Nucl. Fusion 61 (2021) 106040 (10pp) https://doi.org/10.1088/1741-4326/ac207e

Neural network model of the multi-mode anomalous transport module for accelerated transport simulations

S.M. Morosohk*, A. Pajares, T. Rafiq and E. Schuster

Lehigh University, Bethlehem, Pennsylvania 18015-3085, United States of America

E-mail: morosohk@lehigh.edu

Received 28 May 2021, revised 26 July 2021
Accepted for publication 24 August 2021
Published 8 October 2021

Abstract

A neural network version of the multi-mode anomalous transport module, known as MMMnet, has been developed to calculate plasma turbulent diffusivities in DIII-D with a calculation time suitable for control applications. MMMnet uses a simple artificial neural network structure to predict the ion thermal, electron thermal, and toroidal momentum diffusivities while reproducing Multi-Mode Model (MMM) data with good accuracy and keeping the calculation time as a fraction of that associated with MMM. Model-based control techniques require models with fast calculation times, making many existing physics-oriented predictive codes unsuitable. The control-oriented predictive code Control Oriented Transport Simulator (COTSIM) calculates the most significant plasma dynamics in response to the different actuators while running at a speed useful for control design. In order to achieve this calculation speed, COTSIM often relies on scaling laws and control-level models. Replacing some of these scaling laws and control-level models with neural network versions of more complex physics-level models has the potential of increasing the range of validity and the level of accuracy of COTSIM without compromising its computational speed. In this work, MMMnet is integrated into COTSIM to improve the turbulent diffusivity predictions, which will in turn improve the prediction accuracy associated with the dynamics of many plasma properties.

Keywords: neural network, Multi-Mode Model, Control Oriented Transport Simulator, control, nuclear fusion

(Some figures may appear in colour only in the online journal)

1. Introduction

In order for tokamak plasmas to achieve both stable operation and high performance, the 1D spatial distributions of plasma parameters such as density, temperature, current, and momentum must be carefully controlled. The evolution of these profiles in time is described by a system of nonlinear transport equations that are too complicated to be modeled from first principles with reasonable calculation times using presently available real-time hardware systems. Reduced physics-based models are available, but they are still too computationally intensive to be well suited for control applications.

Model-based control applications require response models with fast (on the order of seconds, e.g. for closed-loop offline simulations) to extremely fast (on the order of milliseconds, e.g. for real-time control and estimation) calculation times, making well-established physics-oriented predictive transport codes challenging or impossible to use. Instead, control-oriented models with significantly faster calculation times must be developed; however, this speed often comes with a cost. Reducing physics models to the point that they run at the required speeds can also reduce accuracy beyond the point where the model is useful. Empirical scaling laws can achieve high levels of accuracy, but may only be valid for specific plasma scenarios. Machine learning models have the uncommon ability to meet all three goals: high levels of

* Author to whom any correspondence should be addressed.

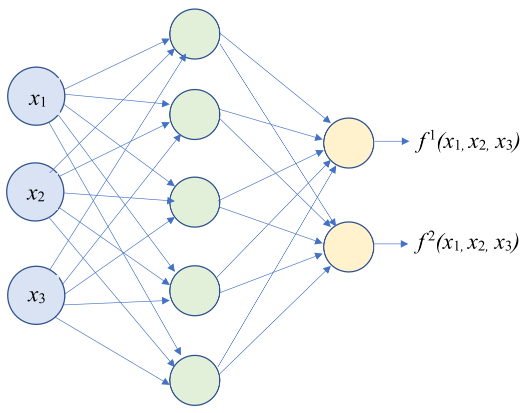


Figure 1. Structure of a multi-layer perceptron neural network [8].

accuracy, fast calculation times, and applicability to a broader range of plasma scenarios. Recently, neural-network versions of a number of physics-based codes for both transport coefficients [1, 2] and current, particle, and heat sources [3, 4] have been developed, which significantly reduce the calculation time required while maintaining relatively high prediction accuracies across many different scenarios. Inspired by these recent developments on machine-learning-based plasma-response modeling, a neural network that reproduces the predictions of the Multi-Mode Model (MMM) [5] for anomalous transport has been developed for DIII-D. This neural network is known as MMMnet.

The MMMnet model has been trained to reproduce the calculation of the ion thermal, electron thermal, and toroidal momentum turbulent diffusivities. A simple artificial neural network structure (see figure 1) was used with three hidden layers and 100 nodes per hidden layer. Five separate networks were trained with different initializations of the weights; the final prediction returned is the average of the five predictions. A low standard deviation between the predictions indicates that the training effectively eliminates any randomness introduced by the initial weights. Training data was generated by calling 1000 predictive TRANSP [6, 7] runs based on 83 different DIII-D shots and using MMM as the transport model. Instead of relying on a convolutional neural network to handle spatially-varying data, principle component analysis was applied to the plasma profiles to reduce the amount of data used to describe each profile. This reduced the number of input and output nodes in the network, thereby limiting the network complexity and calculation time.

The Control Oriented Transport Simulator (COTSIM) is a 1D transport code designed to run at speeds useful for control applications. It uses either a prescribed MHD equilibrium or a fixed-boundary analytical solver, although the coupling with a free-boundary numerical solver is currently in progress. It uses a modular configuration which allows the user to easily choose from transport and source models of varying complexities, ranging from empirical scalings to reduced analytical models to machine learning models. A neural network version of NUBEAM [4] is already available to calculate beam driven current, heating, and torque. Depending on the models chosen, COTSIM takes between a fraction of a second and several seconds to simulate a full DIII-D discharge. In this work, the neural network version of MMM is integrated into COTSIM to enhance its fast prediction capabilities.

This paper is organized as follows. In section 2, the process used to obtain the dataset needed to train and evaluate the neural network model is described. In section 3, the method used to process the data and determine the topology and training parameters of the model is explained. In section 4, the results of the final neural network model are presented. In section 5, the neural network is integrated into COTSIM and predictive-simulation results are shown. In section 6, conclusions and plans for future work are discussed.

2. Dataset development

The input data used in this work was taken from a set of 1000 predictive TRANSP runs using MMM7.1 as the transport model and evolving the electron and ion temperature profiles. Using TRANSP helps to ensure that the inputs to MMM are in a physically reasonable range for DIII-D Deuterium plasmas. The TRANSP runs are based on 83 different shots from the 2018 DIII-D campaign, including both L-mode and H-mode plasmas. For each TRANSP run, using a random number generator, the mean effective charge (Z_{eff}) was assigned a value between 1.5 and 5, the edge neutral density ($n_{0,\text{out}}$) was assigned a value between 5×10^{10} and $1 \times 10^{13.5}$ cm^{-3}, and the anomalous fast ion diffusivity (D_f) was assigned either a classical, flat, or peaked shape and a maximum value between 1 and 50,000 cm^2 s^{-1}. Changing the values of Z_{eff} directly affects the calculation of the diffusivities by MMM. Changing $n_{0,\text{out}}$ and D_f changes the calculation of the effects of neutral beam injection by NUBEAM, which changes the temperature inputs to MMM and thus indirectly alters the calculated diffusivities. By this process, the original 83 shots are expanded to 1000 TRANSP runs, or 201,612 total time steps in the dataset.

The total dataset is then divided into three subsets used for training, validation, and testing. One of the main concerns when training a neural network is that the network could be learning the exact training data more than the underlying function which describes the data; this issue is referred to as overfitting. If this is the case, for any point that was not included in the original training set, the network would be unable to generalize and would produce less useful predictions. In order to check for this, a portion of the data is held back during training. The training set, made up of 80% of the TRANSP runs in the total dataset, is what the neural network actually sees during the training process. Another 10% of the TRANSP runs go into the validation set, which is used during the parameter tuning process to determine the values of the manually assigned network parameters, or hyperparameters (see section 3.2). The remaining 10% of the TRANSP runs make up the testing set, and are used to assess the performance of the final model.

2.1. Model inputs and outputs

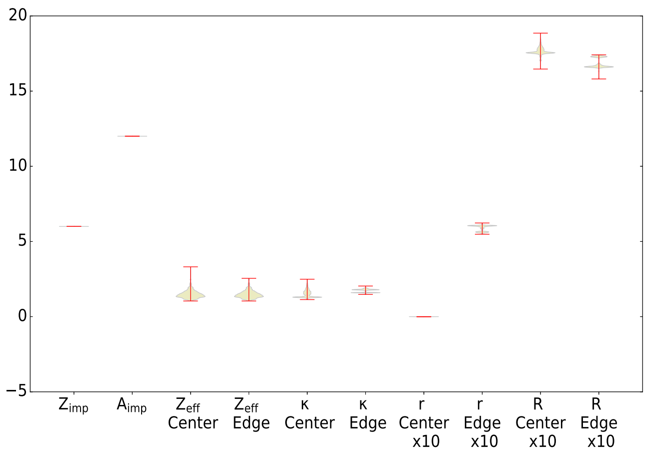
The inputs used in this network are most of the inputs used by the standalone version of MMM. The parallel velocity was assumed to be equal to the toroidal velocity, and the mean atomic mass of ions was derived from the average charge and the atomic mass of impurities. The inputs listed in table 1 provide enough information for the neural network to learn the diffusivity calculations. For some of the input quantities, the normalized gradient is also used and is treated as a separate input, as also indicated in table 1. Figure 2 shows the range of each input in the training dataset. Because machine learning models cannot extrapolate, this gives a lower-dimensional representation of the range for which the neural network predictions will be valid. The outputs of the neural network are listed in table 2. The version of MMM used in this work is not valid in the pedestal region of H-mode plasmas, so the values of the diffusivity outputs have only been calculated in these TRANSP runs in the spatial region $0 \leqslant \hat{\rho} \leqslant 0.8$. MMMnet is only trained to produce valid predictions in that region.

3. Model structure

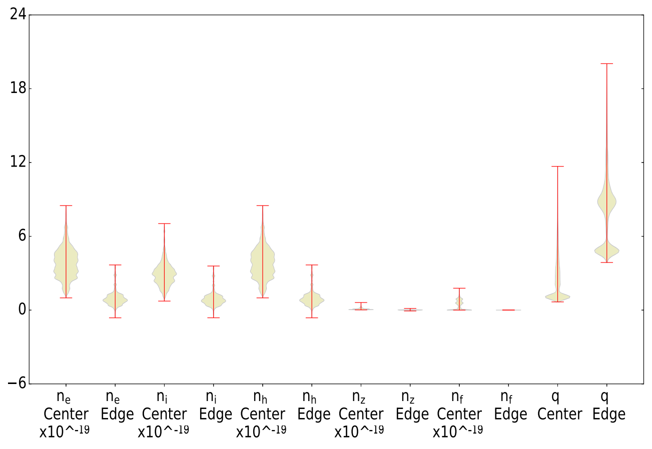
3.1. Data preprocessing

When predicting spatially-varying data, a typical approach is to use a convolutional neural network (CNN). This technique is well-suited to datasets with two or three spatial dimensions and is often used in image recognition and computer vision applications. However, it is a more complicated architecture than the multi-layer perceptron (MLP), and can therefore have slightly higher calculation times. Given the goal of developing a network with as fast a prediction time as possible, it was decided to use an MLP in this work. In order to use this simpler approach, each profile must be reduced to a set of scalar points. This can be accomplished using a principle component analysis (PCA), which projects each profile, including gradients, onto a set of basis functions and determines how much of the total variance in the data each basis function accounts for. In this work, the PCA for each profile and gradient includes every basis function that explains at least 0.1% of the variance in the data. This results in the retention of at least 99.5% of the total variance of each profile. The number of basis functions retained for each profile, and its gradient if the gradient is included, is seen in tables 1 and 2. It is important to point out that the PCA for each profile is derived from the training dataset, and is therefore only valid for profiles in the same range as those in the training data. Just as the neural network cannot extrapolate to regions far from any training data point, the PCA cannot accurately deconstruct and reconstruct a profile that is significantly different from any profile in the training data.

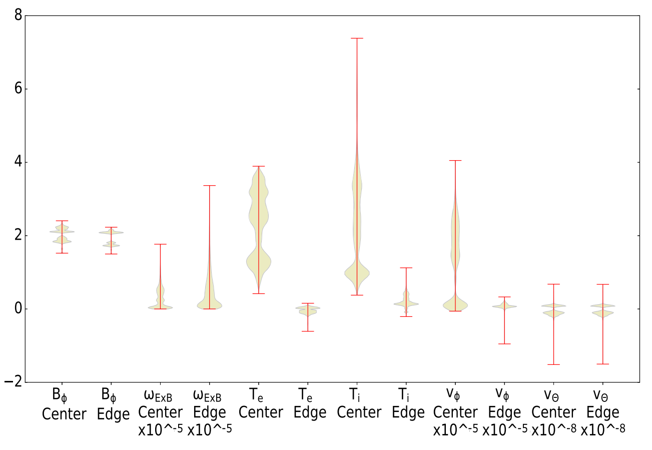
Before the data can be fed to the neural network, each input and output is standardized to a mean of 0 and a standard deviation of 1. The normalized value is calculated as $z = (x - \mu)/s$ where x is the absolute value, μ is the average and s is the standard deviation of the training dataset. This ensures that all inputs are given equal weight in the prediction, that inputs such as densities with a higher absolute value do not overpower the other inputs, and that all outputs are given equal weight when calculating the loss function. For each prediction the neural network makes, the profile data is reduced to a set of scalars through the PCA, all the inputs are standardized, the neural network makes its prediction, the network outputs go through the reverse of the standardization process, and the final profiles are reconstructed by the PCA.



(a)



(b)



(c)

Figure 2. Range of inputs in the training dataset. x-axes show quantities, y-axes reflect the range of values of each quantity, and the width of the green space represents the frequency that that value is seen in the training dataset.

Table 1. Inputs to the neural network model.

Inputs			
Symbol	Name	Units	PCA Modes
Z_{imp}	Average charge of impurities		
A_{imp}	Average atomic mass of impurities	amu	
Z_{eff}	Mean effective charge		1
κ	Elongation		4
R	Major radius	m	4
r	Minor radius	m	4
B_{t}	Toroidal magnetic field	T	2
n_{e}	Electron density	m^{-3}	5
$g_{n_{\text{e}}}$	Normalized electron density gradient		2
n_{i}	Ion density	m^{-3}	6
$g_{n_{\text{i}}}$	Normalized ion density gradient		2
n_{h}	Hydrogenic ion density	m^{-3}	5
$g_{n_{\text{h}}}$	Normalized hydrogenic ion density gradient		2
n_z	Impurity density	m^{-3}	5
g_{n_z}	Normalized impurity density gradient		2
n_{f}	Fast ion density	m^{-3}	2
T_{e}	Electron temperature	keV	4
$g_{T_{\text{e}}}$	Normalized electron temperature gradient		2
T_{i}	Ion temperature	keV	4
$g_{T_{\text{i}}}$	Normalized ion temperature gradient		2
q	Safety factor		5
g_q	Normalized safety factor gradient		7
$\omega_{\text{E}\times\text{B}}$	$E \times B$ shear	rad s^{-1}	8
v_ϕ	Toroidal velocity	m s^{-1}	4
g_{v_ϕ}	Normalized toroidal velocity gradient		7
v_θ	Poloidal velocity	cm s^{-1}	1
g_{v_θ}	Normalized poloidal velocity gradient		6

Table 2. Outputs of the neural network model.

Outputs			
Symbol	Name	Units	PCA Modes Kept
χ_{i}	Ion thermal diffusivity	$\text{m}^2\ \text{s}^{-1}$	15
χ_{e}	Electron thermal diffusivity	$\text{m}^2\ \text{s}^{-1}$	14
χ_ϕ	Toroidal momentum diffusivity	$\text{m}^2\ \text{s}^{-1}$	12

3.2. Determination of hyperparameters

The term hyperparameters refers to all of the network parameters which are not learned during the training process. These include the structure of the network and length of the training process, among others. It is up to the person training the neural network to choose values of these parameters that yield good results. Methods of determining these parameters include genetic-algorithm optimization [9] and Bayesian optimization [10]. However, the most common approach is still to conduct a grid search, or a brute force testing of different hyperparameter values. In this work, network architectures ranging from 1 to 4 hidden layers and 50 to 200 neurons per hidden layer were tested, and the validation accuracy and calculation times for each model are shown in figure 3. Subplots 3(*a*) and 3(*b*) show the model accuracy and time per prediction, respectively, plotted against the model architecture. Subplots 3(*c*) and 3(*d*) show the model accuracy and time per prediction, respectively, plotted against the number of learned parameters in the model. Model accuracy in subplots 3(*a*) and 3(*c*) is reported as the correlation between the neural network prediction and MMM-predicted data for each point in the validation dataset, measured as the R^2 value of a linear regression. Note that the calculation times shown in subplots 3(*b*) and (*d*) came from Python and include the compilation time. They are shown here as a comparison between different model architectures, but should not be taken as the true calculation time for the model. In order to balance the two simultaneous goals of high accuracy and low calculation time, the architecture of 3 hidden layers with 100 neurons per layer was chosen. This architecture displays essentially the same level of accuracy as more complicated architectures, implying that there is enough flexibility for this model to learn the underlying function well. In addition, the calculation time is relatively short compared

Table 3. Hyperparameters determined by results of parameter tuning.

Number of hidden layers	3
Nodes per hidden layer	100
Number of epochs	16
Batch size	9

Table 4. Other inputs to the Keras model.

Solver	'sgd'
Hidden layer activation function	'relu'
Output layer activation function	'linear'
Loss	'mse'
Metrics	'accuracy'

to architectures which display similar levels of accuracy. The length of the training process was determined in a similar way. The length of the training process is measured in epochs, or number of times that the network sees each data point in the training set during the training process. If training is allowed to run for too long, the network will begin to memorize (overfit) the exact training data instead of learning the underlying function, and the network predictions will not be as accurate on any data point not in the training set. Indications of overfitting can be seen in figure 4 as the difference in R^2 between the training and validation sets increases with the number of epochs. In order to both maximize validation accuracy and minimize overfitting, training of the final model was limited to 16 epochs. The batch size, or the number of data points seen by the network in between each update of the weights, was chosen to maximize R^2 on the validation data. The final model was trained with a batch size of 9. The hyperparameters chosen through the parameter tuning process are listed in table 3. All other parameters of the Keras [11] model are listed in table 4.

One source of uncertainty in the model stems from the initial randomization of the weights. There is a possibility that, during the training process, the weights are converging to a local minimum of the loss function instead of the global minimum. In order to account for this, five separate models were trained in parallel using the same hyperparameters and the same training data, but different random initial weights. When all five parallel networks converge to the same minimum of the loss function, that is most likely the global minimum. The final network reports the average, standard deviation, minimum, and maximum of the five predictions. If the network is being used for an application that requires even faster calculation times, the user can choose to not use all five parallel networks or to use parallel computing.

4. Model evaluation

Results of the final neural network model are shown in this section. The correlations between MMM-predicted and MMMnet-predicted data for each output on the training and testing datasets are seen in table 5. Predictions are shown for TRANSP run 176052T05, which is in the testing dataset.

Table 5. Correlations (R^2) between MMM and MMMnet predictions for shots in training and testing datasets.

	R^2 values: training data	R^2 values: testing data
χ_i	0.961	0.883
χ_e	0.941	0.843
χ_ϕ	0.928	0.878

Figure 5 shows the evolution of the MMM and MMMnet predictions at the spatial location $\hat{\rho} = 0.612$ over the course of TRANSP run 176052T05 for (*a*) χ_i, (*b*) χ_e, and (*c*) χ_ϕ. Figure 6 shows the MMM and MMMnet profiles at time $t = 1.82$ s for (*a*) χ_i, (*b*) χ_e, and (*c*) χ_ϕ, and figure 7 shows the same results at time $t = 2.32$ s, both for the same TRANSP run. The red lines show the MMM-predicted data, the dark blue lines show the average of the five MMMnet predictions, and the blue shaded areas show one standard deviation above and below the average prediction. Figure 5 shows that the neural network is able to follow changes in the diffusivities over time. Figures 6 and 7 show that the neural network is able to reconstruct a variety of different profile shapes, even using PCA instead of a CNN. Note that the plots in figures 6 and 7 do not extend all the way to the plasma edge, but cover only the section of the profile that MMMnet predictions are valid for.

In the Cython [12] programming language executed on a non-real-time computer, the model takes an average of 1.35 ms to make a prediction per time step. The calculation time has not yet been tested on the DIII-D plasma control system computer, but is expected to be faster than the Cython value. This would make the network well-suited for real-time control.

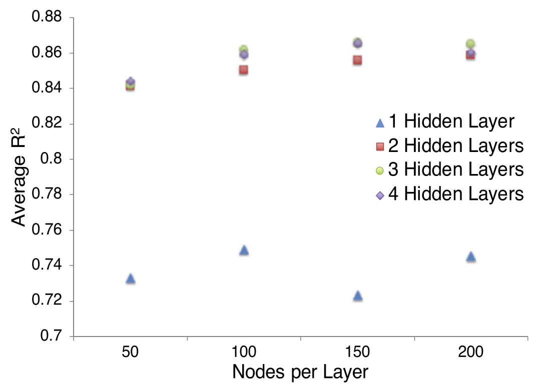
5. Integration into COTSIM

5.1. Electron heat transport modeling

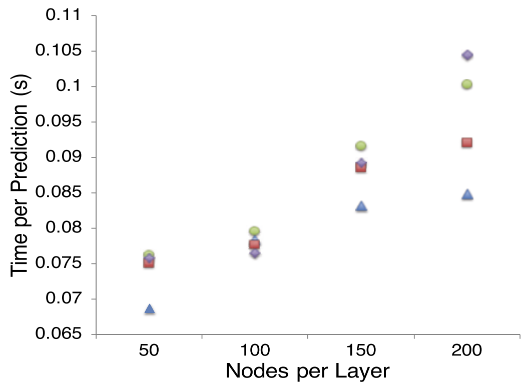
COTSIM has the capability to calculate the electron temperature profile as the numerical solution to the electron heat transport equation [13],

$$\frac{3}{2}\frac{\partial}{\partial t}[n_e T_e] = \frac{1}{\rho_b^2 \hat{H}}\frac{1}{\hat{\rho}}\frac{\partial}{\partial \hat{\rho}}\left[\hat{\rho}\frac{\hat{G}\hat{H}^2}{\hat{F}}\left(\chi_e n_e \frac{\partial T_e}{\partial \hat{\rho}}\right)\right] + Q_e, \quad (1)$$

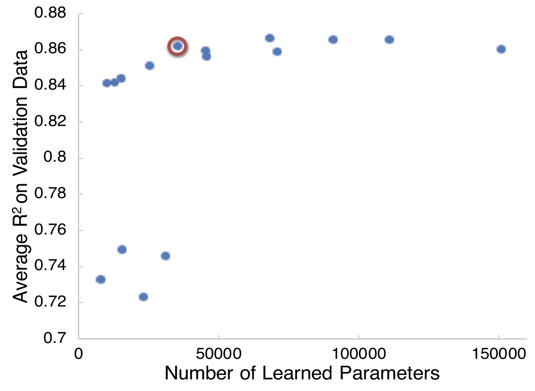
which depends on the electron thermal diffusivity χ_e, as well as electron density n_e, heat deposition Q_e, and the mean effective minor radius of the plasma boundary ρ_b. The mean effective minor radius is defined as $\rho \triangleq \sqrt{\Phi/(B_{\phi,0}\pi)}$, where $B_{\phi,0}$ is the vacuum toroidal magnetic field at the major radius, R_0, and Φ is the toroidal magnetic flux. A normalized version of ρ is defined as $\hat{\rho} \triangleq \rho/\rho_b$. The spatially varying geometrical factors $\hat{F}$, $\hat{G}$, and $\hat{H}$ are related to the magnetic configuration of a particular plasma equilibrium. The electron thermal diffusivity contains both neoclassical and anomalous components, but for the scenarios COTSIM is attempting to simulate in this work the neoclassical component is assumed to be negligible in comparison to the anomalous component, and is therefore neglected. COTSIM's model library includes multiple options to calculate the anomalous component of χ_e, including the



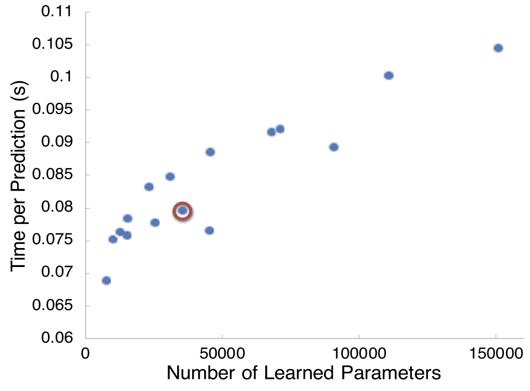
(a) Average R^2 as a function of architecture.



(b) Computation time as a function of architecture.



(c) Average R^2 as a function of parameters.



(d) Computation time as a function of parameters.

Figure 3. (*a*) Computation accuracies averaged across all outputs as a function of model architecture (number of layers and nodes per layer), (*b*) prediction times as a function of model architecture (number of layers and nodes per layer), (*c*) computation accuracies averaged across all outputs as a function of the number of learned parameters, and (*d*) prediction times as a function of the number of learned parameters. In plots (*c*) and (*d*), the circled values represent 3 hidden layers with 100 nodes per layer, which is the architecture chosen for the final network.

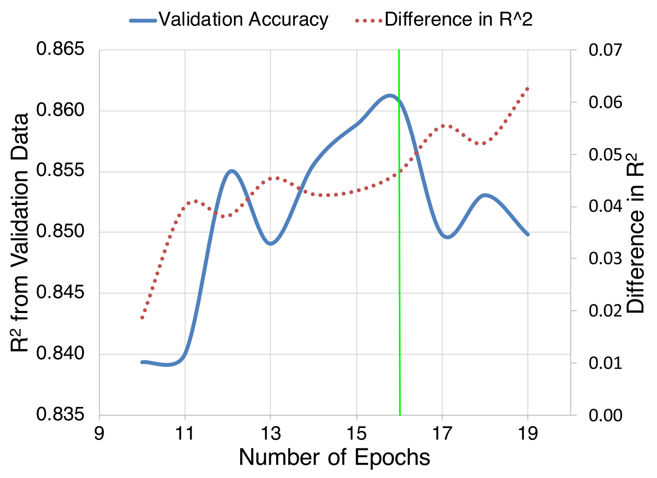


Figure 4. Prediction accuracy on the validation dataset and difference in accuracy between training and validation data vs number of epochs.

Bohm/gyro-Bohm model [14], the Coppi-Tang model [15], and now MMMnet. Most of the inputs to MMMnet are either already simulated in COTSIM or can be calculated from values already simulated in COTSIM; the few that cannot, such as the fast ion density, are assigned a prescribed value representative of the scenario of interest. A pedestal model [16] is used to calculate the edge of the electron temperature profile. A fixed pedestal width is chosen based on the pedestal width found in the simulation scenario, and the model is tuned to match the experimental pedestal height. An extrapolation is made for the MMMnet χ_e prediction from the spatial region where MMMnet is valid to the spatial region where the pedestal model is applied.

The results of two COTSIM simulations of shot 147634, one using MMMnet and the other using the Bohm/gyro-Bohm model to predict χ_e, are shown in figure 8. The COTSIM results are compared to TRANSP run 147634S01, which is an analysis run and therefore uses the experimental T_e profiles. Note that COTSIM is not expected to perfectly replicate experimental data, but to provide an approximate plasma response that is accurate enough for control purposes. The COTSIM simulations match each other exactly at the edge because they both use the pedestal model, but both the magnitude and the shape of the T_e profile in the core match the experimental profile much more closely when MMMnet is used.

The results of the same two COTSIM simulations of shot 147621 are shown in figure 9. The COTSIM results are compared to TRANSP analysis run 147621S01. There is a slight over-prediction of T_e in the core at $t = 2$ s, but overall the

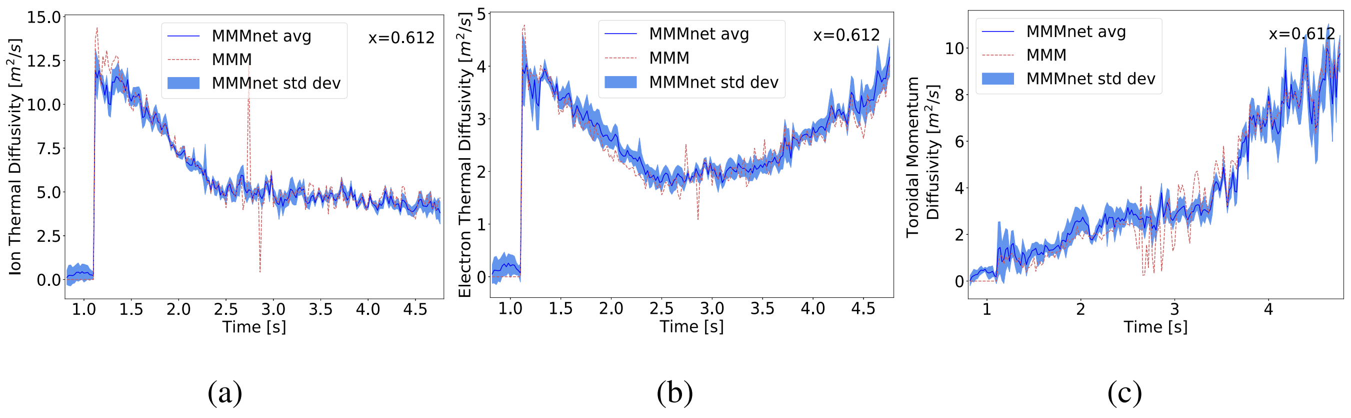


Figure 5. Evolution of a single point ($\hat{\rho} = 0.612$) in the profile over time for (*a*) χ_i, (*b*) χ_e, and (*c*) χ_ϕ for TRANSP run 176052T05. Five parallel neural network predictions are made, and the average and standard deviation are shown.

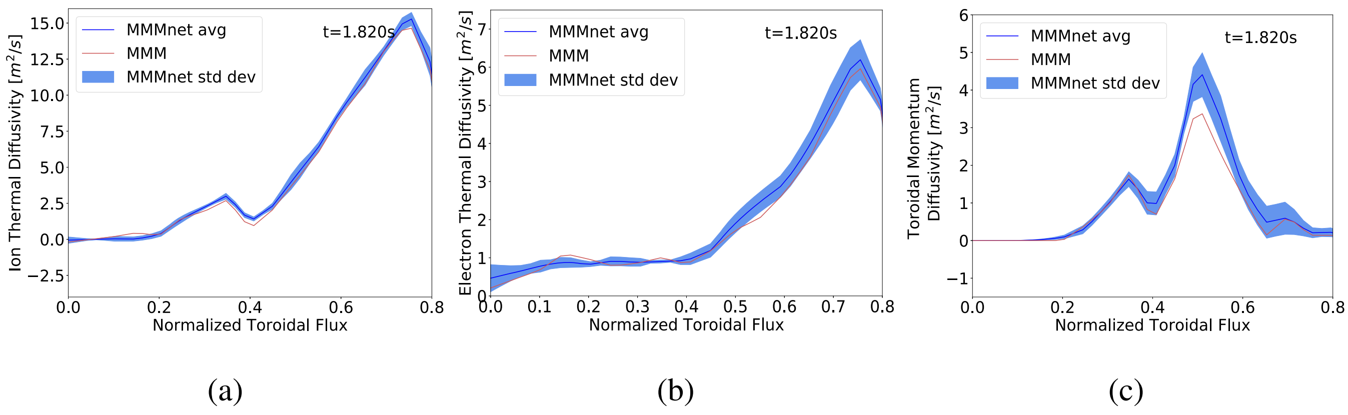


Figure 6. Prediction of the profile at time $t = 1.82$ s for (*a*) χ_i, (*b*) χ_e, and (*c*) χ_ϕ for TRANSP run 176052T05. Five parallel neural network predictions are made, and the average and standard deviation are shown up to a normalized toroidal flux of 0.8.

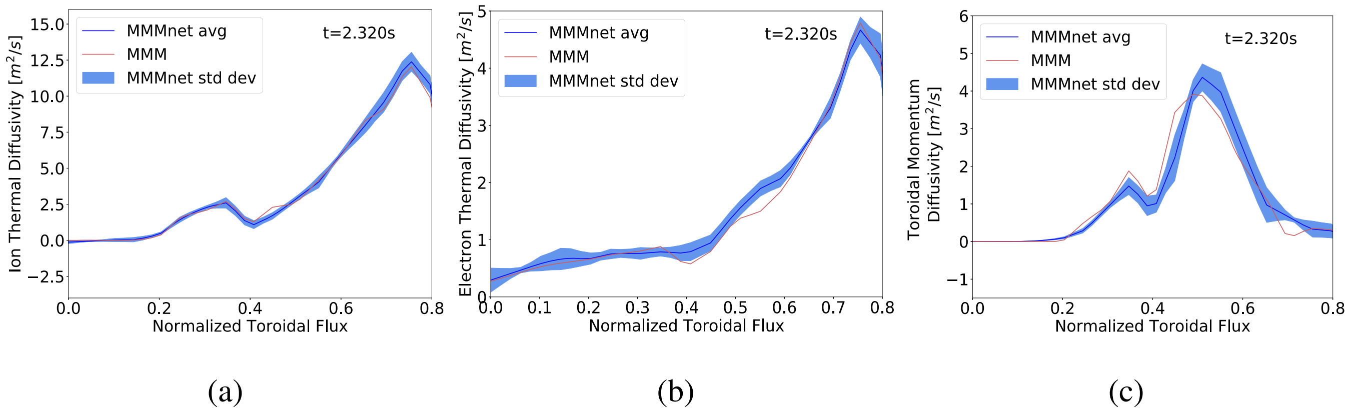


Figure 7. Prediction of the profile at time $t = 2.32$ s for (*a*) χ_i, (*b*) χ_e, and (*c*) χ_ϕ for TRANSP run 176052T05. Five parallel neural network predictions are made, and the average and standard deviation are shown up to a normalized toroidal flux of 0.8.

prediction of the T_e profile using MMMnet is closer to the experimental profile than when the Bohm/gyro-Bohm model is used.

5.2. Momentum transport modeling

COTSIM has the capability to calculate the rotation profile as the numerical solution to the toroidal angular momentum equation [17],

$$n_i m_i \langle R^2 \rangle \frac{\partial \Omega_\phi}{\partial t} + m_i \langle R^2 \rangle \Omega_\phi \frac{\partial n_i}{\partial t} = \tau_{nbi} + \frac{1}{\hat{\rho}\hat{H}} \frac{\partial}{\partial \hat{\rho}} \left[\hat{\rho}\hat{H} n_i m_i \chi_\phi \langle R^2 (\nabla \hat{\rho})^2 \rangle \frac{\partial \Omega_\phi}{\partial \hat{\rho}} \right], \quad (2)$$

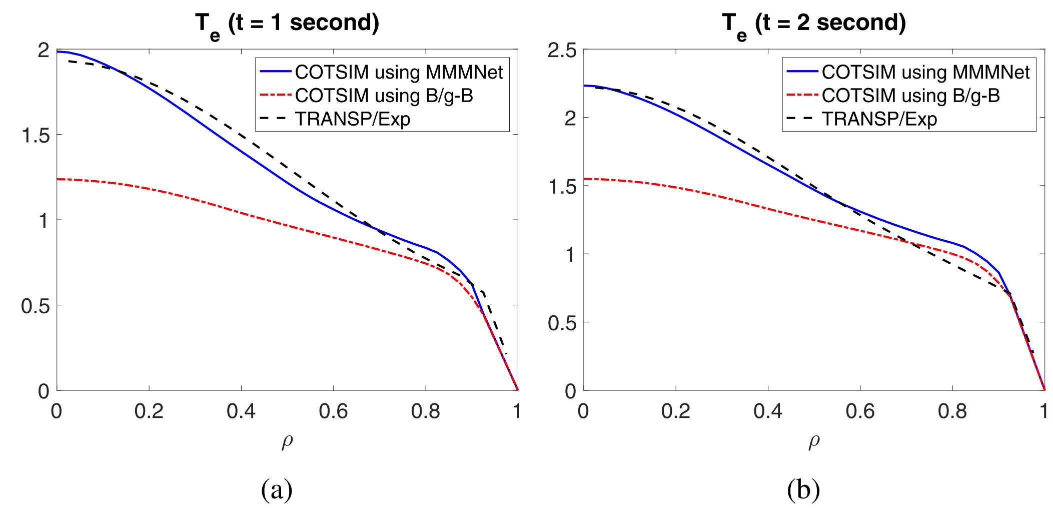


Figure 8. COTSIM prediction of the T_e profile for shot 147634 using χ_e from MMMnet and from the Bohm/gyro-Bohm (B/g-B) model compared to experimental data at (*a*) $t = 1$ s and (*b*) $t = 2$ s.

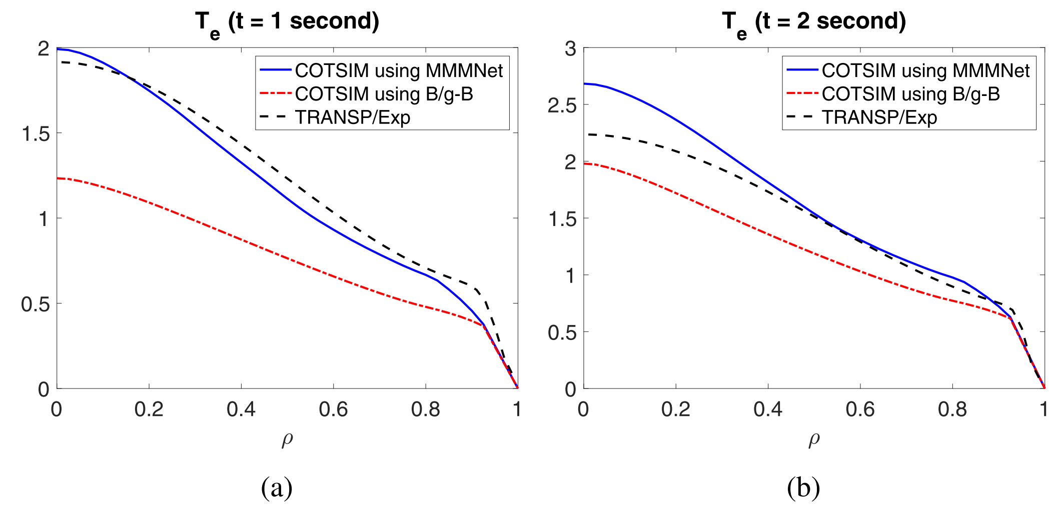


Figure 9. COTSIM prediction of the T_e profile for shot 147621 using χ_e from MMMnet and from the Bohm/gyro-Bohm (B/g-B) model compared to experimental data at (*a*) $t = 1$ s and (*b*) $t = 2$ s.

where Ω_ϕ is the toroidal angular velocity. This equation depends on the toroidal momentum diffusivity χ_ϕ, as well as ion density n_i, ion mass m_i, major radius R, auxiliary torque deposition τ_{nbi}, and the geometrical factor $\hat{H}$. Some analytical options to calculate diffusivities in COTSIM (Bohm/gyro-Bohm and Coppi-Tang) only calculate χ_e, so when one of those models is used an assumption must be made. In the Bohm/gyro-Bohm simulations shown here, it is assumed that the anomalous component of χ_ϕ is a scalar multiple of χ_e. If the χ_e and χ_ϕ profiles have different shapes, this assumption can make it difficult to accurately reproduce the shape of the rotation profile. MMMnet predicts χ_ϕ separately from χ_e, and is therefore expected to be better able to predict the shape of χ_ϕ. An extrapolation is made of the MMMnet prediction of χ_ϕ in the region where $0.8 \leqslant \hat{\rho} \leqslant 1$ because the neural network is not trained to produce valid results in that region. The MMM model does not provide toroidal momentum transport in the center of the profile and overpredicts transport in the confinement region for the discharges studied, and MMMnet replicates this behavior. As a consequence of the zero prediction in the center, a neoclassical component is added to the MMMnet prediction of χ_ϕ in the region from $0 \leqslant \hat{\rho} \leqslant 0.3$. The neoclassical component of χ_ϕ is assumed to be equal to the Chang–Hinton [18] neoclassical prediction of χ_i. To compensate for the overprediction of χ_ϕ in the core region, which leads to an underprediction of toroidal rotation, a coefficient is introduced to the MMMnet momentum transport. Predictions of the rotation profile using the χ_ϕ derived in this way are shown in figure 10 for shot 147634 and in figure 11 for shot 147621. For both shots the predictions using χ_ϕ from MMMnet are noticeably better than when a scalar multiple of the Bohm/gyro-Bohm χ_e prediction is used.

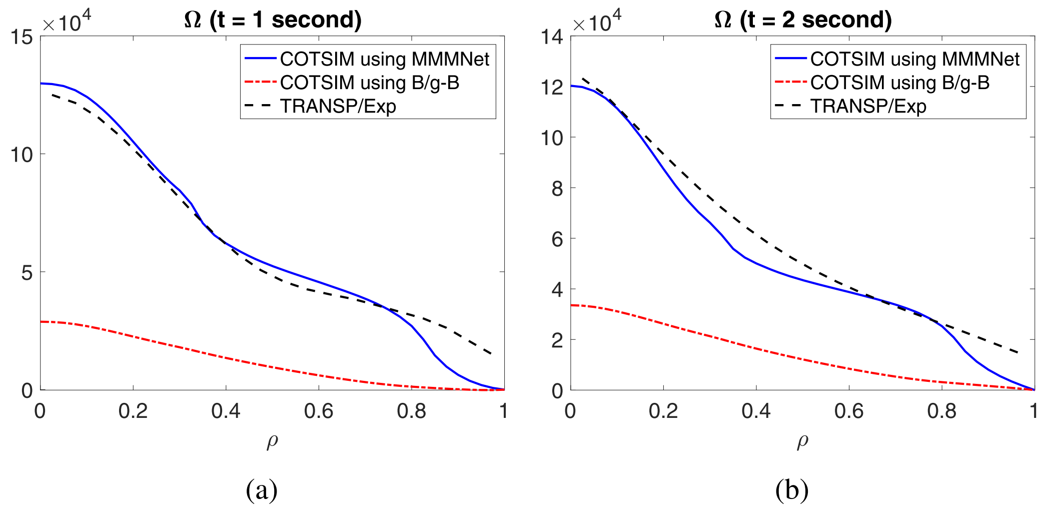


Figure 10. COTSIM prediction of the Ω profile for shot 147634 using χ_ϕ from MMMnet and from the Bohm/gyro-Bohm (B/g-B) model compared to experimental data at (*a*) $t = 1$ s and (*b*) $t = 2$ s.

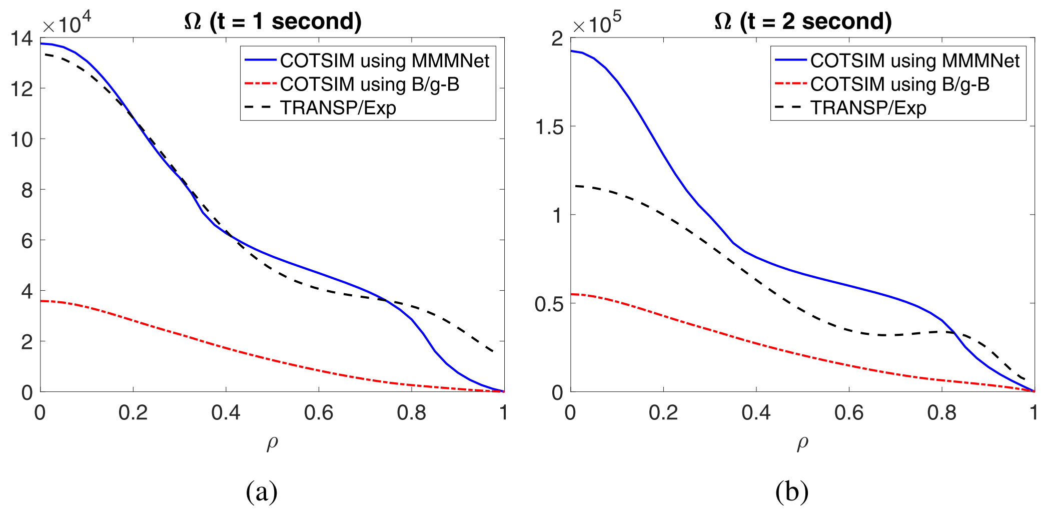


Figure 11. COTSIM prediction of the Ω profile for shot 147621 using χ_ϕ from MMMnet and from the Bohm/gyro-Bohm (B/g-B) model compared to experimental data at (*a*) $t = 1$ s and (*b*) $t = 2$ s.

6. Conclusions and future work

A multi-layer perceptron neural network with 3 hidden layers and 100 nodes per layer has been trained to emulate the results of MMM. This network is shown to be capable of meeting all three goals of control-oriented models: it reproduces MMM calculations with a high level of accuracy and a calculation time suitable for real-time control across a variety of plasma scenarios. In addition, an application of MMMnet in offline control-oriented predictive simulation is shown. MMMnet has been integrated into COTSIM and used to predict the electron thermal and toroidal momentum diffusivities. Preliminary results show that the use of MMMnet could produce electron temperature and rotation profile predictions closer in shape to the experimental profiles.

A number of improvements to MMMnet are being considered for future work. Other outputs of MMM, including impurity, poloidal momentum, and electron particle diffusivities, could be included as additional outputs of the neural network. Also, training data could be extended so that the network predictions are valid across the whole spatial domain, from the magnetic axis to the boundary, once MMM itself is updated to be valid in the pedestal region. An enlarged training data set could also extend applicability to an even broader range of plasma scenarios. These improvements would significantly increase the number of applications MMMnet is useful for. In COTSIM, future work will include testing on a wider variety of shots to determine the effect of using the neural network in different plasma scenarios. In addition, when a transport equation is added to COTSIM to calculate ion temperature, the MMMnet prediction of ion thermal diffusivity will be utilized.

Disclaimer

Acknowledgments

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Fusion Energy Sciences, using the DIII-D National Fusion Facility, a DOE Office of Science user facility, under Awards DE-FC02-04ER54698, DE-SC0010661, DE-SC0013977, and by the National Science Foundation Graduate Research Fellowship Program under Grant No. 1842163.

ORCID iDs

S.M. Morosohk https://orcid.org/0000-0002-3133-5095
A. Pajares https://orcid.org/0000-0001-9251-9675
T. Rafiq https://orcid.org/0000-0002-2164-1582

References

[1] Meneghini O. *et al* 2017 *Nucl. Fusion* **57** 086034
[2] Citrin J. *et al* 2015 *Nucl. Fusion* **55** 092001
[3] Boyer M. *et al* 2019 *Nucl. Fusion* **59** 056008
[4] Morosohk S.M., Boyer M.D. and Schuster E. 2021 *Fusion Eng. Des.* **163** 112125
[5] Rafiq T., Kritz A.H., Weiland J., Pankin A.Y. and Luo L. 2013 *Phys. Plasmas* **20** 032506
[6] Hawryluk R. 1981 *Physics of Plasmas Close to Thermonuclear Conditions* (Oxford: Pergamon)
[7] Breslau J., Gorelenkova M., Poli F., Sachdev J. and Yuan X. 2018 *TRANSP (Computer Software)* (https://doi.org/10.11578/dc.20180627.4)
[8] Nielsen M.A. 2015 *Neural Networks and Deep Learning* (San Francisco: Determination Press)
[9] Bernardos P.G. and G.-C.V. 2007 *Eng. Appl. Artif. Intell.* **20** 365
[10] Klein A., Falkner S., Bartels S., Hennig P. and Hutter F. 2017 Fast Bayesian optimization of machine learning hyperparameters on large datasets (*Proc. 20th Int. Conf. Artificial Intelligence and Statistics* 20 - 22 April 2017) (Fort Lauderdale, FL, USA vol 54) (PMLR) pp 528–36 (http://proceedings.mlr.press/v54/klein17a/klein17a.pdf)
[11] Chollet F. *et al* 2015 *Keras* (https://github.com/fchollet/keras)
[12] Behnel S. *et al* C-extensions for Python (https://cython.org/)
[13] Basiuk V. *et al* 2003 *Nucl. Fusion* **43** 822
[14] Erba M., Aniel T., Basiuk V., Becoulet A. and Litaudon X. 1998 *Nucl. Fusion* **38** 1013
[15] Jardin S.C., Bell M.G. and Pomphrey N. 1993 *Nucl. Fusion* **33** 371
[16] Onjun T., Bateman G., Kritz A.H. and Hammett G. 2002 *Phys. Plasmas* **9** 5018
[17] Goldston R. 1986 Topics in confinement analysis of tokamaks with auxiliary heating *Basic Physical Processes of Toroidal Fusion Plasmas: Proceedings of the Course and Workshop Held at Villa Monastero* vol 1 ed G.P. Lampis (Città di Castello) (Italy: Monotypia Franchi) pp 165–86
[18] Chang C. and Hinton F. 1982 *Phys. Fluids* **25** 536