Nuclear Engineering and Technology 52 (2020) 2709–2716

Contents lists available at ScienceDirect

Nuclear Engineering and Technology

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

Original Article

Application of deep neural networks for high-dimensional large BWR core neutronics

Rabie Abu Saleem [a, *], Majdi I. Radaideh [b], Tomasz Kozlowski [b]

[a] Department of Nuclear Engineering, Jordan University of Science and Technology, P.O. Box 3030, Irbid, 22110, Jordan
[b] Department of Nuclear, Plasma, and Radiological Engineering, University of Illinois at Urbana Champaign, Urbana, IL, 61801, United States

ARTICLE INFO

Article history:
Received 17 February 2020
Received in revised form
12 April 2020
Accepted 11 May 2020
Available online 13 May 2020

Keywords:
Core optimization
Neutronic parameters
BWR
Deep neural networks
Half-symmetry
PARCS

ABSTRACT

Compositions of large nuclear cores (e.g. boiling water reactors) are highly heterogeneous in terms of fuel composition, control rod insertions and flow regimes. For this reason, they usually lack high order of symmetry (e.g. 1/4, 1/8) making it difficult to estimate their neutronic parameters for large spaces of possible loading patterns. A detailed hyperparameter optimization technique (a combination of manual and Gaussian process search) is used to train and optimize deep neural networks for the prediction of three neutronic parameters for the Ringhals-1 BWR unit: power peaking factors (PPF), control rod bank level, and cycle length. Simulation data is generated based on half-symmetry using PARCS core simulator by shuffling a total of 196 assemblies. The results demonstrate a promising performance by the deep networks as acceptable mean absolute error values are found for the global maximum PPF (~0.2) and for the radially and axially averaged PPF (~0.05). The mean difference between targets and predictions for the control rod level is about 5% insertion depth. Lastly, cycle length labels are predicted with 82% accuracy. The results also demonstrate that 10,000 samples are adequate to capture about 80% of the high-dimensional space, with minor improvements found for larger number of samples. The promising findings of this work prove the ability of deep neural networks to resolve high dimensionality issues of large cores in the nuclear area.

1. Introduction

In-Core Fuel Management (ICFM) is an essential part of the operation of all commercial nuclear reactors. At the beginning of each fuel cycle (BOC), an optimal loading pattern is required to yield safe and economic operating conditions such as uniform power distribution and high fuel burnup. These operating conditions are sought for maximum operational life with minimum cost and highest levels of safety, consequently, leading to profit-maximization.

In principle, the achievement of an optimal loading pattern is secured through neutronic analysis using Monte Carlo codes such as MCNP5, Serpent, and OpenMC, or using deterministic core simulators such as PARCS. There are several parameters that determine the optimal loading pattern including fuel enrichment, number of fuel assemblies, and the way these assemblies are loaded into the reactor core. Other non-fuel related parameters include the concentration of burnable poison and the locations of fixed assemblies inside the core (e.g. rodded assemblies).

As it appears, solving the nonlinear ICFM problem is a computationally demanding process that poses a serious challenge to reactor operators and hinders their efforts. Several options have been suggested to tackle the problem of solving the neutronic behaviour of a nuclear reactor for a huge space of possible loading patterns with reasonable time and computational power requirements. One of these options entailed replacing computationally expensive codes (PARCS, MCNP5, etc.) by surrogate models that are capable of performing the calculations with very little computational power and reasonably high accuracy.

Previously, Artificial Neural Networks (ANNs) have been explored as an alternative to predict nuclear reactor physics parameters within a reasonable time and computational power requirements. Mazrou and Hamadouche developed two feed-forward neural networks to predict the multiplication factor (k_{eff}) and the Power Peaking Factor (PPF) for the IAEA Low Enriched Uranium (LEU) research reactor benchmark at the beginning of its cycle [1,2]. In their work, the backpropagation algorithm with

* Corresponding author.
E-mail address: raabusaleem@just.edu.jo (R. Abu Saleem).

https://doi.org/10.1016/j.net.2020.05.010

adaptive learning rate was used for training based on simulation data derived from the neutronic diffusion code MUDICO-2D. They demonstrated that ANNs are capable of emulating sophisticated computer codes with acceptable error margins.

Hedayat et al. developed four cascade feed-forward ANNs to predict four core parameters for the same IAEA benchmark [3]. They used simulation data from the CITVAP v3.2 diffusion code to train their ANNs to estimate the multiplication factor with all control rods out ($k_{eff\text{-}out}$), the multiplication factor with control rods fully inserted ($k_{eff\text{-}in}$), the maximum thermal flux density (φ_{max}), and the Power Peaking Factor (PPF). Their results showed that ANNs can be trained to estimate core parameters quickly and effectively. They also showed that the prediction process depends on several parameters such as the type and the structure of the network as well as the functions used for learning and activation.

Reactor parameters related to other physics inside the core were also predicted using ANNs. Mirvakili, Faghihi and Khalafi, designed one ANN to accurately predict five thermal-hydraulics parameters for a typical VVER-1000 [4]. Training data for the ANN was secured by performing calculations with the sub-channel thermal-hydraulics analysis code COBRA-EN. Another study used the thermal-hydraulics code RELAP5 to perform simulations for a Natural Circulating Boiling Water Reactor (NCBWR) with different design parameters and conditions. The study was aimed at minimizing the volume of material used in the construction of a NCBWR. The ANN in that study was able to predict mass flow rate and void fraction values with reasonable accuracy [5]. In a more recent study, a Radial Basis Function Neural Network (RBFNN) was developed to predict the probability of Pellet Clad Interaction (PCI) failure of fuel rods based on experimental data collected from the Douglas pressurized heavy water reactor in Canada [6]. The network's purpose of deciding whether a fuel rod would exhibit a defect during a given transient or not was achieved with moderate accuracy and faster performance compared to other mechanistic models.

Deep learning is a branch of artificial intelligence that adopts a hierarchical level of ANNs to carry out the process of machine learning. In the context of Deep Neural Networks (DNN), different applications were conducted to analyse nuclear power plants. Group Method of Data Handling (GMDH) networks, which are a special form of ANN/DNN trained by polynomial regression, were used to analyse nuclear reactor simulations and their parametric uncertainty [7]. Bayesian deep learning through deep Gaussian processes was utilized to construct surrogate models to analyse advanced nuclear reactor simulations with different input space dimensionalities, including reactor physics, thermal-hydraulics, and fuel performance [8]. The surrogates achieved very good accuracy on all simulations, and afterward they were used to perform advanced uncertainty quantification tasks in an efficient form (e.g. uncertainty propagation, variance decomposition, parameter screening). Moreover, DNNs were utilized by several authors to study various nuclear severe accidents. Yang and Kim adopted recurrent neural networks through long short-term memory to diagnose nuclear accidents such as loss of coolant accident (LOCA) and a steam generator tube rupture [9]. Radaideh and Kozlowski [10] integrated deep learning as a part of a new framework for advanced energy modeling. Deep learning is demonstrated through a time series regression problem to predict the water level during LOCA using DNN with large amount of simulation data; and a high accuracy was achieved. Additional study on using DNN to predict core water level during LOCA was performed by Do Koo et al. [11]. The authors found good performance by DNN compared to fuzzy neural networks.

It is worth mentioning that most of the research employing ANN/DNNs in the prediction of reactor core parameters targeted Pressurized Water Reactor (PWR) systems due to the advantages of core symmetry (e.g. 1/8) at beginning of cycle and relatively small core size. Nevertheless, few studies investigated the effectiveness of ANNs in predicting core parameters of the more complicated Boiling Water Reactor (BWR) systems [12–14]. Because of the high-dimensionality nature of BWR cores, these studies were based on symmetry assumptions (quarter-symmetry at most) aimed to reduce the input space of their networks and simplify the training process. *This study has two major goals to accomplish: The first goal is to predict the core parameters in large BWR cores, which have issues associated with symmetry and very large number of fuel assemblies to be moved. The second goal is to introduce the deep learning methodology as an option to resolve nuclear high dimensional problems.* In this study, three DNNs are developed to predict reactor physics parameters of the Ringhals-1 BWR based on simulation data derived from the Purdue Advanced Reactor Core Simulator (PARCS) code. In other words, the deep learning performance in predicting neutronic parameters in high-dimensional spaces is investigated in this work.

2. Methodology

2.1. Problem description

The data used in this study is based on the well-known OECD Ringhals stability benchmark [15]. The benchmark contains a comprehensive and well-defined set of geometry and core composition data. It also contains a complete time series data from measurements at the beginning of cycles 14, 15, 16, and 17. The Ringhals-1 unit, which started operation in 1976, is a Swedish BWR designed by ABB-ATOM with a nominal thermal power of 2270 MW. The reactor core consists of 648 fuel assemblies of three different types: 8×8 assemblies with 63 fuel rods, SVEA assemblies with 63 fuel rods, and SVEA assemblies with 64 fuel rods.

Simulations are performed using PARCS core simulator. PARCS is a multi-dimensional reactor kinetics code that can be easily coupled to the thermal-hydraulics code TRACE. It solves the time-dependent two-group neutron diffusion equation in three-dimensional Cartesian geometry using nodal methods [16].

The process of this study entails using a validated PARCS model to simulate core neutronics during cycle 14 based on the data provided in the benchmark specifications. For this model, the core is depleted until criticality is not maintained. At the end of cycle 14, burnup data and Xenon concentrations are extracted from output history files to be used for the development of several PARCS models for the next cycle (cycle 15). Each model in cycle 15 features the removal of one third of the burned fuel (assemblies with highest burnup) and replacing them with fresh fuel assemblies. Moreover, each model comprises 14 different types of Control Rod Banks (CRBs) used to maintain criticality of the core during the depletion process. This is achieved by using the “criticality search” feature of PARCS. For these models, each depletion step represents core operation at 80% power for 35 days.

For all PARCS models, core nodalization is based on one node per assembly in the radial plane (resulting in 32 by 32 planar nodes) and 27 axial nodes. Both radial and axial reflectors are included in the model nodalization. The PARCS neutronics solution for steady state is based on the two-group nodal diffusion option.

To reduce the running time and the computational power required, a 324-assembly model with half-core symmetry is used. This is the only symmetry option available for this large BWR core. It is worth mentioning that 1/2 core symmetry assumption is valid in fresh cores only, as symmetry is lost after fuel burnup. Our symmetry assumption aims to construct a reasonable input space that can be handled by the DNN proposed in this work. Reliance on the symmetry assumption was demonstrated and accepted in many

previous studies [12,17,18]. A schematic of the core corresponding to cycle 15 models is depicted in Fig. 1 and a schematic of the CRB arrangement inside the core is shown in Fig. 2.

2.2. Data generation and processing

Upon the completion of cycle 14, several PARCS models for cycle 15 were developed based on the following general heuristics (see Fig. 1):

a One third of the most burned fuel assemblies are replaced with fresh fuel assemblies.
b No fresh fuel assemblies are loaded to the periphery of the core.
c Fresh fuel assemblies are arranged in a checkerboard pattern that is intended to achieve PPF values as small as possible.
d Burned fuel assemblies are shuffled randomly inside the possible 196 positions remaining after the fresh fuel is loaded.

Theoretically, such arrangement corresponds to a number of possible loading patterns equals to the factorial of 196. In an attempt to capture core behaviour based on this huge space, a total of 17,900 PARCS models for cycle 15 are developed to provide the data necessary for training and testing the neural network.

The loading patterns corresponding to different PARCS models are used as inputs to three neural networks. For each case, the input is a vector of length 196 corresponding to all assembly positions available for loading the burned fuel assemblies. Each entry of the input vector is assigned a number corresponding to the old index of the burned assembly (i.e. the index of the fuel assembly inside the core during cycle 14). To ensure that DNNs interpret these indices as labels (categorical/discrete data), not as numerical values, the input data is converted to a binary matrix through a process known as one-hot encoding. In this process, the categorical data is converted to integer data, such that the numerical value "1" appears in each row at the entry that matches the assembly location, and the numerical value "0" appears elsewhere in that row. For example, consider the vector {3, 1, 6}, which represents labels of three assembly locations, the input vector can be converted into a sparse matrix with one-hot encoding as:

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

This process is applied to all 196 locations in the input data and repeated to all 17,900 samples available from PARCS.

In terms of DNN modelling, two options are available: fitting one complex DNN to predict the three responses: PPF, CRB, CL, or fitting three separate DNNs for each output type. Since the three outputs have different scales and types, we expect the DNN to have a complex and very deep structure, which requires many layers and nodes to train. More importantly, hyperparameter tuning will also be difficult due to the large number of hyperparameters and slow training to evaluate each hyperparameter configuration. Driven by these reasons, we proceeded with the second approach by using three DNNs that are simpler in structure, easier to train and tune, while retaining same accuracy.

The first model (designated as DNN1) predicts the three PPF values: The global maximum (P_{xyz}), the axially-averaged maximum in radial direction (P_{xy}), and the radially-averaged maximum in axial direction (P_z). PPF is an important safety constraint to meet in neutronic calculations.

The second model (designated as DNN2) predicts the CRB level required to maintain criticality at the beginning of cycle 15. CRB level is crucial for BWR control which involves more control rod movements than PWR. To use the "criticality search" feature of PARCS, the twelve CRBs are grouped into seven sets, each set moves independently during the process of criticality search. Among all sets, only one set (the set composed of CRB12 and CRB13) features a variation in position for different simulation cases. Positions of other CRB sets are unchanged for all simulation cases (either fully inserted or fully withdrawn). Hence, DNN2 features one output only representing the position of that set, and "CRB" is used to refer to this output for simplicity.

Finally, the third model (designated as DNN3) predicts the Cycle Length (CL) for cycle 15. Unlike previous studies that predict k_{eff}, we used CL (or burnup) in a form of discrete depletion steps, which are in unit of days. This is because of predicting CRB level at beginning of cycle, which would affect initial core criticality. Also, we used $k_{eff} = 1$ as an indication of the end of cycle. Therefore, all cores will have very different initial k_{eff} at beginning of cycle due to variable control rod insertion and will cross $k_{eff} = 1.0000$ at different times, so CL (or burnup) is more representative of core quality. Unlike the previous outputs (PPF, CRB) which are continuous, CL is a categorical/discrete output, and it is expressed in number of depletion steps (e.g. 8, 9, 10, etc.). For example, a value of 9 for CL means that 9 depletion steps are required to maintain criticality for this loading pattern. Similar to the input data, the CL outputs are processed with one-hot encoding. For this study, generation of random loading patterns, development of PARCS models, and extraction of output data are all programmed using MATLAB.

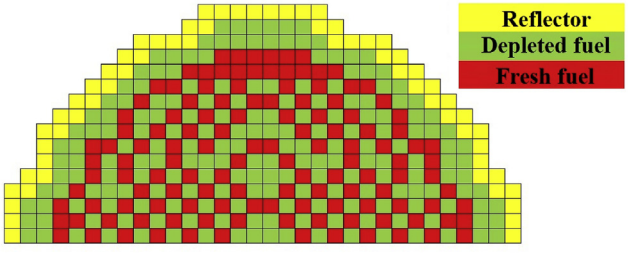

Fig. 1. Generic schematic of the core corresponding to cycle 15 in Ringhals-1.

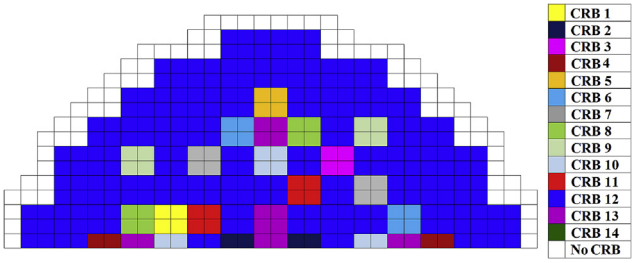

Fig. 2. Control rod bank arrangement inside Ringhals-1 core.

2.3. Deep neural networks

Feedforward neural networks in the form of fully-connected layers are considered as the fundamental deep learning models [19,20]. Deep neural networks (DNN) are used when the relationship between the input and the output is mapped by multiple hidden layers [21]. DNNs in concept aim to approximate any general function $f(\boldsymbol{x})$ using a mapping $f^*(\boldsymbol{x}, \boldsymbol{\theta})$, by adjusting $\boldsymbol{\theta}$ values (model parameters) such that [22]:

$$f(x) \approx f^*(x, \theta). \quad (1)$$

In this work, $f(\boldsymbol{x})$ is the PARCS core simulator, which can be treated as a black-box code, where the user can access only input/output interface.

DNNs consist of several hidden layers connected through nodes, where each hidden layer is used to capture part of the overall relationship. The output of the first hidden layer can be calculated by applying an activation function over the multiplication of the initial weights and the input layer ($\boldsymbol{x}$), i.e. $\boldsymbol{h}_1 = g_1(\boldsymbol{W}_1\boldsymbol{x}+\boldsymbol{b}_1)$. After the first layer, each hidden layer receives input from the preceding layer ($\boldsymbol{h}_{i-1}$), and produces output to the next layer ($\boldsymbol{h}_i$) as follows:

$$\boldsymbol{h}_i = g_i(\boldsymbol{W}_i\boldsymbol{h}_{i-1} + \boldsymbol{b}_i), \tag{2}$$

where i is the index of the current hidden layer, $g(.)$ is the activation function (e.g. sigmoid, ReLU, etc.), $\boldsymbol{W}$ is the weight matrix, and $\boldsymbol{b}$ is the bias/intercept term. After completing the forward propagation step in all layers, the predicted output ($\hat{y}$) and the target output (y) can be used to determine the cost function such as Mean Absolute Error (MAE) for regression problems (e.g. PPF, CRB):

$$\mathrm{MAE} = \frac{1}{\mathrm{n}}\sum_{\mathrm{i}=1}^{\mathrm{n}}[\hat{y}_i - \mathrm{y_i}], \tag{3}$$

where y is the target output and n is the number of samples in the dataset. It is worth mentioning that the authors performed detailed sensitivity tests and concluded that training on MAE or mean squared error (MSE) has similar performance, with MAE provided a slightly better performance. Therefore, we decided to continue using MAE as the main training metric in this work for PPF and CRB responses. For classification problems (e.g. predicting CL), the accuracy metric is defined as:

$$Accuracy = \frac{Number\ of\ Correct\ Labels\ Predicted}{Total\ Number\ of\ Labels\ Predicted} \tag{4}$$

which refers to the percentage of correct labels predicted by the network. For each training step, the errors in Eq. (3) are propagated backwards through gradient and are used to update the weights of the hidden layers so that the predicted output becomes closer to the target output. The current weights are adjusted by the multiplication of the error gradients and the coefficient (α), which is called the Learning Rate (LR). The backpropagation step can be trained by Stochastic Gradient Descent (SGD) [23] or adaptive moment estimation (Adam) [24]. Lastly, it is worth mentioning that DNN input data will change from one fuel cycle to the other, since the output of a fuel cycle is used as an input to the next cycle. This implies that DNN training and validation processes should be performed to each cycle individually.

Table 1
Optimum hyperparameters for the three DNN models.

Item	DNN1	DNN2	DNN3
Hidden Layers	5	3	4
Nodes per layer	66, 59, 51, 46, 41	67, 50, 45	50, 45, 45, 40
Learning rate (initial)	3.00E-03	3.00E-03	2.00E-03
Dropout (after first layer)	0.3	0.3	0.5
Metric	MAE	MAE	Accuracy
Batch Size	64	64	64
Epochs	100	100	100
Activation	ReLU	ReLU	ReLU
Optimizer	Adam	Adam	Adam
Training samples	15,000	15,000	15,000
Validation samples	2000	2000	2000
Test samples	900	900	900

3. Results and discussion

The results of this paper are presented in three subsections. First, the procedures followed to optimize the DNN architecture as well as the optimum hyperparameters are described. Second, training and validation results of all DNN models are presented. Lastly, validated DNN models are used to predict the neutronic parameters for selected test cases, which are unseen by the DNN models. It is worth mentioning that all analysis and results presented in this section are programmed using Python under the Keras deep learning package with TensorFlow backend [25].

3.1. Network optimization

As mentioned before, due to the nature and scale of the outputs predicted, three DNNs are constructed where each network predicts specific output(s). The first model DNN1 predicts the three PPFs, namely, P_{xyz}, P_{xy}, and P_z. The second model DNN2 predicts the CRB level, while the third model DNN3 predicts the CL labels. The three PPF parameters as well as the CRB outputs are continuous variables, while CL is a discrete output. The inputs to the three DNN models are labels referring to the assembly location in the core, which are 196 locations in total for this study.

For hyperparameter tuning, the reader should notice that our objective is not to find the global optimum of hyperparameters, but

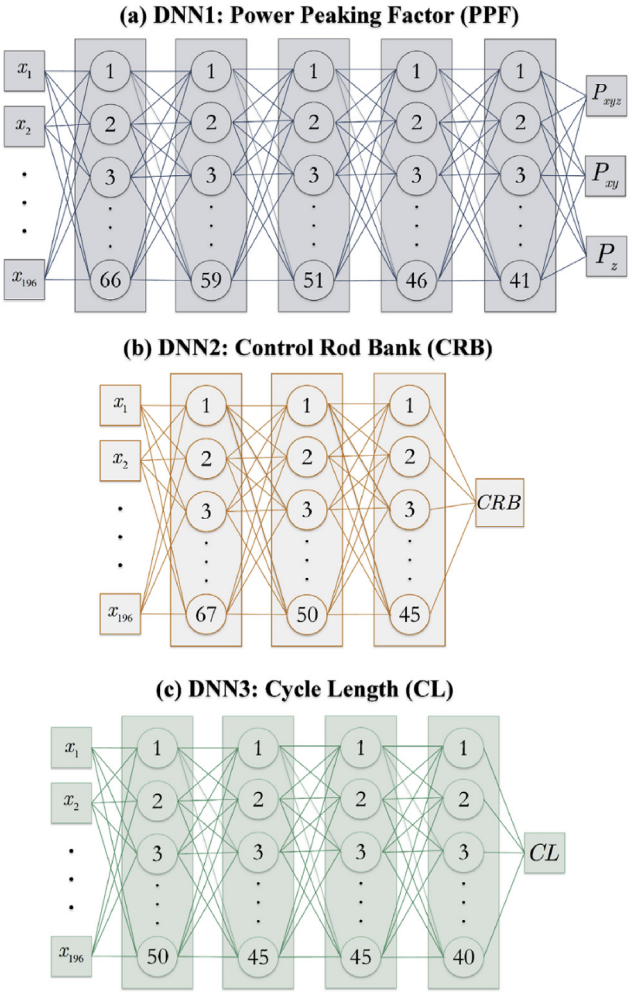

Fig. 3. DNN optimum architecture for (a) DNN1: PPF, (b) DNN2: CRB, and (c) DNN3: CL.

rather a combination that provides satisfactory performance. To facilitate the search, we assumed the number of nodes is independent from the rest of hyperparameters such that the following two-step process can be applied:

1 Grid search: Majority of DNN hyperparameters (number of layers, initial LR, activation, batch size etc.) are optimized using grid search, where different combinations of these hyperparameters are tested to determine the best configuration. This step is applied to most hyperparameters with few discrete possibilities. For example, for activation function (ReLU, Sigmoid), loss metric (MAE, MSE), batch size (16, 32, 64, 128), and number of layers (1, 2, …, 10), finite set of combinations can be generated and evaluated to determine the best set.
2 Gaussian process (GP) search: This option is used to determine the number of nodes in each hidden layer since number of nodes has a large range. GP search incorporates prior knowledge using the information obtained from the evaluated configurations, which helps in guiding the search and reducing the number of trials to determine the optimal number of nodes per layer. For the previous example, based on step 1 results (e.g. 5 layers, 32 batch size, MAE, and ReLU activation), the nodes in these 5 layers are optimized using the GP optimizer.

After applying the previous criteria, the optimum hyperparameters of the three DNN models are determined as listed in Table 1. For example, for DNN1, five hidden layers with 66, 59, 51, 46, and 41 nodes, respectively, yield the best performance for this network. Most of the hyperparameters (batch size, epochs, activation, optimizer, training/validation/test split) seem to be similar across the three networks. A graphical sketch of the optimum architecture for the three DNN models is shown in Fig. 3. A dropout layer is used only after the first hidden layer of each model with dropout value listed in Table 1. Dropout layers are used to decrease network overfitting. Since DNN3 works with discrete/categorical output, the accuracy metric is used instead of MAE.

The LR is adjusted during training when the network improvement reaches a plateau. After 5 epochs with no improvement on the validation MAE/accuracy, the LR is decreased by a factor of 0.8 to enhance the gradient search. Based on step 1 optimization, activation function used for training is the Rectified Linear Unit (ReLU) function and Adam is used for optimization in all three networks.

3.2. Training/validation

After finishing the hyperparameter tuning process as given in Fig. 3 and Table 1, we can start training and validating the models. Fig. 4(a), (b), (c) show the change in the loss/metric function (i.e. MAE for PPF and CRB and accuracy for CL), with number of training epochs for the three DNN models. Since LR is adjusted every 5 epochs of no improvement on the validation metrics, Fig. 4(d), (e), (f) show the change in LR with number of training epochs for the three DNN models. Both training and validation losses decrease as desired with increasing number of epochs for DNN1 and DNN2, and training and validation accuracy increase with increasing number of epochs for DNN3. We can notice that all networks show similar behaviour as they tend to converge to the best possible result after about 20 epochs. Afterward, the network shows slight to no improvement until the end of training. This can be confirmed as well from the similarity in the LR schedule in Fig. 4(d), (e), (f), which show that the three networks experience same LR values during their training, and majority of these LR adjustments could not provide any improvement, especially after 30 epochs (i.e. LR is dropped constantly by 20% after every 5 epochs). At higher epochs, the network starts to show slight overfitting, especially for DNN2 and DNN3, meaning that the network becomes overly trained at this point. For example, in Fig. 4(c), training accuracy of DNN3 starts to become larger than the validation accuracy after about 20 epochs. This implies that the models after this range are more overfitted to the training data, and cannot be useful to extrapolate to new data. To resolve this issue, *the best model at the best epoch,*

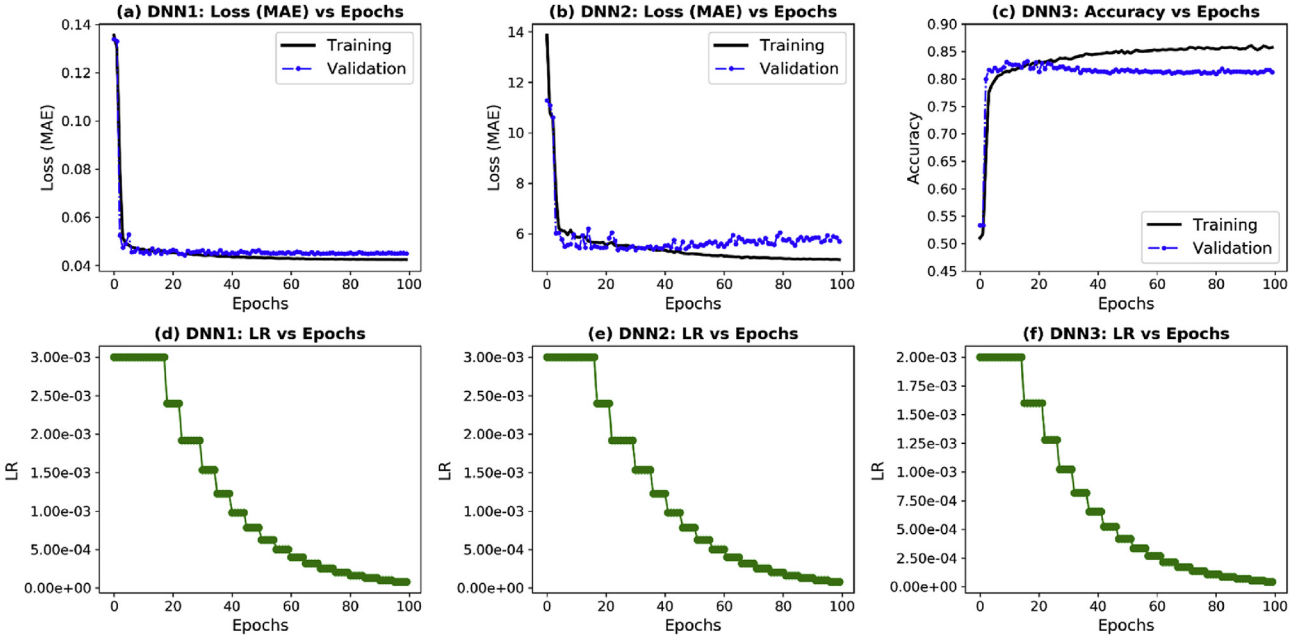

Fig 4. (a), (b), (c) The convergence of network training/validation loss (MAE) and accuracy metrics and (d), (e), (f) the variation of learning rate with number of epochs for the three DNN models.

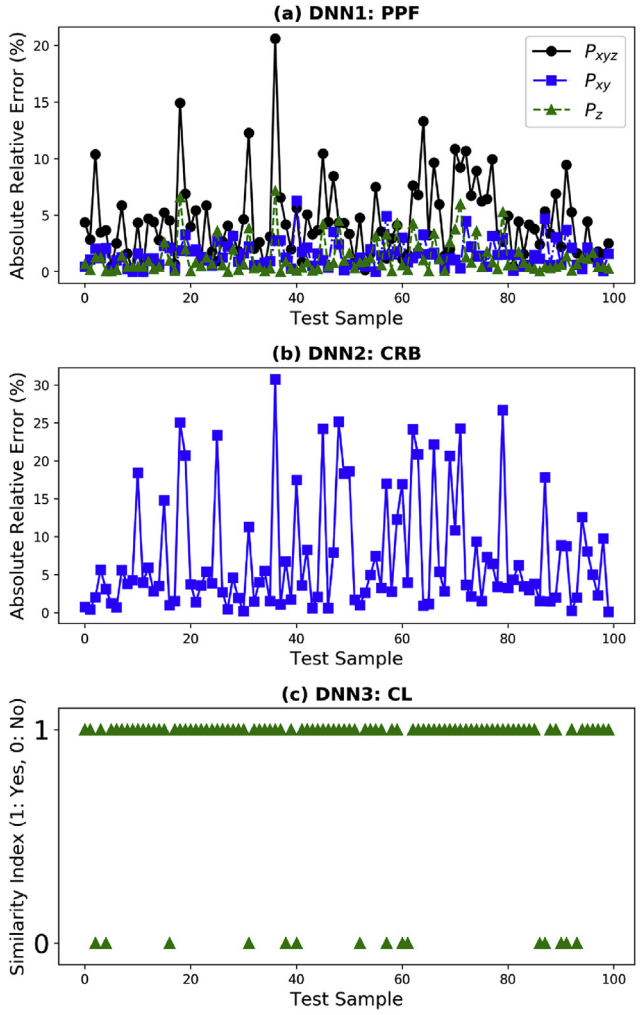

Fig. 5. Model testing for 100 test samples: (a) DNN1 absolute relative error (|Target − Prediction|/Target × 100%), (b) DNN2 absolute relative error, and (c) DNN3 (similarity index with **1** when Target = Prediction and **0** when Target ≠ Prediction).

which has highest validation accuracy, is saved and used for further analysis in the paper.

3.3. Testing results

After confirming the validity of the trained models, we can utilize them to predict the test cases, which were not used neither in model training nor validation. Fig. 5(a)-(b) show the absolute relative error (|Target-Prediction|/Target × 100%) for 100 test cases for the PPF and CRB level outputs, while Fig. 5(c) illustrates the similarity index in binary format to predict the CL labels in the test set. The value of **1** occurs when the target and prediction match, and **0** when they mismatch.

Table 3
MAE/accuracy metrics over the test set (900 samples) for all outputs.

Output	Metric Value
P_{xyz} (MAE) [−]	0.195
P_{xy} (MAE) [−]	0.059
P_z (MAE) [−]	0.053
CRB (MAE) [%]	5.6
CL (Accuracy) [%]	81.8

The results show that P_{xyz} have larger prediction error than both P_{xy} and P_z with relative error can reach to 20%. Both P_{xy} and P_z have similar trends with absolute errors of order of 6% or less. For CRB level, the relative error captured by DNN2 seems to be higher, implying that the prediction of CRB level for this large core is more difficult. Majority of the CRB absolute relative errors are within 5%, but they can reach 30% for certain core patterns. Lastly, prediction of the CL is satisfactory, as most of the CL labels are predicted correctly by DNN3. Out of the 100 test cases shown in Fig. 5(c), 14 cases are mis-predicted, meaning that 86% accuracy can be achieved by DNN3 for this case.

Quantitatively, target and predicted values are compared in Table 2 for 4 test cases selected randomly from the test set. The summary of MAE (see Eq. (3)) for PPF and CRB and accuracy (see Eq. (4)) for CL over all test samples (i.e. 900 in total) is listed in Table 3. The results show that P_{xyz} has MAE of ~0.2, which is about 4 times larger than P_{xy} and P_z. CRB has MAE of about 5%, while the accuracy of CL test predictions is about 82%.

Moreover, it is worth analyzing the 18% wrong predictions of DNN3. More specifically, determining the fraction of labels that are mis-predicted by 1 burnup step, 2 steps, etc. This information is given in Table 4, which shows how many test samples out of 900 are predicted correctly, mismatched by 1 step, 2 steps, or 3 steps. The results clearly show that most of the mismatched labels (about 17.6%) are off by 1 step, only 4 labels are mis-predicted by two steps (<0.5%), and no labels are mis-predicted by three steps.

Finally, it is important to analyse the effect of number of training samples on the network accuracy to confirm whether enough samples are used in this study or not. It is known that the full space for this problem is the factorial of 196 (which ensures that each assembly is in every location of the 196 possible locations). This number is astronomical and cannot be generated practically in PARCS. Fig. 6 shows the convergence of DNN3 accuracy on both training and validation sets with total number of samples. Notice that we considered three different cases for the validation split. The

Table 4
Classification of the test predictions of the CL labels by DNN3.

Item	Num. of Test Samples
Match	737
Mismatch by 1 step	159
Mismatch by 2 steps	4
Mismatch by 3 steps	0
Sum	**900**

Table 2
Comparison of target and predicted values of all outputs for 4 random test cases.

Output	Test Case 1		Test Case 2		Test Case 3		Test Case 4	
	Target	Predict	Target	Predict	Target	Predict	Target	Predict
P_{xyz}	4.314	4.256	3.787	3.421	4.849	4.681	4.879	5.002
P_{xy}	2.082	2.128	1.887	1.840	2.253	2.253	2.344	2.327
P_z	2.049	1.984	1.980	1.901	2.125	2.078	2.098	2.088
CRB (% insertion)	59	60	68	67	83	76	50	46
CL (depletion steps)	9	9	8	8	9	8	9	9

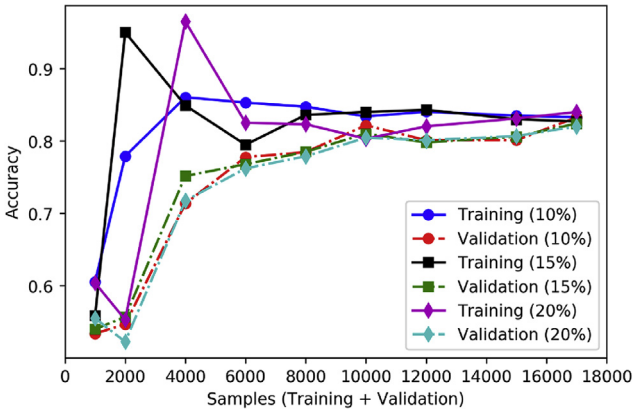

Fig. 6. Convergence of training and validation accuracy of the DNN3 with number of samples for different validation split (fraction of validation split from the total is expressed between parentheses in the legend).

first case is **10%** split (e.g. 600 validation samples and 5400 training samples are used for the 6000 test). The second and third cases have **15%** and **20%** validation split, respectively. The results clearly show that the validation accuracy improves significantly when moving from 1000 to 10,000 samples. Afterward for larger samples, the accuracy experiences minor improvement. Therefore, we can conclude that 10,000 samples are adequate to achieve about 80% accuracy for this extremely high-dimensional problem, which is promising. In addition, we can notice that for small sample size (1000–4000), the training accuracy can be high (>0.9), but the validation accuracy is much lower, which is a clear model overfitting due to limited data availability in these sample sets. Therefore, the convergence of the validation accuracy is more relevant to ensure that the model is not overfitted to the training data, which occurs at about 10,000 sample size or more.

Fig. 6 also shows that the validation split has insignificant effect on the final converged accuracy as all models tend to converge to about 80% accuracy. It is worth mentioning that the small inconsistencies in the training/validation accuracy trends between 10,000 and 17000 samples may be attributed to using the optimum hyperparameters listed in Table 1 for all cases. Notice that these hyperparameters are optimized based on 17,000 samples, and so minor adjustments are needed for other sample sets such that the continuous improvement with more data can be seen. Nevertheless, the accuracy differences in the range 10,000–17000 samples are still small and can be of no significance compared to how much the accuracy has changed in the range 1000–10000 samples.

The practical implications of this work include using the built DNN models for uncertainty quantification, sensitivity analysis, and/or model calibration, which require hundreds of thousands of model execution. DNN models can significantly accelerate these tasks. However, the analyst needs to propagate the DNN model uncertainty into the final quantified uncertainty to obtain realistic confidence bounds. Another application of this work would include combining the DNN models in core optimization, where DNN can be used with reinforcement learning or genetic algorithm to accelerate the optimization search. The optimizer along with DNN can be used to filter hundreds of core patterns that satisfy certain constraints. Afterward, these patterns can be evaluated with the real neutronic solver to avoid any discrepancy from the DNN model.

4. Conclusions

The problem of simulating the reactor physics of large cores such as BWRs is characterized by high-dimensional spaces due to the lack of high-order symmetry (e.g. 1/4, 1/8), making it a difficult and computationally demanding problem. Deep learning through well-optimized deep feed-forward neural networks (using detailed hyperparameter optimization) was utilized in this study to predict core neutronic parameters including power peaking factor, control rod bank level, and cycle length of Ringhals-1 BWR unit. The shuffling process includes 196 assemblies, which imposes additional difficulty on the optimization process. The results found were promising and they demonstrate that the mean absolute error in the test set of the global maximum PPF is about 0.2, while 0.05 is found for the radially and axially averaged PPF. The neural network prediction performance for control rod level is off by 5% insertion depth, while cycle length labels are predicted with 82% accuracy. The analysis also demonstrated that 10,000 samples are adequate to capture about 80% of this extremely high-dimensional problem, with minor improvements are expected for larger number of samples. In summary, this work proves the ability of deep neural networks to resolve high dimensionality issues in the nuclear area, with possibility for extension to other problems in future such as nuclear data uncertainty characterization and prediction of core parameters in multiphysics coupled simulations. Ongoing work includes combining deep learning models developed in this work with reinforcement learning algorithms to perform core optimisation.

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.

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.net.2020.05.010.

References

[1] H. Mazrou, M. Hamadouche, Application of artificial neural network for safety core parameters prediction in LWRs, Prog. Nucl. Energy 44 (2004) 263–275.
[2] International Atomic Energy Agence, Research Reactor Core Conversion from the Use of Highly Enriched Uranium to the Use of Low Enriched Uranium Fuels Guidebook, vol. 233, IAEA-TECDOC, 1980.
[3] A. Hedayat, H. Davilu, A.A. Barfrosh, K. Sepanloo, Estimation of research reactor core parameters using cascade feed forward artificial neural networks, Prog. Nucl. Energy 51 (2009) 709–718.
[4] S.M. Mirvakili, F. Faghihi, H. Khalafi, Developing a computational tool for predicting physical parameters of a typical VVER-1000 core based on artificial neural network, Ann. Nucl. Energy 50 (2012) 82–93.
[5] A. Garg, P.S. Sastry, M. Pandey, U.S. Dixit, S.K. Gupta, Numerical simulation and artificial neural network modeling of natural circulation boiling water reactor, Nucl. Eng. Des. 237 (2007) 230–239.
[6] X. Wei, J. Wan, F. Zhao, Prediction study on PCI failure of reactor fuel based on a radial basis function neural network, Science and Technology of Nuclear Installations (2016) 1–6. Article ID 4720685.
[7] M.I. Radaideh, T. Kozlowski, Analyzing nuclear reactor simulation data and uncertainty with the Group Method of Data Handling, Nucl. Eng. Technol. 52 (2020) 287–295.
[8] M.I. Radaideh, T. Kozlowski, Surrogate modeling of advanced computer simulations using deep Gaussian processes, Reliab. Eng. Syst. Saf. 195 (2020), 106731.
[9] J. Yang, J. Kim, An accident diagnosis algorithm using long short-term memory, Nucl. Eng. Technol. 50 (2018) 582–588.
[10] M.I. Radaideh, T. Kozlowski, Combining simulations and data with deep learning and uncertainty quantification for advanced energy modelling, Int. J. Energy Res. 43 (2019) 7866–7890.
[11] Y. Do Koo, Y.J. An, C.H. Kim, M.G. Na, Nuclear reactor vessel water level prediction during severe accidents using deep neural networks, Nucl. Eng. Technol. 51 (2019) 723–730.
[12] J.J. Ortiz, I. Requena, Using a multi-state recurrent neural network to optimize loading patterns in BWRs, Ann. Nucl. Energy 31 (2004) 789–803.
[13] J.L. Montes, J.L. Francois, J.J. Ortiz, C. Martin-Del-Campo, R. Perusquia, Local power peaking factor estimation in nuclear fuel by artificial neural networks,

Ann. Nucl. Energy 36 (2009) 121–130.
[14] V.A. Phung, D. Grishchenko, S. Galushin, P. Kudinov, Prediction of in-vessel debris bed properties in BWR severe accident scenarios using MELCOR and neural networks, Ann. Nucl. Energy 120 (2018) 461–476.
[15] T. Lefvert, Ringhals-1 Stability Benchmark: Final Report, NEA/NSC/DOC(96) 22, OECD Nuclear Energy Agency, 1996.
[16] T. Downar, Y. Xu, V. Seker, A. Ward, PARCS: Purdue Advance Reactor Core Simulator, Presented at Physics of Reactors (PHYSOR-2002), Seoul, Korea, October 7-10, 2002.
[17] A. Zameer, S.M. Mirza, N.M. Mirza, Core loading pattern optimization of a typical two-loop 300 MWe PWR using Simulated Annealing (SA), novel crossover Genetic Algorithms (GA) and hybrid GA (SA) schemes, Ann. Nucl. Energy 65 (2014) 122–131.
[18] E.F. Faria, C. Pereira, Nuclear fuel loading pattern optimisation using a neural network, Ann. Nucl. Energy 30 (2003) 603–613.
[19] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (2015) 436–444.
[20] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, F.E. Alsaadi, A survey of deep neural network architectures and their applications, Neurocomputing 234 (2017) 11–26.
[21] G. Hinton, L. Deng, D. Yu, G. Dahl, A.R. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, B. Kingsbury, T. Sainath, Deep neural networks for acoustic modeling in speech recognition, IEEE Signal Process. Mag. 29 (2012) 82–97.
[22] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, The MIT press, London, England, 2016.
[23] L. Bottou, Large-scale machine learning with stochastic gradient descent, in: Proc. of 19th Int. Computational Statistics (COMPSTAT'2010), Paris, France, August 22-27, 2010 vol. 1, Physica-Verlag HD, 2010, pp. 177–186.
[24] D.P. Kingma, J. Ba, Adam: A Method for Stochastic Optimization, 2014 arXiv preprint arXiv:1412.6980.
[25] F. Chollet, Keras. https://keras.io, 2015.