NUBEAM SURROGATE MODELS BASED ON MLP, CNN, AND PARALLEL, CNN-LSTM NEURAL NETWORK ARCHITECTURES

Enabling fast transport simulations for control, estimation, and optimization

Z. WANG, T. RAFIQ, E. SCHUSTER, V. GRABER
Lehigh University
Bethlehem, Pennsylvania, USA
Email: zibo.wang@lehigh.edu

Abstract

The neutral beam injection (NBI) system is one of the most effective H&CD sources in tokamaks, as it generates highly energetic neutral particles that collide with plasma ions and electrons, heating the plasma through Coulomb collisions. It also drives a non-inductive current source through charge-exchange collisions with the plasma's ions and injects a toroidal torque that generates plasma rotation. The NUBEAM module predicts the effects of the NBI system on plasma heating, current drive, neutron rate, momentum transfer, and shine-through, but it is encumbered by its demanding computational requirements due to its reliance on Monte Carlo methods. In order to address this issue, three neural network (NN) architectures are used to develop three NN-based surrogate models that can process simulations more quickly than the NUBEAM module without sacrificing computational accuracy. The three NN-based surrogate models are (1) NUBEAMnet-1, which is based on Multi-Layer Perceptrons (MLP), (2) NUBEAMnet-2, which is based on a convolutional neural network (CNN), and (3) NUBEAMnet-3, which is based on a novel parallel architecture that combines CNN and Long Short-Term Memory (LSTM) networks. This work evaluates the NN-based surrogate models based on accuracy of prediction and execution time. The results indicate that the three proposed NUBEAMnets are viable alternatives to the NUBEAM module for the rapid simulation of NBI effects, which can benefit tasks such as between-shot analysis and real-time feedback control during an experiment. The work presented in this paper enables the development of control schemes that utilize NUBEAMnet to regulate the current profile by modulating the NBI power.

1. INTRODUCTION

Achieving the commercialization of nuclear fusion energy via tokamak devices requires stable and robust performance, particularly within the domain of advanced tokamak (AT) scenarios, characterized by enhanced plasma confinement, improved magneto-hydro-dynamic (MHD) stability, high fusion gain, and steady-state operation. Non-inductive current drive sources, such as Neutral Beam Injection (NBI) and various Radio Frequency (RF) heating methods, including Electron Cyclotron Resonance Heating (ECRH) and Lower Hybrid Current Drive (LHCD), enable the precise modulation of the plasma current and play an instrumental role in achieving AT scenarios. Extensive research endeavors have been undertaken to develop high-fidelity, physics-oriented computational tools tailored for simulating the effects of different auxiliary power sources on fusion-producing plasmas. For instance, the GENRAY [1] and TORAY [2] codes serve as the algorithmic frameworks for simulating the influence of RF interactions. The effects of NBI are well captured by the NUBEAM module [3], which stands out for its capability to simulate a wide range of complex phenomena caused by NBI, such as plasma heating, current drive, neutron rate, momentum transfer, and shine-through. Although these codes demonstrate outstanding proficiency and have been empirically validated, their significant computational demands limit their broader application, such as online model-based control and feedforward optimization.

Machine learning (ML) techniques have recently garnered attention for their applicability in various fusion-related tasks, providing transformative solutions to complex challenges. These applications include predictive models for plasma disruptions and fault detection [4, 5], fast plasma equilibrium solvers [6], and advanced plasma control schemes [7]. Moreover, ML methodologies serve as the foundation for constructing surrogate models for physics-oriented simulation modules like GENRAY[8] and NUBEAM [9, 10, 11]. The strength of these surrogate models lies in their capacity to deliver accurate predictions with substantially diminished computational requirements, thus facilitating rapid transport simulations when integrated into control-oriented plasma simulators.

In the current study, data-driven approaches based on neural network (NN) architectures has been employed to develop specialized NUBEAM surrogate models for the EAST tokamak. This approach is aimed at accelerating transport simulations without sacrificing accuracy. Building on prior work [11], three distinct NN-based surrogate models have been developed: NUBEAMnet-1, NUBEAMnet-2, and NUBEAMnet-3. These surrogate models leverage various combinations of deep neural network (DNN) architectures, including Multi-Layer Perceptrons

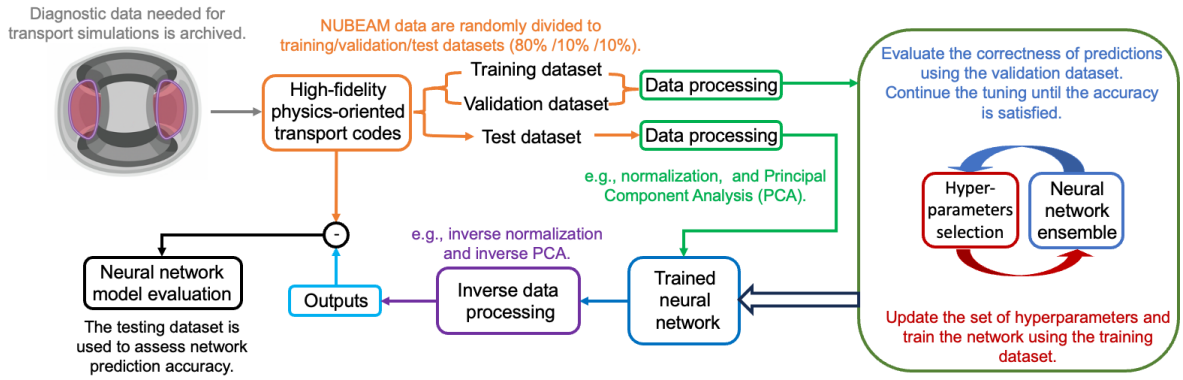

FIG. 1. Workflow diagram for building a real-time-capable, NN-based surrogate model. After data processing (e.g., normalization and standardization), a portion of the data is used to train the network and tune hyperparameters to obtain a good fitting. The surrogate model is then evaluated using a separate testing dataset.

(MLP), Convolutional Neural Networks (CNN), and an innovative parallel architecture integrating both CNN and Long Short-Term Memory (LSTM) networks. These surrogate models are designed to address different challenges that emerge when training a DNN for NUBEAM. For example, MLP networks are included primarily to speed up the training and querying processes. Meanwhile, CNNs are employed to handle 2D data, which is commonly encountered in the field of plasma physics. Finally, the combined CNN-LSTM networks manage the slowing-down time effects of the neutral beam injection [12].

The primary aim of this study is to investigate and develop various neural network architectures suitable for NUBEAM, ranging from simple to complex designs. These architectures will be evaluated using two critical metrics: predictive accuracy and computational speed. The goal of this evaluation is to validate the performance of the new surrogate models and to explore their potential for wider applications, such as enabling quick transport simulations for real-time optimization and analysis between experimental runs. The workflow for training, validating and testing the three proposed NN-based surrogate models is illustrated in Fig. 1.

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 three different versions of NUBEAMnet are introduced in Section 3. Model testing results are presented in Section 4, while conclusions and future work are discussed in Section 5.

2. DATASET GENERATION AND DATA PROCESSING

The Experimental Advanced Superconducting Tokamak (EAST) has been designed with a tungsten divertor similar to that proposed for ITER, and one of its primary objectives is to develop AT operational scenarios for ITER. Heating and Current Drive (H&CD) systems, essential for achieving non-inductive, steady-state operation, are integrated into the EAST design. Among these, the NBI system is particularly noteworthy, especially after its recent upgrade to align all beam injections in the direction of the plasma current [13].

To train machine learning models, a dataset, hereafter referred to as D_e, was constructed from nearly 120 experimental runs conducted after the NBI system's upgrade; this dataset consists of 14 inputs (d_{in}) and 11 outputs (d_{out}) as delineated by the NUBEAM model and listed in Table 1. Because the performance of ML-based surrogate models can worsen when applied outside the range of their training dataset, the parameter space of D_e is broadened by adjusting the effective atomic number, the edge neutral density, and the anomalous fast ion diffusivity within scientifically reasonable ranges. The ranges for these parameters are detailed in a separate study [11]. This led to an enriched dataset, which was subsequently used as an input to the TRANSP code with the NUBEAM module enabled, to generate a training dataset D_t.

In simulations carried out using TRANSP, the NUBEAM module was set to operate with a 1 ms time step and a spatial grid composed of 20 points. For the Monte Carlo simulations within NUBEAM, 16,000 particles were used to enhance the quality of the generated data. As a result, the dataset D_t consists of approximately 100,000 time slices, providing a substantial foundation for developing machine-learning-based predictive models. The data collection D_t is randomly divided into three distinct subsets: 80% is allocated for training the neural network, 10% is set aside for hyperparameter tuning through validation, and the remaining portion is used to evaluate the model's predictive capabilities.

TABLE 1. LIST OF INPUTS AND OUTPUTS OF NEUTRAL BEAM MODEL

	Symbol	Description
Input: d_{in}	Z_{eff}	Effective atomic number
	$n_{0,edg}$	Edge neutral density (m^{-3})
	R_0	Major radius (m)
	κ	Elongation
	I_p	Plasma current (A)
	a	Minor radius (m)
	$B_{\phi,\nu}R$	Vacuum toroidal field ($T \cdot m$)
	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 ($W \cdot m^{-3}$)
	$P_{b,i}$	Beam heating to ions ($W \cdot m^{-3}$)
	$T_{b,e}$	Beam torque to electrons (Nm/m^3)
	$T_{b,i}$	Beam torque to ions (Nm/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)

2.1. Data Normalization

Normalization and standardization are common techniques employed to transform feature variables (i.e., d_{in} and d_{out}) and improve the convergence behavior during model training. Standardization modifies the feature distribution to have zero mean and unit variance. Although standardization is advantageous when handling datasets with Gaussian distributions, it may not be suitable for all datasets. On the other hand, normalizing feature variables to lie within a predefined range, often $[0, 1]$, is a more generalizable technique. In this study, the feature variables are normalized, which imposes fewer assumptions on the data distribution. The formula used for normalization is given by

$$\hat{x} = f_n(x) = 0.01 + 0.99\frac{x - \min(x)}{\max(x) - \min(x)}, \tag{1}$$

where $\min(\cdot)$ and $\max(\cdot)$ denote the minimum and maximum values in the feature vector. Vector x is the original data, and $\hat{x}$ is the normalized feature within the range $[0.01, 1]$. The inclusion of a slight bias term (1) ensures the network remains trainable across the entire feature range. Notably, this transformation is reversible, facilitating the re-scaling of predictions to the original data scale.

3. METHOD AND WORKFLOW

The workflow for developing data-driven surrogate models using Python is outlined in Fig. 1. Initially, data gathered from experimental runs are used as input to high-fidelity transport codes oriented toward physics simulations. The results are then normalized (1), and then they are fed into different neural network architectures. Hyperparameter tuning follows, leveraging a validation dataset to refine the model's predictive accuracy. Once validated, the model's performance is assessed using a separate testing dataset. The model's output is then denormalized to its original scale using the inverse of (1). Finally, the output is compared with predictions from the original transport code to evaluate the model's accuracy.

3.1. NUBEAMnet-1 (MLP-based Surrogate Model)

NUBEAMnet-1, which leverages a fully connected MLP, was developed in prior work [11]. To reduce the complexity of the two-dimensional data in d_{in}, Principal Component Analysis (PCA) was used. The PCA method transformed the spatially dependent variables, which were distributed at uniformly-spaced points across the normalized minor radius $\hat{\rho}$, into a set of orthogonal components. By keeping only the components that account for a large portion of the data variance (greater than 99.9%), the computational time for both training and prediction was significantly reduced. Additionally, this approach filters out noise in the dataset. Because the effects of neutral beam injection on the plasma are not an instantaneous (i.e., beam slowing-down time effects), the time history of injected beam power was taken into account. Therefore, one hundred snapshots of the NBI power were taken every second for each individual beam. In another words, at time $t = t_j$ the input $P_{NBI,i}^{t_j}$ can be written as

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

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

The implemented MLP network comprises an input layer followed by three hidden layers. The first hidden layer contains 120 neurons with a Rectified Linear Unit (ReLU) activation function. The second hidden layer consists of 80 neurons each, and it is also activated by ReLU functions. Subsequently, another hidden layer featuring 40 neurons uses the hyperbolic tangent (tanh) as its activation function. Finally, the network culminates in an output layer with a single neuron activated by a linear function. The model employs the Mean Squared Error (MSE) loss function and was optimized using the Adam algorithm with a learning rate of 10^{-4} for the backpropagation algorithm.

3.2. NUBEAMnet-2 (CNN-based Surrogate Model)

NUBEAMnet-2 has been developed based on a CNN. When training a neural network based on two-dimensional data (i.e., spatially and temporally varying dataset), deep learning models based on CNNs are preferred over those based on PCA-MLPs because they can adaptively learn and detect complex patterns in the data. Because PCA uses a fixed linear transformation to reduce dimensionality, some information in the dataset gets left behind such that important nonlinear relationships may not be fully captured. CNNs, however, can dynamically adapt their feature extraction process based on the specific prediction task, thus offering improved accuracy when processing two-dimensional data. Therefore, when handling complex datasets with spatial and temporal distribution, CNNs are generally more accurate than PCA-MLP-based methods.

3.2.1. Convolutional Neural Network Design

One-dimensional Convolutional Neural Networks (1D-CNNs) are a specialized subset of CNNs tailored for the analysis of time-series spatial data. These networks incorporate one-dimensional convolutional layers, which perform convolution operations with learnable kernels. These kernels slide across the input sequence to yield feature maps, that summarize localized characteristics of the input. Typically, convolutional layers are denoted mathematically as

$$F(x) = \sum_{\tau} x(\tau) w(t - \tau), \tag{3}$$

where $F(x)$ is the feature map, x represents the input, w signifies the kernel, and t indicates the time index. Following the convolutional layers are pooling layers, which serve to reduce the feature map's dimensions. Max-pooling, a commonly used technique, extracts the maximum value from each group of values in the feature map. This operation reduces the computational burden and introduces a form of translational invariance.

The CNN topology in this study is formulated through the TensorFlow package. The architecture commences with a 1D-convolution (1D-Conv) layer that contains 32 filters, each of a kernel size of 10. The ReLU activation function is applied at this stage. This layer receives inputs with dimensions corresponding to the number of timesteps and features. Following the initial convolutional layer, a max-pooling operation is conducted with a pool size of 3, primarily to reduce dimensionality and to emphasize the most salient features. Subsequently, the pooled features are flattened to create a single vector, thereby preparing the dataset for a transition from convolutional layers to fully connected layers. The fully connected segment of the network consists of three layers. In each case, the activation function is ReLU. Ultimately, the architecture culminates in an output layer. To optimize the network's performance, the Adam optimizer is employed with a learning rate of 0.01. The network is trained to minimize the MSE loss function and aims to maximize accuracy as the metric of interest. This design choice reflects the complex and high-dimensional nature of the input dataset that is relevant to this work.

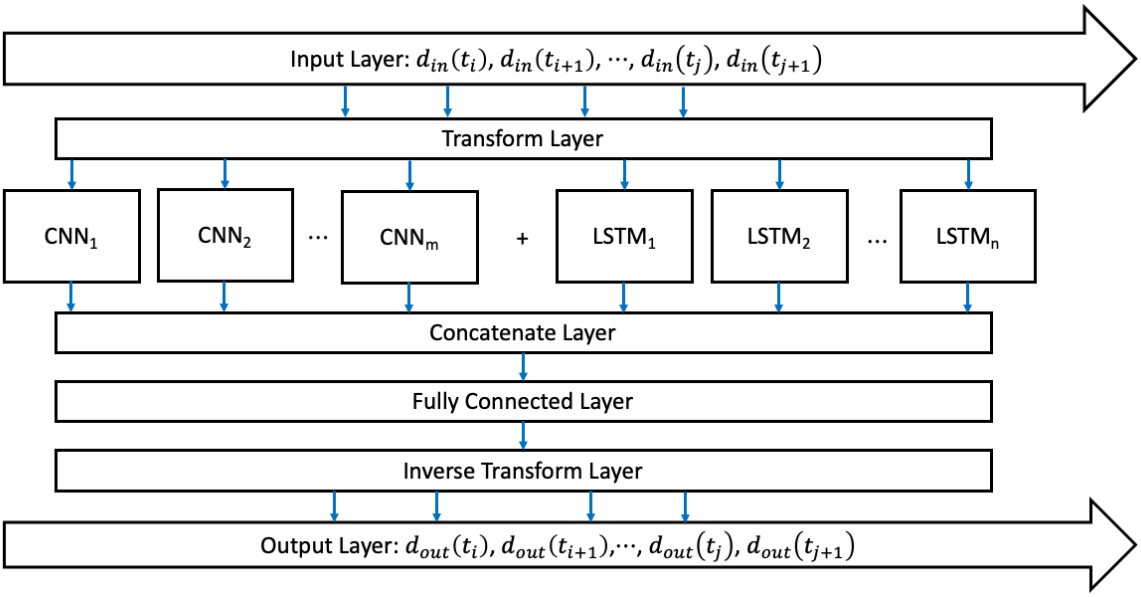

FIG. 2. The structure of a parallel CNN-LSTM network.

3.3. NUBEAMnet-3 (CNN-LSTM-based Surrogate Model)

When the NBI is activated, NUBEAM computes the distribution function based on the current plasma conditions. These conditions, saved as Monte Carlo particles, are stored after each NUBEAM time step. Next, the TRANSP code updates the plasma profiles and establishes a new equilibrium. In the following time step, NUBEAM uses the stored particles to update the distribution function according to the updated plasma state. This process ensures that the distribution function reflects the cumulative history of both the NBI and the discharge. Therefore, for an accurate approximation of the NUBEAM results, consideration of the temporal evolution of plasma variables, including parameters like the electron temperature and electron density, is crucial when designing the input set for the surrogate model.

NUBEAMnet-3 has been developed by following a novel deep-learning approach based on the combination of LSTM networks and CNNs. This hybrid approach is followed to take care at the same time of both the beam slowing-down time effects and the 2D data structures arising from profile data. The proposed NN topology is novel because the convolution and pooling operations are only applied to profile data while the LSTM layers extract the sequential information from beam power injection, which forms a parallel CNN-LSTM. The topology of the network is shown in Fig. 2. The initial layer transforms the multivariate time-series data from NUBEAM, rendering it suitable for model training. After this preprocessing, the data is divided and undergoes parallel training within the CNN and LSTM frameworks (the CNN parameters are consistent with those specified in Section 3.2).

3.3.1. Long Short-Term Memory Network

LSTM networks, a specialized subclass of recurrent neural networks (RNN), have gained prominence for their aptitude in modeling complex temporal sequences. The LSTM unit encompasses a cell state and three regulatory gates: input, output, and forget gates, formulated to mitigate the shortcomings of gradient vanishing and exploding, prevalent in traditional recurrent architectures. The mathematical formulation for an LSTM unit is as follows:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f), \tag{4}$$
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i), \tag{5}$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C), \tag{6}$$
$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t, \tag{7}$$
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o), \tag{8}$$
$$h_t = o_t \odot \tanh(C_t), \tag{9}$$

where f_t, i_t, $\tilde{C}_t$, C_t, and o_t are the forget gate, input gate, candidate cell state, cell state, and output gate at the t-th time step, respectively. In this formulation, W and b signify weights and biases, σ is the sigmoid activation function, h and x are the hidden state and input state, and $\odot$ symbolizes element-wise multiplication. The structural

TABLE 2. Summary of the average R^2 correlation coefficients between the NUBEAM and NUBEAMnet predictions from the training and testing datasets.

	NUBEAMnet-1		NUBEAMnet-2		NUBEAMnet-3	
	training	testing	training	testing	training	testing
P_{shine}	0.991	0.997	0.992	0.994	0.998	0.983
P_{cx}	0.985	0.991	0.985	0.971	0.961	0.956
P_{orb}	0.855	0.839	0.915	0.891	0.981	0.944
$P_{b,e}$	0.982	0.975	0.984	0.971	0.986	0.987
$P_{b,i}$	0.981	0.961	0.972	0.975	0.969	0.979
$T_{b,e}$	0.955	0.945	0.975	0.959	0.993	0.988
$T_{b,i}$	0.832	0.811	0.932	0.909	0.951	0.939
n_b	0.985	0.982	0.975	0.979	0.963	0.961
P_{fast}	0.991	0.953	0.982	0.962	0.995	0.971
j^{dep}_{NBI1-4}	0.985	0.962	0.982	0.989	0.997	0.977

design of LSTM networks makes them particularly effective in capturing long-range dependencies in temporal data, which is indispensable for time-series analysis in various scientific and engineering disciplines.

In the LSTM configuration, the ReLU is used as the activation function. Specifically, the primary LSTM layer consists of 256 hidden nodes on average, while the subsequent layer is equipped with 64 hidden nodes, tailored for temporal feature extraction. To mitigate the risk of overfitting, a dropout strategy is employed with a rate of 0.1 in the LSTM layers. A final concatenation layer serves to integrate the disparate features derived from the preceding layers, yielding a comprehensive feature set.

3.3.2. Handling Unequal Sequence Lengths in LSTM Models

To enrich the dataset for LSTM-CNN training, each time-series entry is divided into multiple shorter time-series segments. Each segment spans a minimum duration of three seconds to ensure the preservation of critical temporal information. Although some segments may share data points, no two segments are identical. After segmentation, the dataset undergoes a random shuffling to mitigate potential order-induced biases, thus enhancing the model's training efficacy.

A fundamental challenge arises when dealing with multiple time series sequences of varying lengths. Traditional approaches often resort to padding, a technique where shorter sequences are artificially extended to match the length of the longest sequence in the dataset by appending zero-value or neutral elements. While effective, padding introduces computational overhead and may distort the temporal dynamics that LSTMs are designed to capture. To overcome these limitations, we adopt the 'packing' methodology, which is a more computationally efficient alternative offered by ML libraries like PyTorch. Packing allows LSTMs to process sequences of varying lengths in a batch without performing unnecessary computations on padded elements, thereby preserving the genuine temporal relationships within each series. This method is particularly advantageous for our dataset, given its intrinsically varying sequence lengths due to different simulation durations.

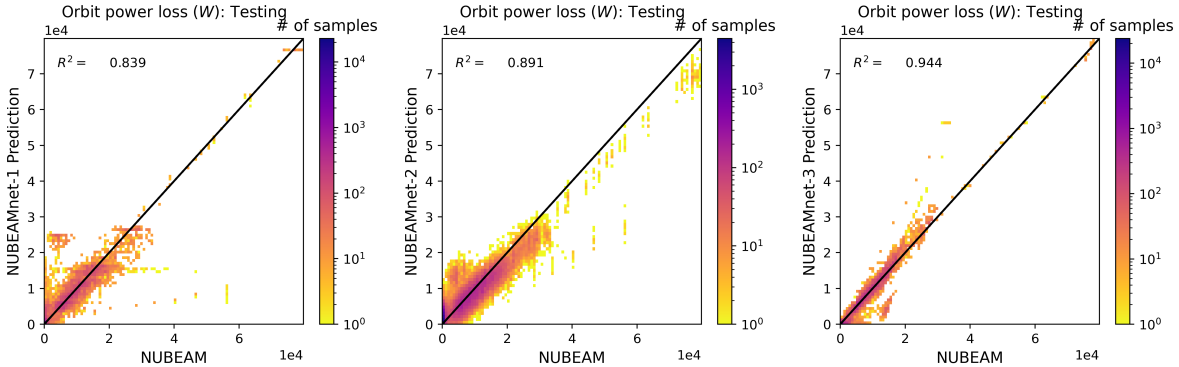

FIG. 3. Histograms of the regression of the orbit power loss in the testing dataset for NUBEAMnet-1 (left), NUBEAMnet-2 (center), and NUBEAMnet-3 (right).

4. SIMULATION RESULTS

In this section, an in-depth evaluation of the proposed NUBEAMnet models is conducted through a large number of simulations. The simulation outcomes highlight the respective strengths and weaknesses of the three NN-based surrogate models under investigation. While the MLP-based surrogate model (NUBEAMnet-1) demonstrates expedient computational speeds, particularly in a Python environment (averaging 0.1 ms per time-step for scalar predictions and approximately 1 ms for profile data), it falls short in terms of predictive accuracy (especially for profile data) in comparison to its more complex counterparts, NUBEAMnet-2 and NUBEAMnet-3. The longer computational times for NUBEAMnet-2 and NUBEAMnet-3 are comparable, each requiring roughly 5 ms per time-step within a Python environment because of their utilization of a transform layer.

Results obtained from the testing dataset show that NUBEAMnet-2 and NUBEAMnet-3 have superior prediction capabilities in comparison to NUBEAMnet-1. This is revealed in Table 2 where the average R^2 correlation coefficients between the predictions made by the NUBEAM module and the corresponding NUBEAMnet models are summarized. In particular, Fig. 3 shows that NUBEAMnet-3 significantly outperforms NUBEAMnet-1 in predicting the orbit power loss. Fig. 4 illustrates the predictive accuracy of the three NN-based surrogate models in regard to the following profiles: beam current drive, beam heating to ions, beam heating to electrons, beam torque

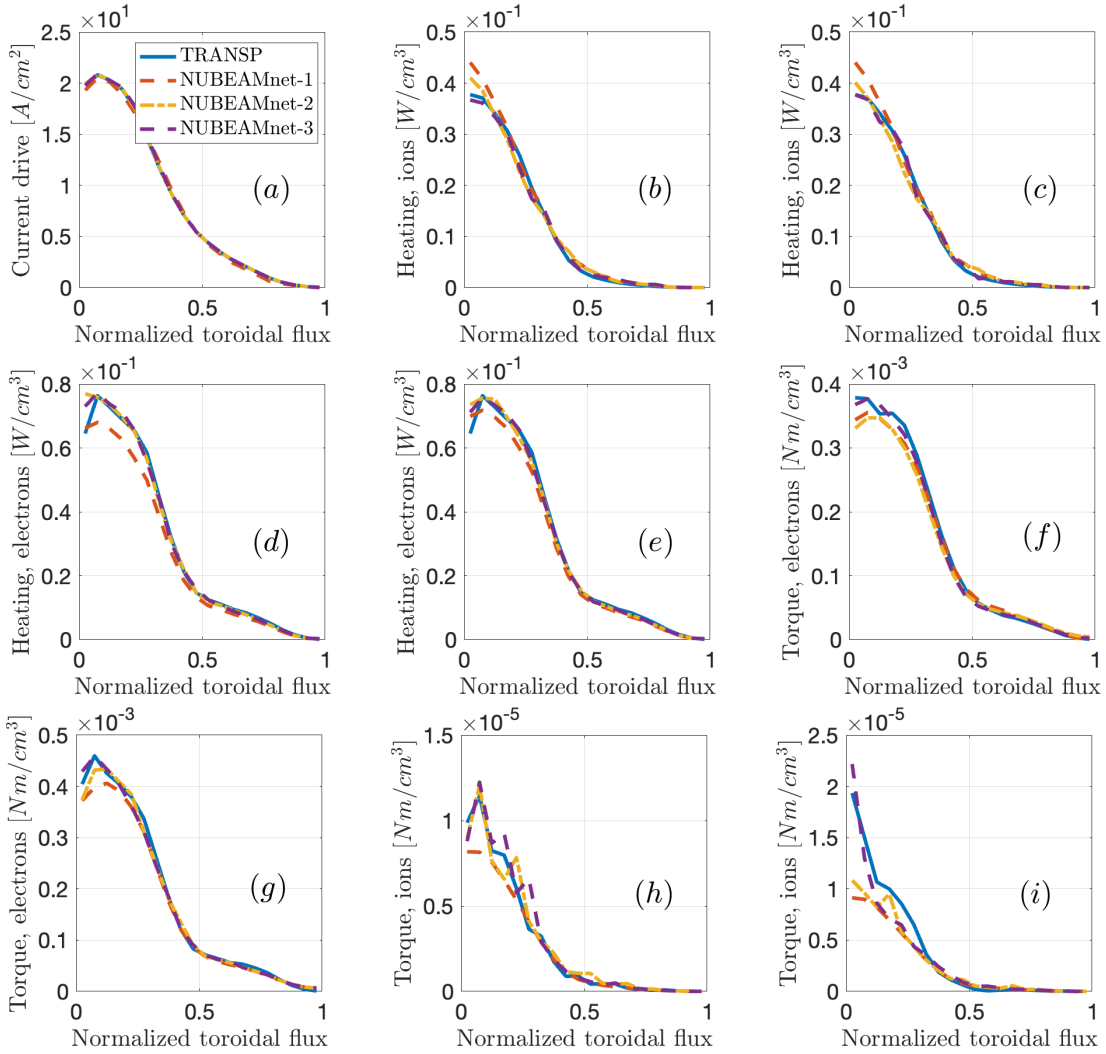

FIG. 4. Profiles predicted by NUBEAM (solid blue line) and the three different versions of NUBEAMnet for EAST: the MLP-based surrogate model (dashed red line), the CNN-based surrogate model (dashed orange line), and the CNN-LSTM-based surrogate model (dot-dashed purple line). The plotted profiles include (a) the current drive at $t = 4$ s, (b) the heating to the electrons at $t = 4$ s, (c) the heating to the electrons at $t = 5$ s, (d) the heating to the ions at $t = 4$ s, (e) the heating to the ions at $t = 5$ s, (f) the torque applied to the electrons at $t = 4$ s, (g) the torque applied to the electrons at $t = 5$ s, (h) the torque applied to the ions at $t = 4$ s, and (i) the torque applied to the ions at $t = 5$ s.

to electrons, and beam torque to ions. While all three trained neural networks exhibit high predictive accuracy in relation to beam current drive, Fig. 4 indicates that both NUBEAMnet-2 and NUBEAMnet-3 offer improved predictive accuracy. Because of the higher data values typically observed in the core region, the prediction accuracy generally improves as the normalized toroidal flux approaches zero.

5. CONCLUSIONS AND FUTURE WORK

This work offers a methodological contribution by introducing neural-network-based surrogate models for the NUBEAM module with a particular focus on the updates made to the NBI system in EAST in 2020. These surrogate models enable expedient transport simulations and present varied strengths tailored for different applications. Specifically, NUBEAMnet-1, although less precise in profile prediction, operates with remarkable execution speed, which is more favorable for rapid between-shot transport analysis. Meanwhile, NUBEAMnet-2 and NUBEAMnet-3 exhibit more fidelity in profile prediction, making them more suitable for applications that tolerate longer execution times and benefit from a higher prediction accuracy such as off-line scenario planning by model-based optimization. In addition, the NUBEAMnet models could be used for model-based control synthesis.

Future work will focus on several key aspects to improve the network architectures. First, hyperparameter optimization techniques will be explored to enhance the predictive accuracy and computational efficiency of the networks. Second, an analysis will be conducted to identify the critical time-sensitive variables for the LSTM components, which could lead to more efficient and accurate time-series predictions. Lastly, these optimized networks may be integrated into the COTSIM (Control-Oriented Transport SIMulator) code, which is a rapid 1D transport solver, with the aim to facilitate real-time control applications and between-shot scenario planning.

ACKNOWLEDGEMENTS

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] SMIRNOV, A. HARVEY, R., The GENRAY ray tracing code, CompX Report CompX-2000-01 (2001).

[2] LIN-LIU, Y., CHAN, V., PRATER, R., Electron cyclotron current drive efficiency in general tokamak geometry, Physics of Plasmas 10 (2003) 4064.

[3] PANKIN, A., MCCUNE, D., ANDRE, R., BATEMAN, G., KRITZ, A., The tokamak Monte Carlo fast ion module NUBEAM in the National Transport Code Collaboration library, Computer Physics Communications 159 (2004) 157.

[4] MONTES, K. J., REA, C., GRANETZ, R., et al., Machine learning for disruption warnings on Alcator C-Mod, DIII-D, and EAST, Nuclear Fusion 59 (2019) 096015.

[5] MOHAPATRA, D., SUBUDHI, B., DANIEL, R., Real-time sensor fault detection in tokamak using different machine learning algorithms, Fusion Engineering and Design 151 (2020) 111401.

[6] JOUNG, S., KIM, J., KWAK, S., et al., Deep neural network Grad-Shafranov solver constrained with measured magnetic signals, Nuclear Fusion 60 (2019) 016034.

[7] DEGRAVE, J., FELICI, F., BUCHLI, J., et al., Magnetic control of tokamak plasmas through deep reinforcement learning, Nature 602 (2022) 414.

[8] WALLACE, G., BAI, Z., SADRE, R., et al., Towards fast and accurate predictions of radio frequency power deposition and current profile via data-driven modelling: applications to lower hybrid current drive, Journal of Plasma Physics 88 (2022) 895880401.

[9] BOYER, M., KAYE, S., ERICKSON, K., Real-time capable modeling of neutral beam injection on NSTX-U using neural networks, Nuclear Fusion 59 (2019) 056008.

[10] MOROSOHK, S. M., BOYER, M. D., SCHUSTER, E., Accelerated version of nubeam capabilities in DIII-D using neural networks, Fusion Engineering and Design 163 (2021) 112125.

[11] WANG, Z., MOROSOHK, S., RAFIQ, T., et al., Neural network model of neutral beam injection in the EAST tokamak to enable fast transport simulations, Fusion Engineering and Design 191 (2023) 113514.

[12] ZHENG, Y., XIAO, J., HAO, B., et al., Modeling of beam ions loss and slowing down with Coulomb collisions in EAST, Chinese Physics B (2022).

[13] WANG, J., CHEN, Y., WU, B., et al., Injection performance prediction of the upgraded neutral beam on EAST, Fusion Engineering and Design 166 (2021) 112277.