^{-1}
release rate of H2	q_{H2}	release_rate_H2	1e+21	$m^{-3}s^{-1}$
letdown flow rate	Q	letdown_flow_rate	3.0	kgs^{-1}
total coolant mass	M	tot_coolant_mass	1.07e+06	kg
BRS removal rate	B	BRS_removal_rate	1e−05	s^{-1}
coolant leak rate	L	coolant_leak_rate	2.3e−03	kgs^{-1}
Temperature of ext surface of cladding	T_{cl}^{ext}	Temp_cl_ext	597.14	K
Avg. temperature of pellet	T_{pel}	Temp_pel	1261.05	K
Time of defect onset		t_defect	10	days

3. Computational database generation

Due to CPU time considerations, the database production comprises 3 steps: at first, a database of physical calculation is generated, which leads to the production of a surrogate model for the activity in case of defect or in case of absence of defect, so that we are able to generate defect sequences in a small amount of time; next, the sampling method to feed these surrogate models is built so that bigger samples are generated, but yet remain consistent with the physical sample that was used to build the ANN; and at last, the database generation is performed, calling the surrogate activity models to generate the sequences.

3.1. Physical database computation

In the physical model, we identify 16 input parameters (see Fig. 2) which can be varied to generate the sample data. The number of input parameters determines the dimension and the complexity of our problem. If we wanted to fully sample each dimension of space (full factorial design sample), the size of the resulting database would be a function of the number of parameters raised to the power of the number of sampling points. Thus, for a complete factorial design with only three points (lower bound, center, upper bound, for example) and 16 parameters, we would need to perform $16^3 = 4096$ calculations, and adding one sampling point would result in 65 536 calculations. Assuming one calculation takes approximately one minute, this would correspond to computation times of approximately 4 days or 7 weeks, respectively.

The use of the Latin Hypercube sampling (LHS) method was preferred instead, as it guaranties that, given the size $\mathbf{n}$ of the sample, every $\mathbf{n}^{th}$ probability value within the probability distribution will be sampled for sure, but without guaranteeing that the most interesting combination of all values will be used as they are random.

Building a precise enough surrogate model based on such a database enables us to multiply our computational capabilities by a factor of several hundreds, and enables the generation of large bases of sequences for the final goal of the project, i.e. defect detection. The surrogate model is an Artificial Neural Network (ANN), and was built using the physical database that was generated as follows :

- The data sampling for the possible input variables related to the release are done by using the Transuranus code (Lassmann, 1992) in the statistics mode. By providing the absolute values of parameters, such as the linear power and the outer pin pressure, and varying them in an a priori range of 10% in a uniform distribution, we can obtain the equivalent values for parameters such as the pellet center-line temperature, the cladding external temperature, the radius of the pellet and the gap width as outputs. The input parameters obtained by the Transuranus code are the six parameters in Fig. 2(a). We generate 2000 samples for these six parameters using the Transuranus code.
- For the remaining 10 input parameters of the model (Fig. 2(b)), we have used an in-house tool, comprising a collection of R (R Core Team, 2022) scripts, to carry out the sampling of the data. For these parameters, we varied the absolute values, a priori, in the range of 10%, using uniform distributions. The size of the sample is 2000, and the LHS algorithm was used for this part of the sampling.
- The assembled 2000 samples, comprising different set of values for the 16 input parameters, are provided to the physical model, leading to 2000 calculations of coolant activities through time of the four isotopes in the decay chain. Fig. 3 presents the data flow charts for the production of the physical database.

3.2. A meta model for activity prediction in case of defect

In this section we present the ANN : 80% of the physical computational database was used for training, and the rest of the database was set aside for testing. The development of this ANN, and of all the neural networks of this paper, is done in Python, using the Keras API with TensorFlow backend Abadi et al. (2015).

Prior to the learning phase, the data are reprocessed as below:

(a)

Parameter name	Absolute value *(in corresponding SI units)*	Probability law	Range (fraction)	
			Minimum	Maximum
power	$22.77*10^{3}$	uniform	0.9	1.1
Pressure_int	$15.5*10^{6}$	uniform	0.99	1
radius_pel	$4.15*10^{-3}$	-	0.989	1.00
Temp_pel	1261.05	-	0.837	1.181
Temp_cl_ext	597.14	-	0.973	1.027
gap_width	$24.32*10^{-6}$	-	0	2.39

(b)

defect_loc	0.084	uniform	0	2
release_rate_H2	$1.0*10^{21}$	uniform	0.9	1.1
letdown_flow_rate	3.0	uniform	0.9	1.1
tot_coolant_mass	$1.07*10^{6}$	uniform	0.9	1.1
resin_eff_Te	0.9	uniform	0	1.1
resin_eff_I	0.99	uniform	0	1
resin_eff_Xe	0.008	uniform	0	123.7
BRS_removal_rate	$1.0*10^{-5}$	uniform	0.9	1.1
coolant_leak_rate	$2.3*10^{-3}$	uniform	0.9	1.1
t_defect	$8.64*10^{5}$	uniform	0	2.89

Fig. 2. (a) Input parameters sampled using Transuranus statistics code; (b) The sample file used to generate the sample data for the remaining parameters.

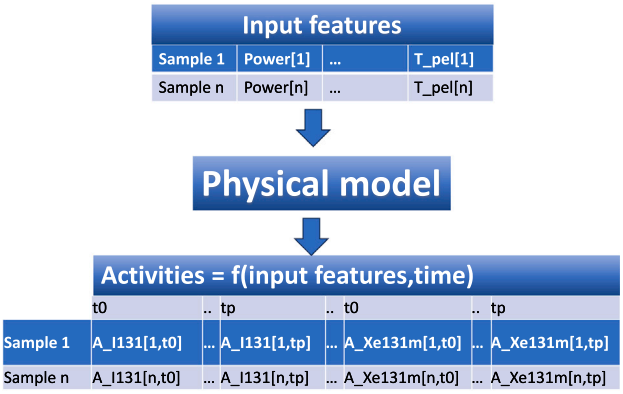


Fig. 3. Data flow chart for physical computations.

1. For each input feature, the bi-dimensional sequences of value activities of shape (number of time steps of the sequence of activities,number of samples) are flattened into one dimensional data arrays of shape (1,number of time steps of the sequence of activities × number of samples).
2. The times, that are output of the physical database, have then been added as input features and converted into relative time since the occurrence of the defect, with negative values of times meaning that defect has not occurred yet, and positive meaning the time elapsed since defect onset. For the modeling of the activities due to defects, only positive values of times have been kept. Since activity and time range of variation span multiple decades, their values have been transformed to their logarithms. In the end, the size of the sample is about 1.5e6, with 2000 different values of physical parameters, and 1395 positive time steps.
3. The data have been separated in two sets : the learning set, composed of 80% of the data, and the test set, composed of the remaining portion. The test set is completely independent of the learning set, as it is never used in any part of the learning.
4. Then, in order to facilitate the learning phase, the data have been normalized between 0 and 1. The data flow chart of this process in presented in a graphical manner at Fig. 4.

The optimal architecture of the neural network is not known a priori, but Tensorflow allows, through its Keras tuning features, to determine fastly the most interesting one among a set of propositions. For this purpose, the following hyperparameters were used and tuned using the Hyperband tuner: the number of layers (between 2 and 4), the number of cells per layer (between 64 and 128, by steps of 32). The loss function was mean absolute error, learning rate was 0.001. To prevent over-fitting, L2 regularization was used with values 2e-4 for the first layer, and 2e-7 for the following ones.

After the tuning, the best model has 4 hidden layers of (128,96,96,64) cells. The predictions of the test set, equivalent to about 400 time sequences, were performed in 9 s on a standard laptop, which would have taken about 6 h on a calculation cluster using the real physical model.

Fig. 5 shows the entire predictions set versus the test dataset. To sum up, the maximum relative error of the model is about 2.6%, the 99th percentile of the absolute error being about 0.3%, which is satisfactory.

3.3. A meta model for activity prediction in absence of defect

In this section we present a meta model for the prediction of activities when there is no defect. The data were processed as in previous section, except that only negative values of time were kept, and that duplicates in the database had to be removed due to absence of activity evolution : in such a case, the activities are due only to the presence of tramp uranium, and activities are at equilibrium if no change is done to primary operating conditions. The parameters that are relevant are the various primary flow rates (inlet, outlet), and the purification efficiencies for the various nuclides. As this situation is quite simple, a simple lasso method, plus order 2 polynomial features of the input features were used. The result is precise, as depicted in Fig. 6.

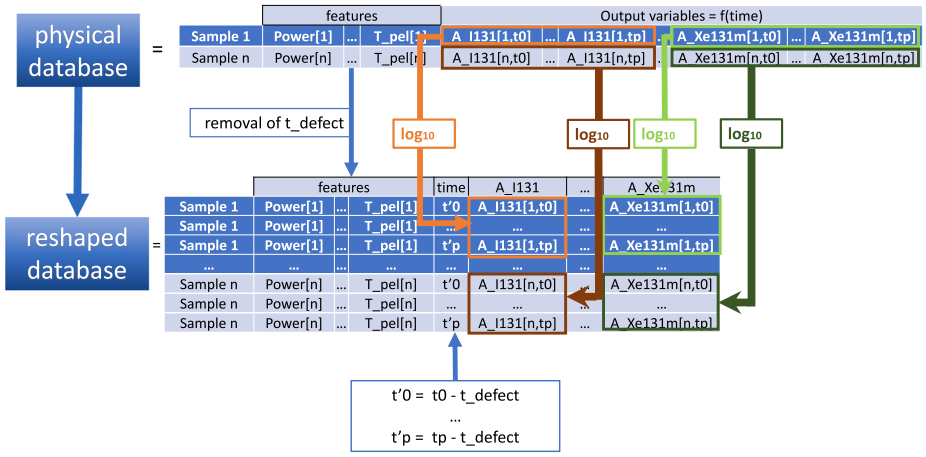


Fig. 4. Data flow chart for physical computations.

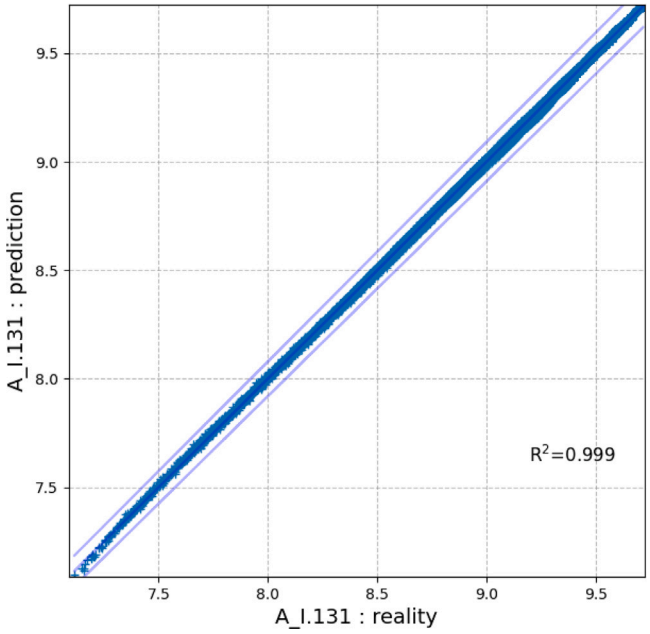


Fig. 5. Outputs of I131 meta model (prediction) vs outputs of the physical model (reality) in case of defect. The space between the lines above and below the bisector correspond to the ±1% error range.

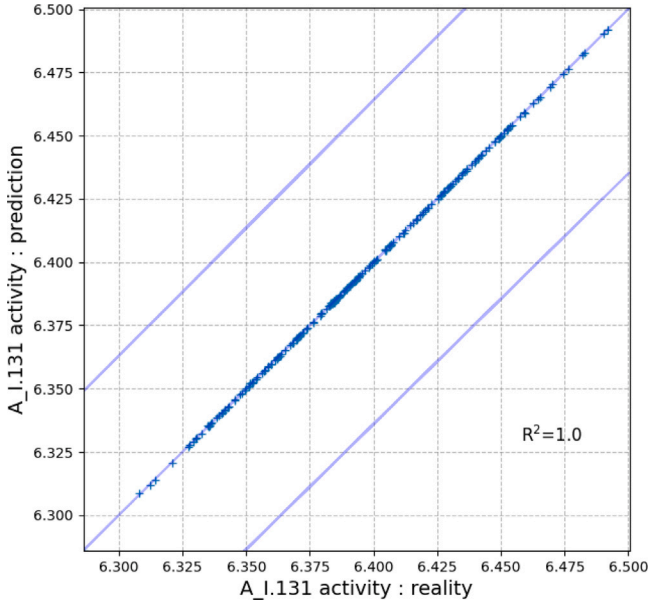


Fig. 6. Outputs of I131 meta model (prediction) vs outputs of the physical model (reality) in absence of defect. The space between the lines above and below the bisector correspond to the ±1% error range.

3.4. Sample generation for the ANN surrogate models

The sample to be fed to these surrogate models has to comply with the original physical database in order to stay in the validation domain of the models. For the original samples, LHS was used for all the data, except the ones that are related to the release from fuel. These latter variables are correlated (as presented in the heatmap in Fig. 7) so we can not use LHS for sample generation related to this part of the model. The heatmap is a graphical representation of the correlation matrix of fuel related input features. The two values in the squares are respectively the correlation coefficient and the associated *p*-value of the correlation coefficient. We can see, for example, that pellet radius and gap width are unsurprisingly strongly correlated. We can also see that pellet temperature is strongly correlated to power and pellet radius.

A good workaround solution is trying to get back to a situation where all features are independent so that we can use LHS or any other method where no dependency is required. In that regard, a Principal Component Analysis (PCA) is conducted on the fuel related input features. All the fuel related features were fed to the scikit-learn (Pedregosa et al. (2011)) PCA analysis tool: the result is that we can represent this 8 dimensional system assuming only 6 independent components, keeping more than 99.8% of the variance. This allows to sample these components using LHS, and then back-transform them to their original variable values using the PCA inverse model. The PCA inverse model was validated on the physical test database, giving correct results, that are depicted in Fig. 8 for linear power, as an example.

The generation of activity sequences results then in a several steps process:

1. Choose the size of the sample; we have chosen to generate 50 000 sequences for training
2. LHS sampling for primary coolant related variables, the six independent components related to fuel release, and the beginning times of the sequences relative to the onset of a defect. The values of the upper and lower bounds are the same as the ones observed for the input features (or the PCA transform) of the sequences generated with the physical model. The time steps of

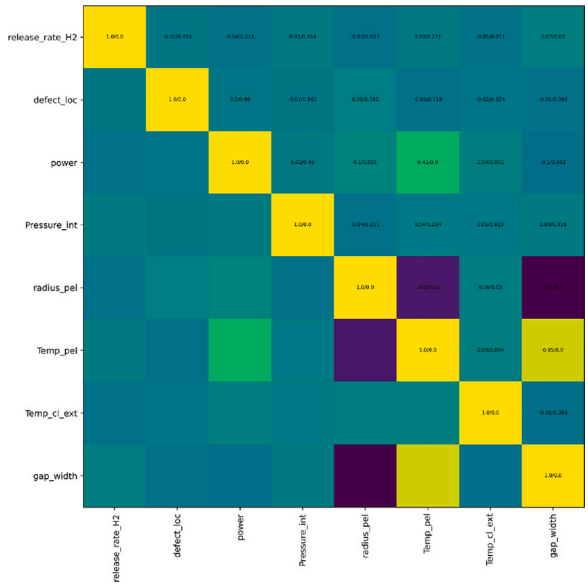


Fig. 7. Heat map of the fuel related input features.

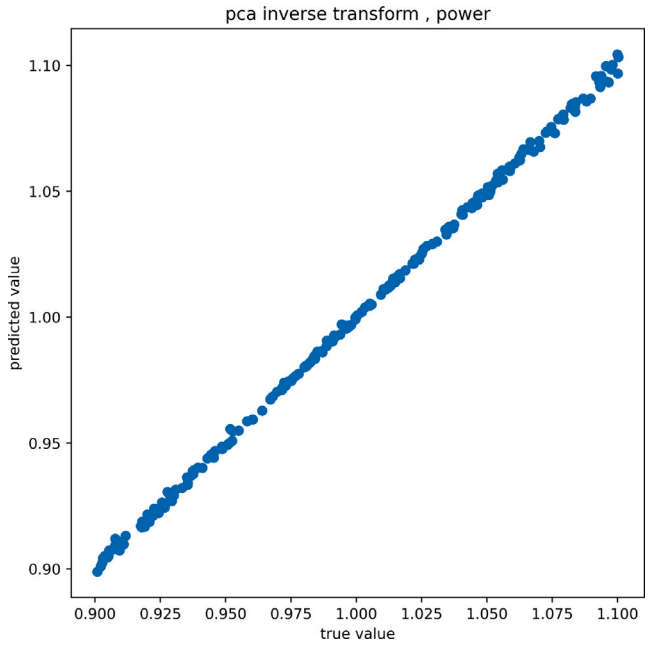


Fig. 8. Outputs of PCA inverse metamodel (prediction) vs real values (test values).

the outputs of the physical model is 1000 s. We assume here that the measurement frequency is lower; it has been decided that time steps ranging from one to 24 h should be explored, leading to 5 different database with time steps of 1, 2, 6, 12 or 24 h. In real life, it would be better to be able to detect an anomaly the sooner after its onset. Moreover, it would be preferable to avoid exposure to eventual missing measurement in the past. For this reason, the anomaly detector should rely on the shortest possible sequences. A database of sequences of 3 time steps was generated, and also another one of 4 time steps. Considering the beginning time of the sequence, it has been sampled in various ranges: such that only negative times compose the set of sequences, or such that either positive or negative values compose the set of sequences, to simulate sequences

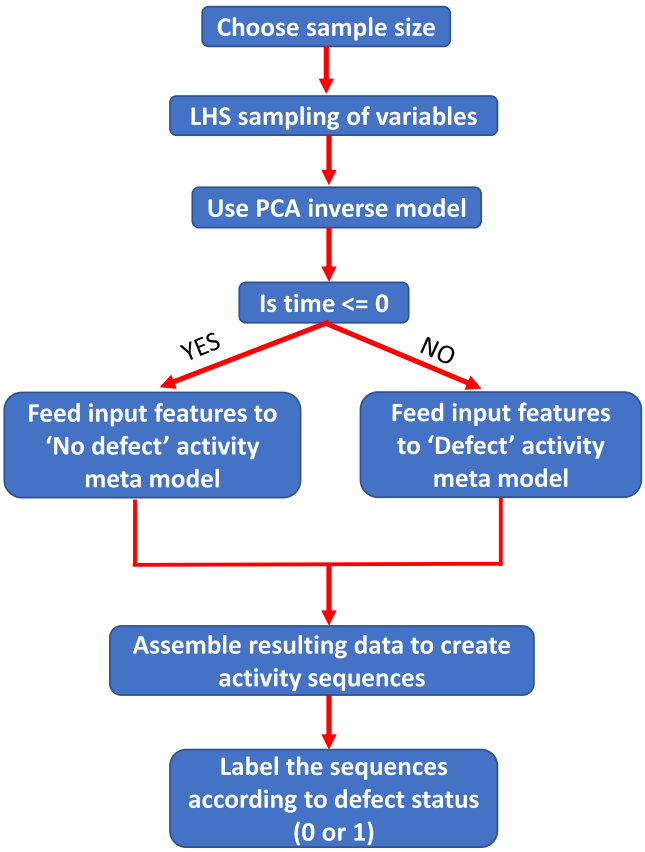


Fig. 9. Flow chart depicting the steps for generation of activity sequences.

starting with no defect, and eventually a defect opening leading to activity evolution(see Section 4).

3. Use PCA inverse model to produce the fuel related features
4. If time is lower or equal to zero: feed the input features to the no defect activity model
5. If time is greater that zero: feed the input features to the defect activity model
6. Assemble the resulting data to create the activity sequences; some sequences can therefore be composed of defective activities only, or non defective activities, or both if the defect onset occurs during the sequence
7. Label the sequences according to the defect status: if no defect occurs (no sampled time was greater than zero), the sequence is labeled with value 0, and with 1 otherwise.
   This finally results in 5 × 2, i.e. 10 sets of 50 000 training sequences.

The above mentioned algorithm steps are depicted as a flow chart in Fig. 9.

4. Defect diagnostics

4.1. The principle of defect detection

In this section, we show the results of various neural network tools used to answer the question: 'Considering the last k values of primary coolant activity, could we say that there exists a defect (1) or not (0)?'

As the activities are time related, we use neural network tools designed for that kind of data: Recurrent Neural Networks (RNN) and their descendants, Long Short Term Memory (LSTM) neural networks (Hochreiter and Schmidhuber (1997)), or Gated Recurrent Units (GRU) networks (Cho et al. (2014)). In the present scenario, the sequences

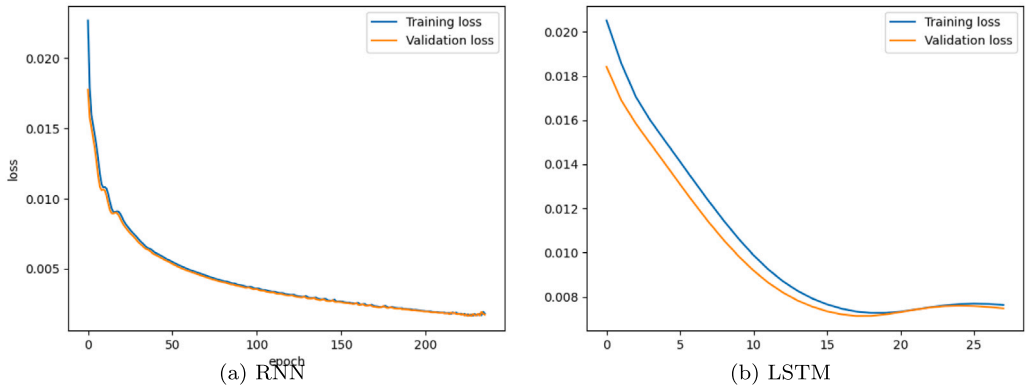


Fig. 10. History learning examples for (a) RNN and (b) LSTM, with 1 h time step and 3 time steps per sequence.

being short, the latter ones may prove less useful. The input features of the neural network are the activities of the sequence, as in real life, they are the observable variables.

An often used neural network tool for the prediction of an anomaly is autoencoder. An autoencoder is a neural network where the output equals the input, and they are often said to be unsupervised methods since there is no additional data to the input features to be used as the output labels. The autoencoder gives good results if the error, abs(output - input), is close to zero; one might see them as an identity operator. If fed with data corresponding only to 'normal' situations during the learning step, the error, when testing normal situation sequences, will be low. On the other hand, if fed with abnormal sequences, the error will be bigger, as the situation they encounter is not like the one it was trained for, and therefore does not belong to the same distribution. The problem is then to separate the two distributions, i.e., setting a threshold, which has been chosen in our case as the maximum absolute error value during training. The ideal case is when the two distributions do not overlap, otherwise possible confusions occur.

To sum up, we test either RNN/LSTM/GRU shaped autoencoders to build the anomaly detector. The interest of this kind of tool is that we only have to know what a normal situation looks like and do not bother about being precise about abnormal situation values: the training data set is composed only of normal activity sequences, and this is the reason why the sampling of the times is composed only of negative times as mentioned in Section 3.4. On the contrary, the test set is composed of about 50% of defects, and 50% of normal situations. The loss function was mean absolute error. There was no regularization parameter.

4.2. Prediction results

For all the training and test datasets, the results are 100% precise, whether the sequences are composed of 3 or 4 time steps (k = 3 or 4), whether the time step is one hour or 24 h, whether the type of autoencoder is made of RNN, GRU, or LSTM.

The difference between RNN and GRU or the LSTM is the learning speed, which is higher for the latter than for RNN, as depicted in Fig. 10. As an illustration, Fig. 11 shows the confusion matrix for the cases when time step is one hour and the sequences are 3 time steps long.

4.3. Sensitivity analysis of performances

The performance of the defect detector is good whatever the chosen type of anomaly detector, RNN, GRU or LSTM, due to the scale variation of activity when a defect occurs, and also to the absence of noise : the activity increase is linked to the defect size, and we want to know the behavior of the anomaly detector when the activity increase tends to small values. Additionally, the activity values that were generated were 'perfect'. So gaussian noise was added to the values of activity sequences, to simulate aleatory measurement errors. A scale factor was applied to calculation results after defect onset, whose value is 1 when no scale modification is applied. This factor is defined in Eq. (1).

Table 3
Input features of sensitivity analysis.

Parameter name	Values
sequence length	3 , 4 , 5, 6 , 10
time step (in hours)	1 , 2 , 3
scale factor	1 , 0.5 , 0.1 , 0.05 , 0.03
noise standard deviation	0, 0.001 , 0.01 , 0.02 , 0.05 , 0.1 , 0.2 , 0.3 , 0.5
auto encoder detector type	RNN , GRU , LSTM

$$scale = \frac{A'_1 - A_0}{A_1 - A_0} \tag{1}$$

where

A_0 is the computed activity before defect onset,

A_1 is the activity computed after defect onset,

A'_1 is the activity after scale modification.

The noise factor is a Gaussian noise applied to the activity levels during the sequence, defined in Eq. (2) as

$$A'' = A' \cdot N(1, \sigma) \tag{2}$$

where A'' is the activity of a given isotope at a given time after noise introduction,

A' is the activity of a given isotope at a given time before noise introduction (but after scaling),

σ is the standard deviation of noise.

In the end, a factorial design (FD) of scale, noise, sequence lengths, time steps, anomaly detector types(RNN,GRU,LSTM). The Table 3 describes the input features of the FD.

The chosen predictive performance metrics is F_1, whose definition is given in . The highest possible value of F_1 is 1, indicating perfect performance, and the lowest possible value is 0, indicating poor performance.

$$F_1 = \frac{2 \cdot TP}{2 \cdot TP + FP + FN} \tag{3}$$

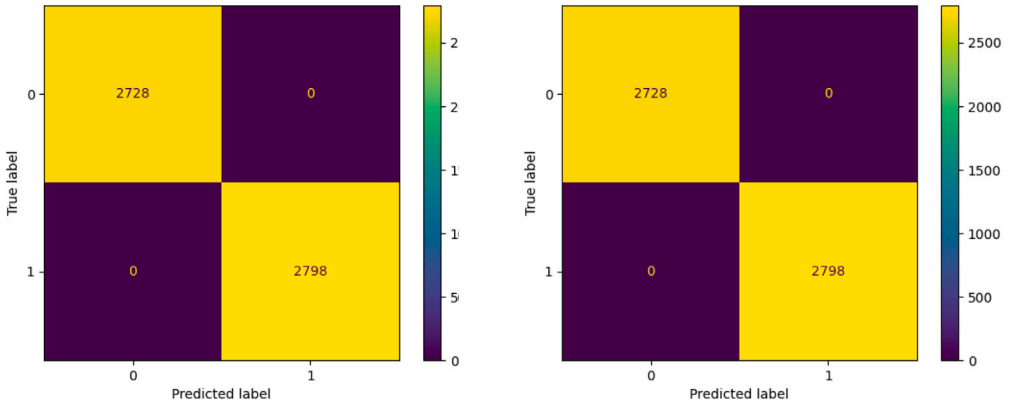
where

TP : True Positive of confusion matrix,

FP : False Positive of confusion matrix,

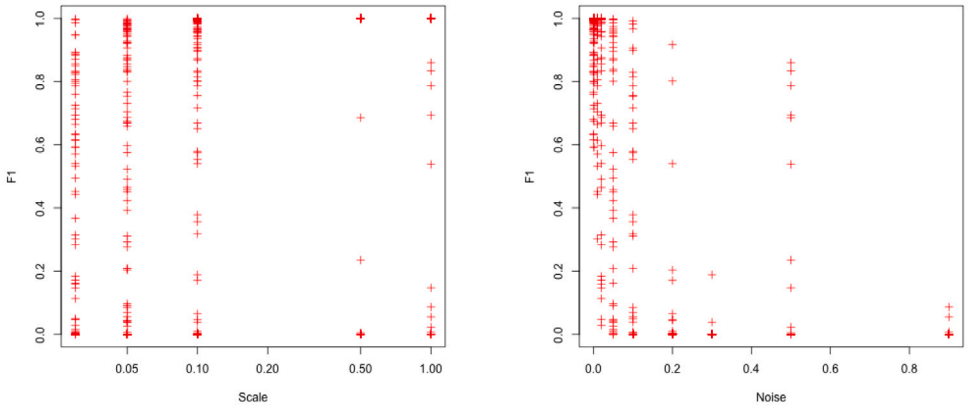
FN : False Negative of confusion matrix.

Figs. 12 show a scatter plot of F1 as function of scale and noise. As expected, the higher the noise, the lower F1. For scale, results are less graphically visible, and scale is not the best indicator. A more relevant metrics is the ratio between noise and scale, as shown at Fig. 13. Figure (a) shows a transition from low to good performance when the ratio roughly passes the value 1. Figure (b) is a focus on low



(a) RNN (b) LSTM

Fig. 11. Confusion matrix examples for (a) RNN and (b) LSTM, with 1 h time step and 3 time steps per sequence.



(a) Scale effect (b) Noise effect

Fig. 12. Scale and noise effect on F1.

values of the ratio. LSTM, RNN, and GRU types are highlighted. The horizontal black line on the top of the graph correspond to the 0.96 F1 value. In our study, RNN are the type of network that best supports lower scale to noise ratio. Fig. 14 gives the partial ranked correlation coefficients (PRCC) of F1 vs the sequence length, the scale to noise ratio, and the time step for both LSTM and RNN. PRCC were obtained using the sensitivity R package, largely based on Saltelli et al. (1999), activating bootstrap option for uncertainty range assessment of the PRCC. As expected, the scale to noise ratio is an important parameter. Then, PRCC for time steps are positive, but not very important, and consequently they might not be the first parameters to tune while building our anomaly detector. At last, the effect of sequence length is positive for LSTM, and negative for RNN, which is normal, because RNN are well known to be prone for gradient vanishing (and which is why LSTM were invented).

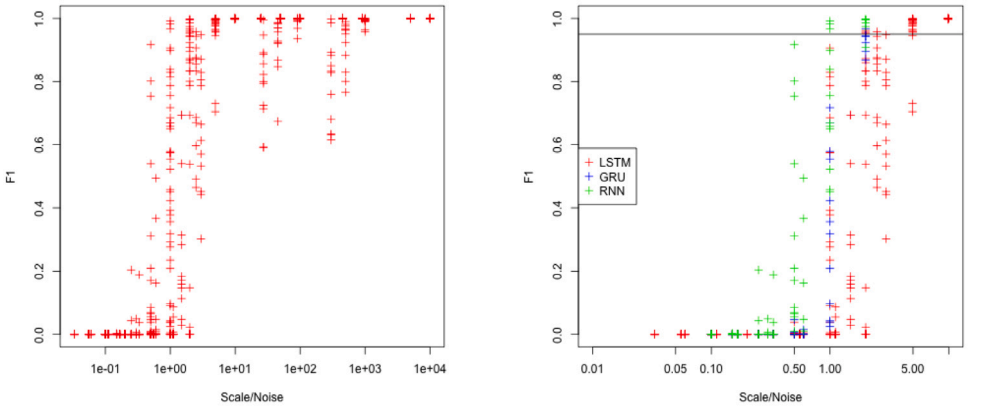
5. Conclusions

During normal reactor operation the activity of isotopes in the primary coolant is a good indicator for defect detection. A physical model for fission products release and primary activity calculation was developed. It was used to generate a 2000 computations database that trained an Artificial Neural Network for the purpose of building a surrogate model dedicated to activity predictions given fuel defect characteristics and primary coolant operating conditions, which then allows precise and fast calculations on demand on a simple laptop. A simple polynomial surrogate model of primary activity computation was also built for normal (no defect) situations.

These models were then used to build activity sequences, that were used to train recurrent neural networks autoencoders, the primary activity values being the features. These recurrent autoencoders gave 100% precise results with the genuine results of the surrogate models sequences, and a sensitivity study do noise and activity increases had to be conducted. The important metrics was the ratio between the activity increase and the noise. An influential parameter was the nature of the autoencoder : RNN gave good results when the sequence length was short, but were outperformed by LSTM when the sequence length increased, and the best anomaly detector might then be a combination of RNN autoencoder for short length sequences plus LSTM when longer sequences of data are available.

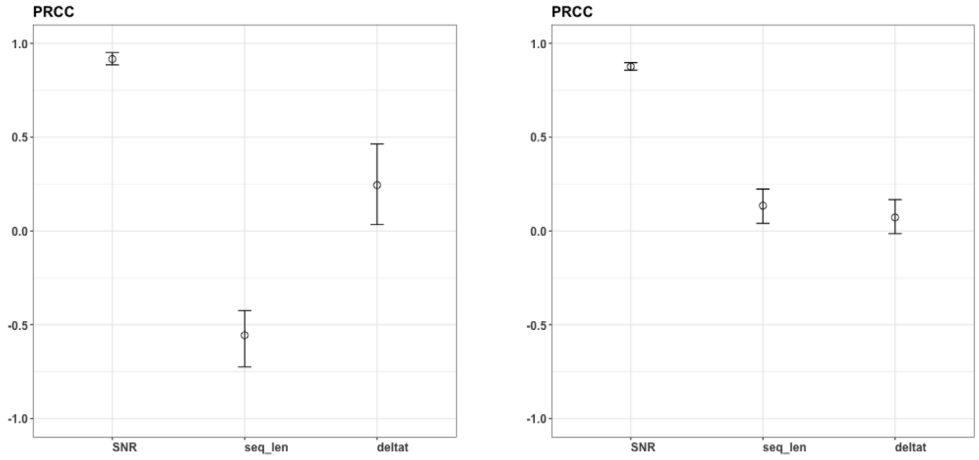
Some comments for improvements or limitations in the current work are:

1. The aim of this work was the development of a chain calculation in order to test the capabilities of machine learning for defect detection, and therefore the fact that the physical model was not tested against experiments was not a problem and was not our goal; but for real life use, this would be mandatory and probably result in further improvements to the physical model;
2. the fact that background primary activity level due to tramp uranium is lower than the activity level in case of defect made the defect detection easier, because the possible confusion situations were rare. In real life though, it is well known that sometimes small defects are not detected due to their activity level being lower than background primary activity;
3. other additional decay chains would probably add valuable information and improve the diagnostics;



(a) Effect of scale to noise ratio on F1 (b) Focus on small values of scale to noise ratios

Fig. 13. Effect of scale to noise ratio on F1.



(a) sensitivty of F1 for RNN (b) sensitivity of F1 for LSTM

Fig. 14. Ranked partial coefficient correlation of F1 vs timestep, sequence length, and scale to noise ratio.

4. the providing of additional information can help to improve the diagnostic in low scale to noise ratio area. The plant operating parameters like let down flow rate or BRS removal rate are generally well known and could be added as input features;
5. regarding defect characterization, the use of simple artificial networks gave poor results (undescribed in this article), due to slow convergence problems, but also to the too small amount of available isotopes; it was not possible to test other architectures during the course of this study, but convolutional neural networks or LSTM could be tested to that end.

Disclaimer

Views and opinions expressed in this paper reflect only the author's view and the European Commission is not responsible for any use that may be made of the information it contains.

CRediT authorship contribution statement

Karine Chevalier-Jabet: Writing – original draft, Validation, Supervision, Software, Resources, Methodology, Formal analysis, Data curation, Conceptualization. **Lokesh Verma:** Writing – review & editing, Visualization, Validation, Software, Methodology. **Francois Kremer:** Validation, Supervision, Methodology, Conceptualization.

Declaration of competing interest

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

Data availability

The authors do not have permission to share data.

Acknowledgments

The authors would like to thank the R2CA Project of the EU H2020 for financing the work carried out in this paper. This project has received funding from the Euratom Research and Training Programme 2014–2018 under Grant Agreement N. 847656. The authors also thank Jean Denis, Nathalie Girault and Didier Vola (IRSN) for the discussions and follow ups throughout the duration of the work.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X., 2015.

TensorFlow: Large-scale machine learning on heterogeneous systems. URL https://www.tensorflow.org/.

Abiodun, O.I., Jantan, A., Omolara, A.E., Dada, K.V., Mohamed, N.A., Arshad, H., 2018. State-of-the-art in artificial neural network applications: A survey. Heliyon 4 (11), e00938. http://dx.doi.org/10.1016/j.heliyon.2018.e00938.

Alfeo, A.L., Cimino, M.G., Manco, G., Ritacco, E., Vaglini, G., 2020. Using an autoencoder in the design of an anomaly detector for smart manufacturing. Pattern Recognit. Lett. 136, 272–278. http://dx.doi.org/10.1016/j.patrec.2020.06.008.

Chalapathy, R., Menon, A.K., Chawla, S., 2018. Anomaly detection using one-class neural networks. URL arXiv:1802.06360.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y., 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. URL arXiv:1406.1078.

Cortes C., V.V., 1995. Support-Vector Networks. Vol. 20, Kluwer Academic Publisher, pp. 273–297. http://dx.doi.org/10.1007/BF00994018.

Dong, B., Xiao, W., Yin, J., Wang, D., 2020. Detection of fuel failure in pressurized water reactor with artificial neural network. Ann. Nucl. Energy 140, 107104. http://dx.doi.org/10.1016/j.anucene.2019.107104.

Fiore, U., Palmieri, F., Castiglione, A., De Santis, A., 2013. Network anomaly detection with the restricted Boltzmann machine. Neurocomputing 122, 13–23. http://dx.doi.org/10.1016/j.neucom.2012.11.050, URL https://www.sciencedirect.com/science/article/pii/S0925231213005547.

Friedman, J.H., 2001. Greedy function approximation: a gradient boosting machine. Ann. Statist. 1189–1232.

Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y., 2014. Generative adversarial networks. URL arXiv:1406.2661.

Hinton, G.E., 2012. In: Montavon, G., Orr, G.B., Muller, K.-R. (Eds.), A Practical Guide to Training Restricted Boltzmann Machines BT - Neural Networks: Tricks of the Trade: Second Edition. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 599–619.

Hinton, G., Vinyals, O., Dean, J., 2015. Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531.

Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural Comput. 9 (8), 1735–1780. http://dx.doi.org/10.1162/neco.1997.9.8.1735.

IAEA, 2019. Review of Fuel Failures in Water Cooled Reactors (2006–2015). Nuclear Energy Series, (NF-T-2.5), IAEA, Vienna.

Iqbal, M.J., Mirza, N.M., Mirza, S.M., 2008. Stochastic simulation of fission product activity in primary coolant due to fuel rod failures in typical PWRs under power transients. J. Nucl. Mater. 372 (1), 132–140.

JEFF 3.3, 2017. The joint evaluated fission and fusion file. https://www.oecd-nea.org/dbdata/jeff/jeff33/.

Kieu, T., Yang, B., Guo, C., Jensen, C.S., 2019. Outlier detection for time series with recurrent autoencoder ensembles. IJCAI Int. Jt. Conf. Artif. Intell. 2019-Augus, 2725–2732. http://dx.doi.org/10.24963/ijcai.2019/378.

Lassmann, K., 1992. TRANSURANUS: a fuel rod analysis code ready for use. In: MATZ.K.E., H., SCHUMACHER, G. (Eds.), Nuclear Materials for Fission Reactors. In: European Materials Research Society Symposia Proceedings, Elsevier, Oxford, pp. 295–302. http://dx.doi.org/10.1016/B978-0-444-89571-4.50046-3.

Likhanskii, V., Afanasieva, E., Sorokin, A., Evdokimov, I., Kanukova, V., Khromov, A., 2006. Failed fuel diagnosis during WWER reactor operation using the RTOP-CA code.

Liu, F.T., Ting, K.M., Zhou, Z.-H., 2008. Isolation forest. In: 2008 Eighth IEEE International Conference on Data Mining. IEEE, pp. 413–422. http://dx.doi.org/10.1109/ICDM.2008.17, URL http://ieeexplore.ieee.org/document/4781136/.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E., 2011. Scikit-learn: Machine learning in Python. J. Mach. Learn. Res. 12, 2825–2830.

Pimentel, M.A., Clifton, D.A., Clifton, L., Tarassenko, L., 2014. A review of novelty detection. Signal Process. 99, 215–249. http://dx.doi.org/10.1016/j.sigpro.2013.12.026, https://linkinghub.elsevier.com/retrieve/pii/S016516841300515X.

R Core Team, 2022. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria, URL https://www.R-project.org/.

Ren, H., Xu, B., Wang, Y., Yi, C., Huang, C., Kou, X., Xing, T., Yang, M., Tong, J., Zhang, Q., 2019. Time-series anomaly detection service at microsoft. http://dx.doi.org/10.1145/3292500.3330680, arXiv:1906.03821.

Saltelli, A., Tarantola, S., Chan, K.P., 1999. A quantitative model-independent method for global sensitivity analysis of model output. Technometrics 41, 39–56. http://dx.doi.org/10.1080/00401706.1999.10485594.

Verma, L., Kremer, F., Chevalier-Jabet, K., 2023. Defective PWR fuel rods detection and characterization using an artificial neural network. Prog. Nucl. Energy 160, 104686. http://dx.doi.org/10.1016/j.pnucene.2023.104686.

Wallace, C., McEwan, C., West, G., Aylward, W., McArthur, S., 2020. Improved online localization of CANDU fuel defects using ancillary data sources and neural networks. Nucl. Technol. 206 (5), 697–705. http://dx.doi.org/10.1080/00295450.2019.1697174.

Yan, K., Chong, A., Mo, Y., 2020. Generative adversarial network for fault detection diagnosis of chillers. Build. Environ. 172, 106698. http://dx.doi.org/10.1016/j.buildenv.2020.106698, URL https://linkinghub.elsevier.com/retrieve/pii/S0360132320300561.