Fusion Engineering and Design 191 (2023) 113514

Contents lists available at ScienceDirect

Fusion Engineering and Design

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

Neural network model of neutral beam injection in the EAST tokamak to enable fast transport simulations

Z. Wang [a,*], S. Morosohk [a], T. Rafiq [a], E. Schuster [a], M.D. Boyer [b], W. Choi [c]

[a] Department of Mechanical Engineering and Mechanics, Lehigh University, Bethlehem, PA 18015, USA
[b] Princeton Plasma Physics Laboratory, Princeton, NJ 08543-0451, USA
[c] General Atomics, San Diego, CA 92121, USA

ARTICLE INFO

Keywords:
NUBEAM
Neutral beam injection
Neural network model
DNN
EAST

ABSTRACT

The neutral beam injection (NBI) system in EAST produces energetic neutral particles, which collide with electrons and ions in tokamak plasmas and heat the plasmas through Coulomb collisions. Moreover, it drives a non-inductive source of current, due to the charge-exchange collision between neutral particles and ions, and injects toroidal torque, which generates a toroidal rotation of the plasma. The effect caused by the NBI system, such as plasma heating, current drive, total neutron rate, momentum transfer, and shine-through, are modeled by a comprehensive module called NUBEAM. However, NUBEAM is computationally intensive since it relies on Monte Carlo methods. In this work, a neural network model has been developed as a surrogate model for NUBEAM in EAST. The database for neural-network model training, validation and testing is generated by running TRANSP for experimental discharges from recent EAST campaigns (after the latest NBI upgrade) while using the NUBEAM module. Simulation results illustrate that the trained neural network has the capability of replicating the predictions made by NUBEAM while demanding a significantly shorter execution time. These results indicate that surrogate models like the one proposed in this work could enable fast transport simulations for EAST after integrating them into a control-oriented predictive code such as COTSIM.

1. Introduction

Making nuclear-fusion energy commercially viable on a tokamak device requires stable and high-performance operation under advanced tokamak (AT) scenarios [1], which are characterized by improved confinement, magneto-hydro-dynamics (MHD) stability, high fusion gain, and possible steady-state operation. The realization of those AT scenarios relies on handling several plasma control problems simultaneously with the limited set of available plasma actuators. Developing such control capability is feasible only with a good understanding of both the complex physics governing the transport in tokamak plasmas and the plasma response to the different available actuators. Thus, tremendous effort has been put on building predictive codes, such as TRANSP, ONE-TWO, CRONOS, and ASTRA, with the capability of carrying out transport simulations. Those high-fidelity physics-oriented simulation codes have been proven to be reliable for analyzing shots after experiments and predicting plasma performance before experiments. However, because they are computationally intensive, embedding them in real-time closed-loop control algorithms or using them to run feedforward optimizations between shots become unaccomplishable tasks. Therefore, having fast transport simulation codes that can deliver prediction results with a level of accuracy similar to those offered by more computationally expensive codes is crucial for many control applications.

Machine learning (ML) techniques have been recently employed for many fusion applications such as disruption prediction and fault detection [2–4], plasma control [5,6], and fast plasma equilibrium solvers [7,8]. ML has also been used for the development of surrogate models for physics-oriented codes such as GENRAY [9], NUBEAM [10,11], TGLF/EPED [12], and MMM [13]. These surrogate models have the potential of producing high levels of prediction accuracy with a much lower computational burden, enabling in this way fast transport simulations once they are integrated into control-oriented predictive codes. A data-driven modeling approach based on a deep neural network is used in this work for the development of a surrogate model of NUBEAM with the ultimate goal of enabling fast transport simulations for EAST. Inspired by a similar concept originally presented in [10,11], this work uses the same machine learning technique (i.e., Multi-Layer Perceptron (MLP)) but proposes a different data-processing approach. Instead of using a first-order low-pass filter to handle the beam slowing-down time effects [14,15], the history information of the beam power

* Corresponding author.
E-mail address: zibo.wang@lehigh.edu (Z. Wang).

https://doi.org/10.1016/j.fusengdes.2023.113514
Received 1 November 2022; Received in revised form 2 January 2023; Accepted 31 January 2023
Available online 13 February 2023

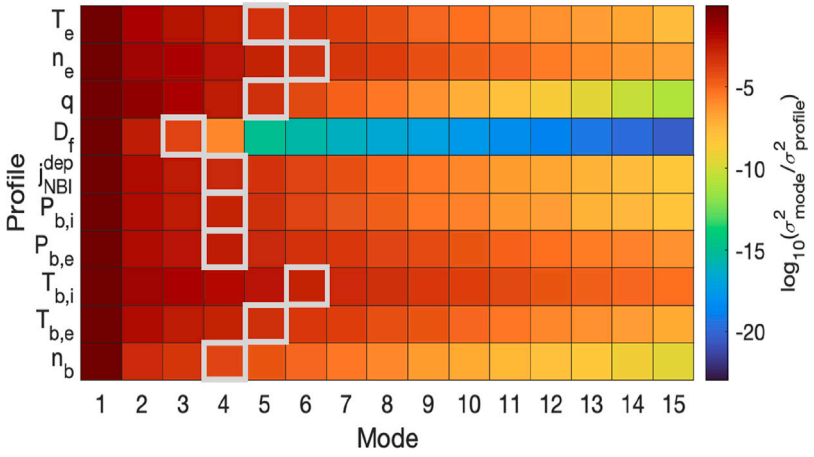

Fig. 1. Relative explained variance of modes generated by PCA. The number of modes kept for each profile to achieve greater-than-99.9% relative explained variance is highlighted with white boxes.

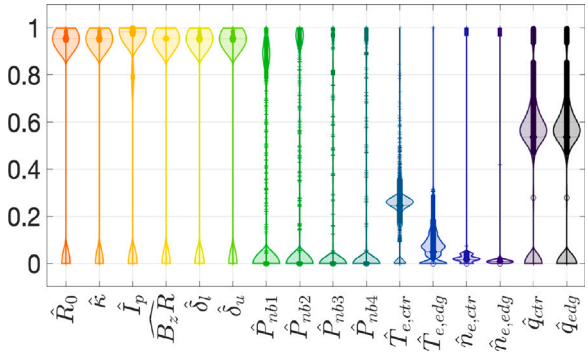

Fig. 2. Range of normalized inputs in the training dataset, where the notation $(\cdot)_{edg}$ stands for values at the plasma edge, $(\cdot)_{ctr}$ stands for values at the plasma core, and BzR is the vacuum toroidal field.

is directly used as input for training. Moreover, data is normalized instead of being standardized since in this work the assumption that data follows a Gaussian distribution is not adopted. Finally, the neural network configuration (i.e., number of layers and active function at each layer) is different when building a neural-network-based NUBEAM model for EAST.

This paper is organized as follows. In Section 2, the collection, generation, and division of the dataset used for neural-network training and testing are discussed. Next, the neural network model selection as well as the training and validation workflow are introduced in Section 3. Model-testing results are discussed in Section 4, while conclusions and future work are presented in Section 5.

2. Dataset generation and data processing

EAST is a long pulse superconducting tokamak with an ITER-like tungsten divertor. One of its scientific goals is to develop AT scenarios for ITER operations. Heating and current drives (H&CD) are crucial to realize fully non-inductive operation with a high bootstrap current fraction. The NBI system, which is one of the most effective H&CD sources at EAST, has two beamlines named NBI1 and NBI2. For each beamline, two hot-cathode ion sources coded with 'L' and 'R' are operated individually. This results in an NBI system with a total of four independent beams (i.e., NBI1L, NBI1R, NBI2L, and NBI2R). After recent upgrades, all beams are oriented in a counterclockwise injection direction, which is consistent with the direction of the plasma current [16].

The computationally intensive NUBEAM module [17] is frequently used to predict neutral-beam effects, such as plasma heating, current drive, total neutron rate, momentum transfer, and shine-through. A surrogate model of NUBEAM is proposed in this work by following a data-driven modeling approach based on neural networks. Experimental data collected from around 100 experimental discharges after the EAST recent upgrade have been used to build a database D_e. Because the neural network architecture used in this work cannot extrapolate, the accuracy of prediction with inputs outside the range of the training data is not guaranteed. Therefore, the training dataset should be large enough to generate a practical and reliable model. With this goal in mind, the database D_e has been further augmented by modifying parameters in reasonable ranges, such as the effective atomic number ($Z_{eff} \subset [1.5, 2.5]$) and the edge neutral density ($n_{0,edg} \subset [10^{11}, 10^{12}]$) to create a larger database D_x. Then, the database D_x has been fed into TRANSP to generate another dataset D_t for training a neural network model. When running TRANSP, the NUBEAM module is activated, the time step is set to 1 ms, the number of points of the spatial grid is chosen as 20, and the number of particles used in the Monte Carlo simulation is selected as 16,000 in order to smoothen the predictions and increase their fidelity. As a result, roughly 100,000 time slices have been collected and used as the dataset for neural network modeling. The 14 inputs (d_{in}) and 11 outputs (d_{out}) for the model are listed in Table 1.

It is worth noting that the anomalous fast ion diffusivity is computed as

$$D_f(\hat{\rho}) = D_{f,1} + (D_{f,0} - D_{f,1})(1 - \hat{\rho}^{\alpha_f})^{\beta_f}, \tag{1}$$

where $D_{f,0}$, $D_{f,1}$, α_f, and β_f are scaling parameters. For each simulation, either the classical ($D_{f,0} = D_{f,1} = 0, \alpha_f = 1, \beta_f = 1$), flat ($D_{f,0} = D_{f,1} = D_{f,mag}, \alpha_f = 1, \beta_f = 1$), or peak ($D_{f,0} = 0, D_{f,1} = D_{f,mag}, \alpha_f = 2, \beta_f = 4$) spatial profile is selected at the beginning, where $D_{f,mag} \in [10^{-4}, 5]$ m^2 s^{-1} is a randomly picked number. Before using the results from the TRANSP simulations to train a neural network model, the NUBEAM input–output data d_{in} and d_{out} is processed by including the history of beam powers to account for the possible beam slowing-down time effects, reducing the dimension of the profile data by using principal component analysis, and normalizing the data to make each feature standardized.

2.1. Beam slowing down time effects

Since the effect of neutral beam injection acting on the plasma is not an instant process, the history of beam power injection has to be considered. Many approaches can realize the goal of incorporating history information into the training process, such as low-pass filtering

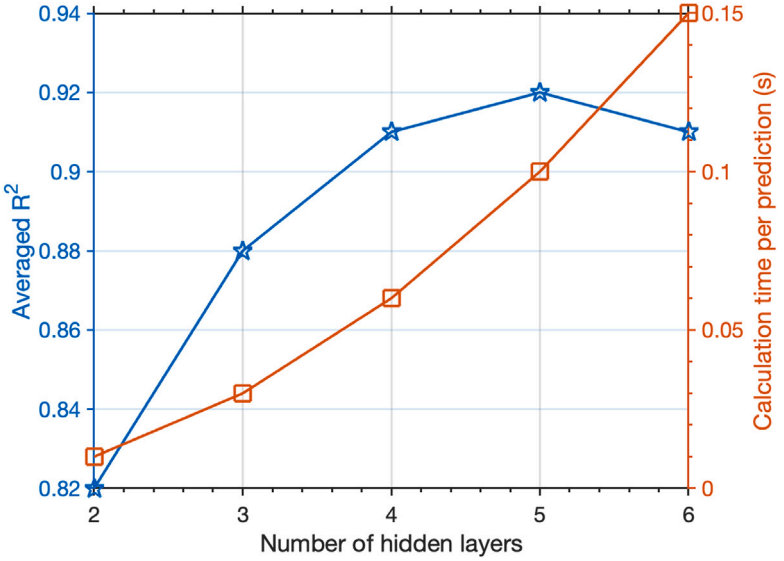

Fig. 3. Comparison of prediction accuracy and execution time with different hidden layers in the neural network structure.

Table 1
List of inputs and outputs of neutral beam model.

	Symbol	Explanation
Input: d_{in}	Z_{eff}	Mean charge of impurities
	$n_{0,edg}$	Edge neutral density (m^{-3})
	R_0	Major radius (m)
	κ	Elongation
	I_p	Plasma current (A)
	a	Minor radius (m)
	$B_{\phi,v}R$	Vacuum toroidal field (T m)
	δ_u	Upper triangularity
	δ_l	Lower triangularity
	$P_{NBI,1-4}$	Injected power for each beam (W)
	T_e	Electron temperature profile (eV)
	n_e	Electron density profile (m^{-3})
	q	Safety factor profile
	D_f	Anomalous fast ion diffusivity (m^2/s)
Output: d_{out}	$S_{neutron}$	Total neutron rate (s^{-1})
	P_{shine}	Shine-through power (W)
	P_{cx}	Charge-exchange power loss (W)
	P_{orb}	Orbit power loss (W)
	$P_{b,e}$	Beam heating to electrons (W m^{-3})
	$P_{b,i}$	Beam heating to ions (W m^{-3})
	$T_{b,e}$	Beam torque to electrons (N m/m^3)
	$T_{b,i}$	Beam torque to ions (N m/m^3)
	n_b	Beam ion density (m^{-3})
	j^{dep}_{NBI1-4}	Beam current drive for each beam (A/m^2)
	P_{fast}	Fast ion pressure (Pa)

the individual beam power as shown in [10,11] or using recurrent neural networks (RNNs). In this work, one-second history information of individual beam power injections is added to the input dataset. In another words, at time $t = t_j$ the input $P^{t_j}_{NBI,i}$ is expanded as

$$P^{t_j}_{NBI,i} = [P^{t=t_j}_{NBI,i}, P^{t=t_{j-1}}_{NBI,i}, \ldots, P^{t=t_{j-100}}_{NBI,i}], \tag{2}$$

where $i \in \{1,2,3,4\}$ and $\Delta_t = t_j - t_{j-1} = 10$ ms.

2.2. Profile data reduction via PCA

As the size of the input dataset becomes larger, the amount of time spent in training a neural network increases. And more importantly, it takes more time to make predictions when executing the trained neural network. Because the ultimate goal of using surrogate models is to enable fast transport simulations, the speed of prediction cannot be sacrificed. In TRANSP, spatially dependent plasma properties (usually referred to as profiles) such as the electron temperature and electron density are represented as discrete points uniformly distributed on the toroidal normalized mean effective minor radius $\hat{\rho} \in (0,1)$. In this work $\Delta\hat{\rho} = 0.025$, which demands the handling of 40-point profiles. While a convolutional neural network (CNN) architecture is widely used for two-dimensional training data (i.e., spatially + temporally varying dataset), CNN also requires a larger dataset compared to MLP and more time to make a prediction. In this work, an alternative method known as the Principal Component Analysis (PCA) is employed to handle two-dimensional data, which is commonly used in MLP architectures. In general, techniques such as PCA are applied to reduce the dimensionality of profile data and accelerate prediction speed. Essentially, the PCA is an orthogonal linear transformation. It orderly projects each spatially dependent quantity onto a set of basis functions with a rank of variances from the maximum to the minimum. As a result, the profile data is written as a linear combination of the basis functions. The profile data is reduced by only keeping a minimum number of modes (e.g., keeping two modes means only retaining two basis functions with the greatest and second greatest variances) while capturing the majority of the variance. The number of kept modes is determined by the minimum number of modes making the relative explained variance σ_r^2 greater than 99.9%, where σ_r^2 is defined as

$$\sigma_r^2 \triangleq \sigma^2_{mode}/\sigma^2_{profile}. \tag{3}$$

In Eq. (3), σ^2_{mode} is the explained variance corresponding to the kept modes and $\sigma^2_{profile}$ is the explained variance of the profile. The heat map of $\log(\sigma_r^2)$ is plotted in Fig. 1, where the number of modes guaranteeing $\sigma_r^2 > 99.9\%$ are boxed in white rectangles. Applying the PCA technique not only makes training and prediction fast but also filters out measurement noise in the input data and variance caused by the Monte Carlo scheme.

2.3. Data normalization

Neural network training is sensitive to the magnitude of input and output data. Therefore, directly passing unnormalized variables to the training procedure could lead to a slow or unstable learning process. In the worst-case scenario, the neurons in the hidden layers could saturate and the learning process could fail due to exploding gradients. In this work, input and output data are normalized by

$$\hat{x} = f_n(x) = \frac{x - min(x)}{max(x) - min(x)} * 0.99 + 0.01, \tag{4}$$

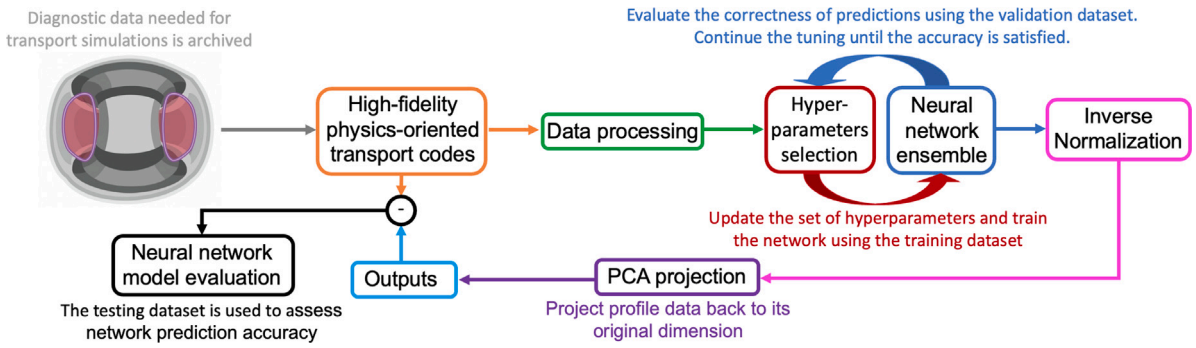

Fig. 4. Workflow diagram for building a real-time capable neural-network-based surrogate model. Raw data used for transport simulation is prepared and fed into transport codes. Then, the data are processed as described in Section 2 and the hyperparameters are tuned as discussed in Section 3 to train the neural network. After inverse normalization and PCA projection, the accuracy of the neural-network model is determined by comparing the prediction results with data in the testing dataset, as shown in Section 4.

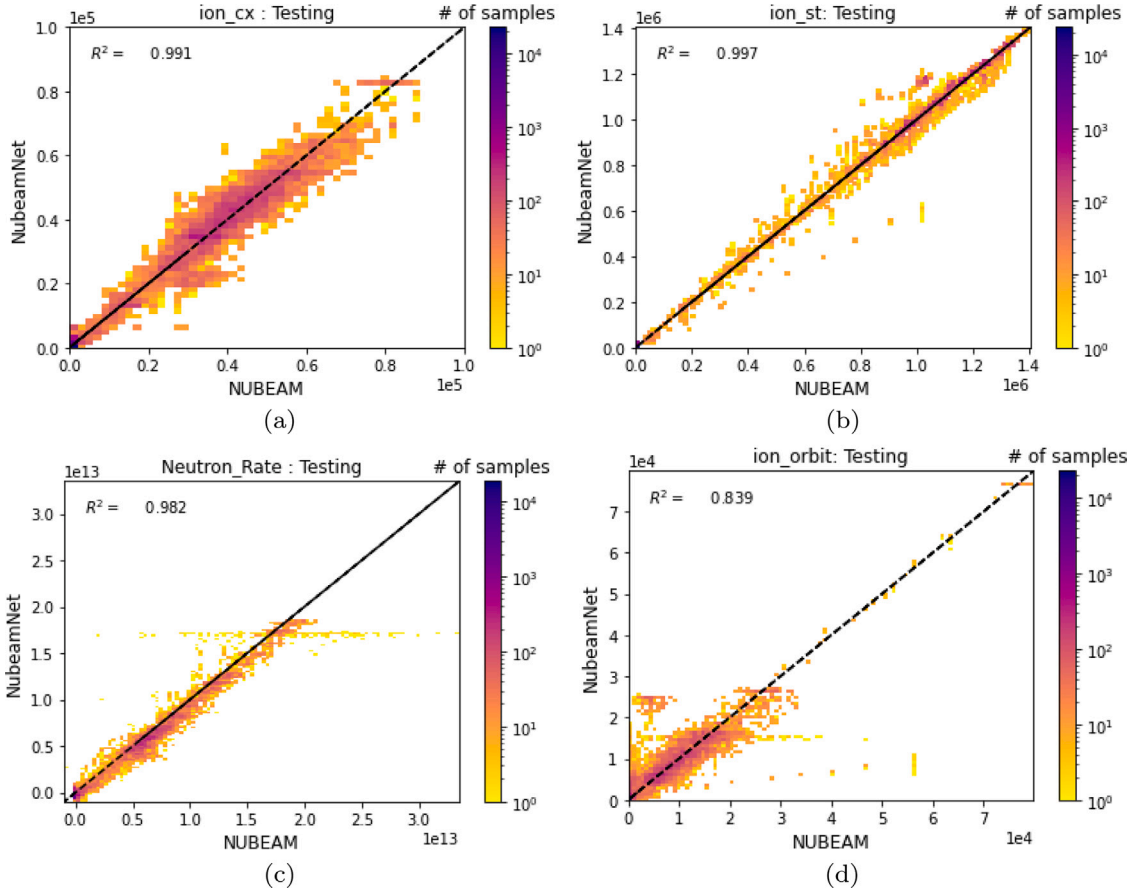

Fig. 5. Histograms of regression results for shots in the testing dataset: (a) charge-exchange power loss (W); (b) shine-through power (W); (c) total neutron rate (s^{-1}); (d) orbit power loss (W).

where $min(\cdot)$ and $max(\cdot)$ are the minimum and maximum of a vector, x is the original data, and $\hat{x} \in [0.01, 1]$ is the normalized data. A bias is introduced in (4) to make the neural network model trainable at the minimum value. Predictions can be converted back into the original scale since the transformation can be easily inverted. As a result, the range of normalized inputs in the training dataset is shown in a violin plot (Fig. 2). For example, the figure shows that R_0 in the dataset has a high density around its maximum value but a sparse density between maximum and minimum. And since the maximum and minimum values for each parameter are archived (i.e., $max(R_0) = 1.96$ m and $min(R_0) = 0.95$ m), by applying the inverse function of f_n in (4), the median of R_0 indicated by the thick horizontal line can be expressed in term of its original value (1.87 m).

3. Method and workflow

Training a surrogate model for NUBEAM with a faster execution time and still high prediction accuracy is the overall objective of this work. This goal is achieved by using a fully connected neural network topology. A neural network named 'NUBEAMnet' is coded in a Python environment which takes advantage of the TensorFlow Distributions library [18]. The processed dataset from Section 2 is randomly divided into three groups, where 80% is labeled as training (i.e., to train the neural-network model), 10% is labeled as validation (i.e., to determine optimal values of hyperparameters), and the rest is labeled as testing (i.e., to assess network prediction accuracy).

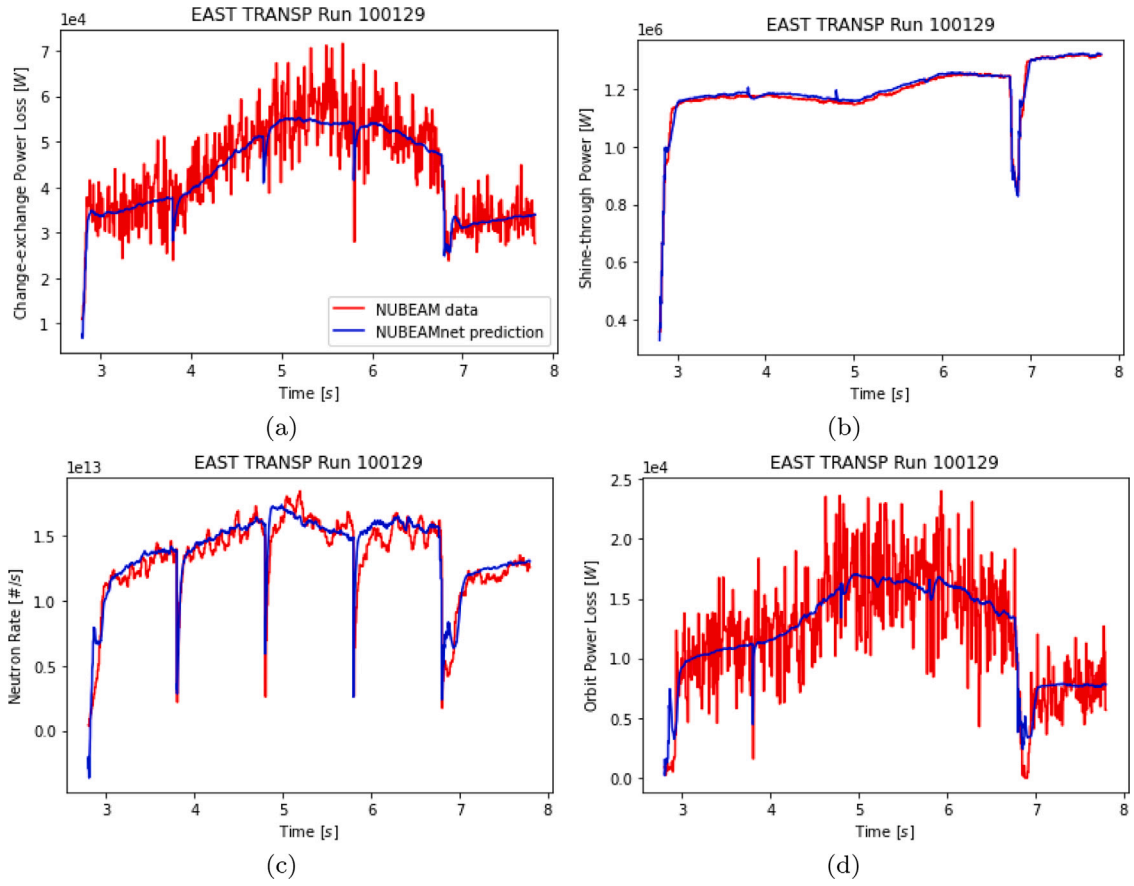

Fig. 6. Comparisons of time evolutions predicted by NUBEAM and NUBEAMnet: (a) charge-exchange power loss (W); (b) shine-through power (W); (c) total neutron rate (s^{-1}); (d) orbit power loss (W).

The initial values for the hyperparameters are set as shown in Table 2 and are considered as the minimum values for the hyperparameters. Then, by gradually increasing the value for each hyperparameter and comparing the R^2 coefficient of linear regressions, a balance of prediction accuracy and training time can be found. An example of tuning the number of hidden layers is shown in Fig. 3. As the number of hidden layers increases, the prediction quality improves rapidly at the beginning as shown in the blue line. However, the red line shows that the training time increases significantly as the number of hidden layers increases, but accuracy improvement slows down when the number of hidden layers reaches four and the execution time increases dramatically. Moreover, a large number of hidden layers could lead to overfitting as the model gets more complex. Thus, a neural network with four hidden layers is preferred in this case. Nevertheless, other parameters, such as nodes in each layer, also affect prediction accuracy and execution time. Therefore, after scanning the accuracy of model predictions with different sets of hyperparameters, the final selection is presented in Table 2. Other hyperparameters, such as activation functions for each layer, loss function used in back-propagation, and accuracy metrics used to evaluate the goodness of predictions, are determined based on a similar methodical scanning in order to optimize prediction accuracy.

A general workflow for developing data-driven surrogate models in a Python environment for any high-fidelity physics-oriented module via MLP is shown in Fig. 4. As shown in the figure, the data archived from experiments is submitted to high-fidelity physics-oriented transport codes. The results from the transport code are processed as discussed in Section 2 and fed to a neural network. Then the hyperparameters of MLP are tuned by evaluating the correctness of predictions using the validation dataset. Finally, the accuracy of the surrogate model is assessed by using the testing dataset. The outputs from the surrogate model are firstly transformed from the normalized form to the original scale using the inverse function of f_n in (4); next, the profile outputs are reconstructed from the reduced PCA modes; finally, the processed outputs are compared with the predictions by the transport code.

Table 2
Selection of hyperparameters via methodical scanning.

	Initial values	Final values
Number of hidden layers	2	3
Nodes per hidden layers	10	72
Number of batch size	5	50
Number of epochs	5	100
Rate of learning	0.1	0.01

4. Model evaluation and discussion

After assembling and training the MLP network, the testing dataset is used to evaluate the quality of the trained network. NUBEAMnet and NUBEAM (from the TRANSP runs) predictions are compared to demonstrate the capability of the neural-network model.

Log-scale histograms of regression results for scalar predictions are presented in Fig. 5, where (a) charge-exchange power loss, (b) shine-through power, (c) total neutron rate, and (d) orbit power loss are illustrated. They all show good concentrations of data points along the diagonal line, especially for the charge-exchange power loss, shine-through power, and total neutron rate. This means that the predictions of scalars made by the trained neural network are highly consistent with NUBEAM calculations, which can also be concluded from the correlation values (R^2) close to 1 in the upper-left corners. However,

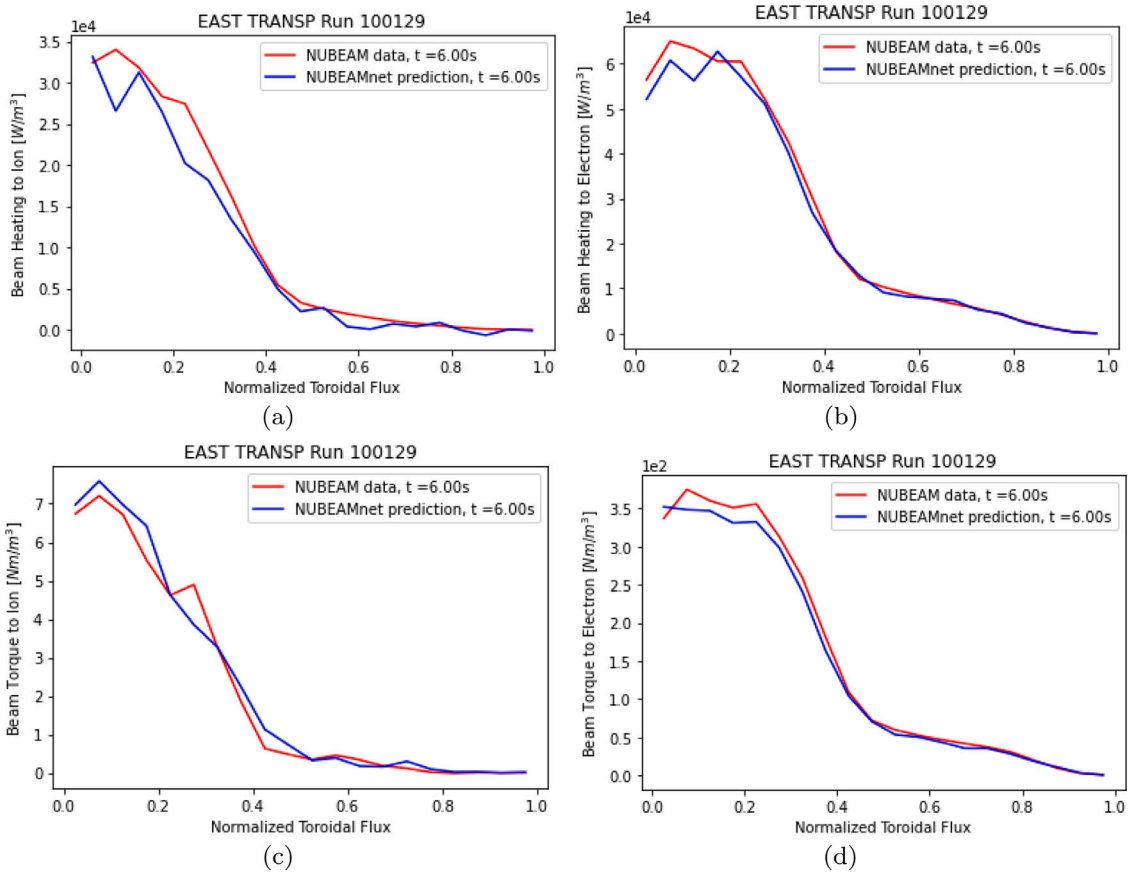

Fig. 7. Comparisons of profiles predicted by NUBEAM and NUBEAMnet, which are plotted as functions of the toroidal normalized mean effective minor radius $\hat{\rho}$: (a) beam heating to ions (W m^{-3}); (b) beam heating to electrons (W m^{-3}); (c) beam torque to ions (N m/m^3); (d) beam torque to electrons (N m/m^3) for shot #100129 at $t = 6$ s.

the prediction of the orbit power loss shows room for improvement. One of the possible reasons for the exhibited lower accuracy is that scenarios with low orbit power loss may not be well learned, which could be improved by expanding the training dataset with more low-orbit power-loss cases. Fig. 6 further compares NUBEAM and NUBEAMnet predictions for shot #100129 by using time traces. The NUBEAMnet prediction results have good agreement with NUBEAM and are much smoother. In a Python environment, the average execution time for NUBEAMnet to make predictions for a scalar output is about 0.1 ms per time step, and for all four scalars is about 0.2 ms per time step, which are orders of magnitude faster than the NUBEAM module (running with 16 parallel processes, the wall time for the NUBEAM calculation of each time step (10 000 particles) is roughly 5 s). These execution times enable the use of NUBEAMnet in both off-line (e.g., closed-loop simulations for control-performance assessment and scenario planning via model-based optimization) and real-time (feedback controllers, state observers and optimizers) control applications.

Fig. 7 draws comparisons between NUBEAM-predicted (red lines) and NUBEAMnet-predicted (blue lines) profile results as functions of the normalized toroidal flux coordinate $\hat{\rho}$ at 6 s. The figure shows that NUBEAM and NUBEAMnet predictions match well. However, NUBEAMnet predictions for beam heating to ions and electrons show deviations at around $\hat{\rho} = 0.05$. Adding more modes for beam heating to ions and electrons when applying the PCA could alleviate this problem. Alternatively, the result could be improved by increasing the number of points in the spatial grid (e.g., 80) when running TRANSP with the NUBEAM module and using CNN architectures to avoid the PCA technique. Comparison between NUBEAM and NUBEAMnet predictions for the beam current deposition profile for NBI_{1L} at 3.5 s and 6 s are shown in Fig. 8(a) and (b); while comparison for NBI_{1R} at 5.5 s and 6.5 s are shown in Fig. 8(c) and (d), respectively. Again, the NUBEAMnet predictions strongly agree with the NUBEAM predictions since the blue lines almost cover the red lines. Fig. 9 provides statistics of the matching errors calculated in terms of the root-mean-square (RMS) deviation, which is defined as

$$RMS = \sqrt{\sum_{j=1}^{N}(y_j^t - y_j^r)^2 / \sum_{j=1}^{N}(y_j^t)^2} \tag{5}$$

and where y_t represents the predictions by NUBEAM, y_r represents the predictions by NUBEAMnet, and N is the number of data points. The results in Fig. 9 show that NUBEAMnet and NUBEAM predictions highly agree in the core and in the middle region of the plasma, while having relatively higher mismatches at the plasma edge. This issue may be related to the PCA method since the magnitude of the beam power deposition is much lower at the edge when compared with the core. The average execution time for NUBEAMnet to make predictions for a profile output is about 1 ms in a Python environment, which is both consistent and comparable with predictions for scalar outputs as discussed before.

The summary of average correlations between NUBEAM and NUBEAMnet predictions is listed in Table 3, which shows that the proposed surrogate model is capable of replacing NUBEAM since the correlation factor R^2 is close to 1 for the testing dataset.

5. Conclusions and future work

In this work, a surrogate model based on neural network techniques has been developed for the NUBEAM module in EAST. The model is aimed at predicting the effects caused by the NBI system in EAST after the recent upgrade in 2020. The surrogate model is able to produce almost the same predictions as the NUBEAM module while taking

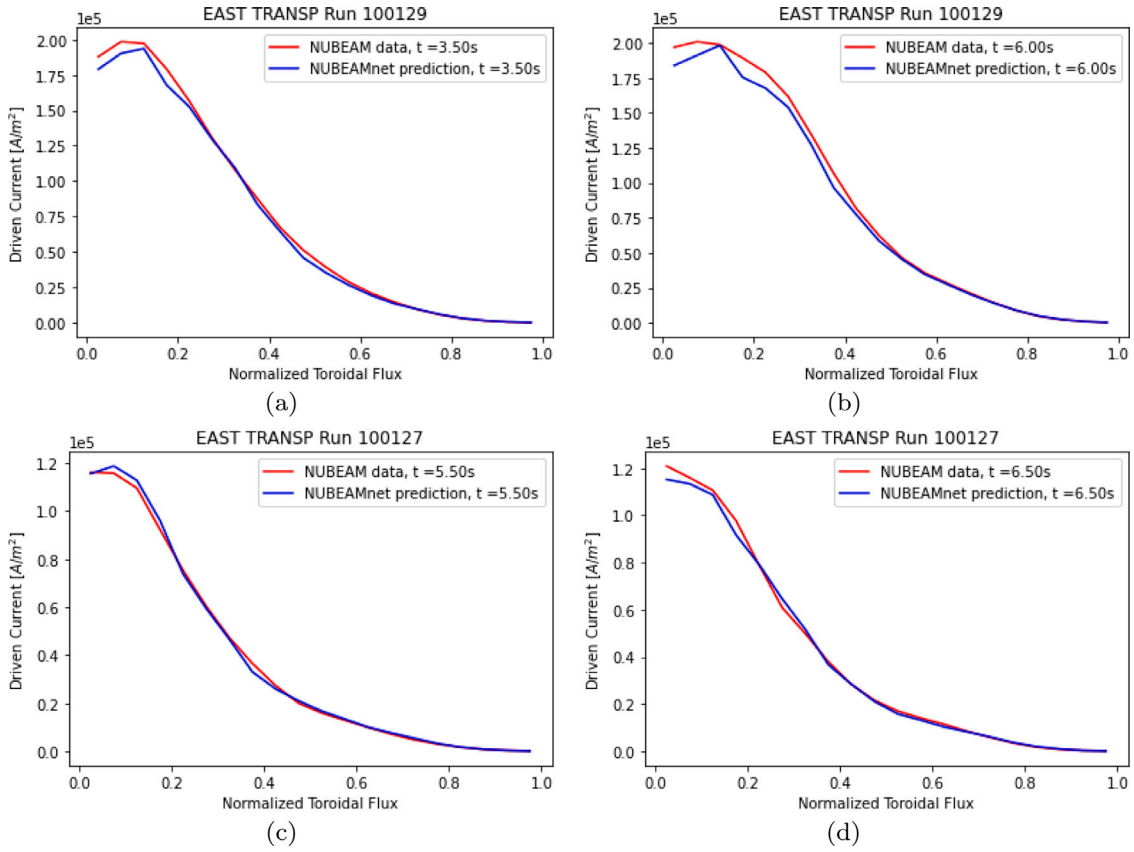

Fig. 8. Comparisons of current-deposition profiles predicted by NUBEAM and NUBEAMnet, which are plotted as functions of the toroidal normalized mean effective minor radius $\hat{\rho}$: (a) NBI_{1L} for shot #100129 at $t = 3.5$ s; (b) NBI_{1L} for shot #100129 at $t = 6$ s; (c) NBI_{1R} for shot #100127 at $t = 5.5$ s; (d) NBI_{1R} for shot #100127 at $t = 6.5$ s.

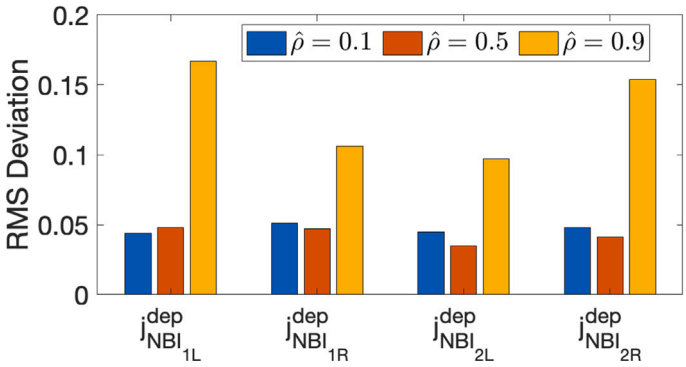

Fig. 9. RMS deviations between NUBEAM and NUBEAMnet predictions for the current deposition profile at three different points.

Table 3
Summary of correlations for NUBEAM and NUBEAMnet predictions for training and testing dataset.

	R^2: training data	R^2: testing data
P_{shine}	0.991	0.997
P_{cx}	0.985	0.991
P_{orb}	0.855	0.839
$P_{b,e}$	0.982	0.975
$P_{b,i}$	0.981	0.961
$T_{b,e}$	0.955	0.945
$T_{b,i}$	0.832	0.811
n_b	0.985	0.982
P_{fast}	0.991	0.953
j^{dep}_{NBI1-4}	0.985	0.962

orders of magnitude shorter execution times. In order to make the neural network model cover greater operating regimes, more discharges with different plasma scenarios could be used to train the network. Thus, future work includes expanding the database needed for neural network training. Once the database size is large enough, work will be conducted toward replacing the PCA technique in the current version of the surrogate model by a CNN in order to increase the profile prediction accuracy.

Another contribution of this work is the development of a workflow that can be applied to data-driven modeling with deep neural networks for any other modules in high-fidelity physics-oriented simulation codes. Thus, future work will start by replacing other modules (i.e., ray-tracing/absorption code (TORAY) [19] and Multi-Mode-Model (MMM) anomalous transport [20] modules) with neural network models. These models will be integrated into COTSIM to enable scenario

planning via feedforward optimization between shots and model-based advanced scenario control in real time.

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.

Acknowledgment

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Fusion Energy Sciences (FES), under Award Number DE-SC0010537.

References

[1] T. Taylor, Physics of advanced tokamaks, Plasma Phys. Control. Fusion 39 (12B) (1997) B47.
[2] K.J. Montes, C. Rea, R. Granetz, R.A. Tinguely, N. Eidietis, O. Meneghini, D. Chen, B. Shen, B. Xiao, K. Erickson, et al., Machine learning for disruption warnings on Alcator C-Mod, DIII-D, and EAST, Nucl. Fusion 59 (9) (2019) 096015.
[3] B. Guo, D. Chen, B. Shen, C. Rea, R. Granetz, L. Zeng, W. Hu, J. Qian, Y. Sun, B. Xiao, Disruption prediction on EAST tokamak using a deep learning algorithm, Plasma Phys. Control. Fusion 63 (11) (2021) 115007.
[4] D. Mohapatra, B. Subudhi, R. Daniel, Real-time sensor fault detection in tokamak using different machine learning algorithms, Fusion Eng. Des. 151 (2020) 111401.
[5] J. Seo, Y.-S. Na, B. Kim, C. Lee, M. Park, S. Park, Y. Lee, Feedforward beta control in the KSTAR tokamak by deep reinforcement learning, Nucl. Fusion 61 (10) (2021) 106010.
[6] J. Degrave, F. Felici, J. Buchli, M. Neunert, B. Tracey, F. Carpanese, T. Ewalds, R. Hafner, A. Abdolmaleki, D. de Las Casas, et al., Magnetic control of tokamak plasmas through deep reinforcement learning, Nature 602 (7897) (2022) 414–419.
[7] S. Joung, J. Kim, S. Kwak, J. Bak, S. Lee, H. Han, H. Kim, G. Lee, D. Kwon, Y.-C. Ghim, Deep neural network Grad-Shafranov solver constrained with measured magnetic signals, Nucl. Fusion 60 (1) (2019) 016034.
[8] L.L. Lao, S. Kruger, C. Akcay, P. Balaprakash, T. Bechtel, E. Howell, J. Koo, J. Leddy, M. Leinhauser, Y. Liu, et al., Application of machine learning and artificial intelligence to extend EFIT equilibrium reconstruction, Plasma Phys. Control. Fusion (2022).
[9] G. Wallace, Z. Bai, R. Sadre, T. Perciano, N. Bertelli, S. Shiraiwa, E. Bethel, J. Wright, Towards fast and accurate predictions of radio frequency power deposition and current profile via data-driven modelling: applications to lower hybrid current drive, J. Plasma Phys. 88 (4) (2022) 895880401.
[10] M. Boyer, S. Kaye, K. Erickson, Real-time capable modeling of neutral beam injection on NSTX-U using neural networks, Nucl. Fusion 59 (5) (2019) 056008.
[11] S.M. Morosohk, M.D. Boyer, E. Schuster, Accelerated version of NUBEAM capabilities in DIII-D using neural networks, Fusion Eng. Des. 163 (2021) 112125.
[12] O. Meneghini, S.P. Smith, P.B. Snyder, G.M. Staebler, J. Candy, E. Belli, L. Lao, M. Kostuk, T. Luce, T. Luda, et al., Self-consistent core-pedestal transport simulations with neural network accelerated models, Nucl. Fusion 57 (8) (2017) 086034.
[13] S. Morosohk, A. Pajares, T. Rafiq, E. Schuster, Neural network model of the multi-mode anomalous transport module for accelerated transport simulations, Nucl. Fusion 61 (10) (2021) 106040.
[14] Z. Chen, M. Nocente, M. Tardocchi, T. Fan, G. Gorini, Simulation of neutron emission spectra from neutral beam-heated plasmas in the EAST tokamak, Nucl. Fusion 53 (6) (2013) 063023.
[15] Y. Zheng, J. Xiao, B. Hao, L. Xu, Y. Wang, J. Zheng, G. Zhuang, Modeling of beam ions loss and slowing down with Coulomb collisions in EAST, Chin. Phys. B (2022).
[16] J. Wang, Y. Chen, B. Wu, Z. Yang, C. Hu, Y. Xie, Y. Xie, G. Zhong, L. He, F. Wang, Injection performance prediction of the upgraded neutral beam on EAST, Fusion Eng. Des. 166 (2021) 112277.
[17] A. Pankin, D. McCune, R. Andre, G. Bateman, A. Kritz, The tokamak Monte Carlo fast ion module NUBEAM in the National Transport Code Collaboration library, Comput. Phys. Comm. 159 (3) (2004) 157–184.
[18] B. Pang, E. Nijkamp, Y.N. Wu, Deep learning with tensorflow: A review, J. Educ. Behav. Stat. 45 (2) (2020) 227–248.
[19] A. Kritz, H. Hsuan, R. Goldfinger, D. Batchelor, Ray tracing study of electron cyclotron heating in toroidal geometry, in: Heating in Toroidal Plasmas 1982, Elsevier, 1982, pp. 707–723.
[20] T. Rafiq, A. Kritz, J. Weiland, A. Pankin, L. Luo, Physics basis of Multi-Mode anomalous transport module, Phys. Plasmas 20 (3) (2013) 032506.