




Data-driven profile prediction for DIII-D

J. Abbate^{1,2,4} , R. Conlin^{2,3,4}  and E. Kolemen^{2,3,*} 

¹ Department of Astrophysical Sciences, Princeton University, Princeton, NJ 08544, United States of America

² Princeton Plasma Physics Laboratory, Princeton, NJ 08543, United States of America

³ Department of Mechanical & Aerospace Engineering, Princeton University, Princeton, NJ 08544, United States of America

E-mail: ekolemen@pppl.gov

Received 6 October 2020, revised 15 January 2021

Accepted for publication 27 January 2021

Published 19 March 2021



CrossMark

Abstract

A new, fully data-driven algorithm has been developed that uses a neural network to predict plasma profiles on a scale of τ_E into the future given an actuator trajectory and the plasma state history. The model was trained and tested on DIII-D data from the 2013–2018 experimental campaigns. The model runs in tens of milliseconds and is very simple to use. This makes it a potentially useful tool for operators and physicists when planning plasma scenarios. It is also fast enough to be used for real-time model-predictive control.

Keywords: DIII-D, neural networks, transport, control

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

1. Introduction

Two primary goals for ITER operation are high fusion product and long (steady-state) pulse length. The relevant plasma properties for such optimization are the spatial profiles of electron and ion temperature, electron and ion density, rotation, and safety factor q (or equivalently $\iota \equiv 1/q$). Together, these properties encapsulate both fusion output and stability properties. It is crucial for ITER and other next-step devices to simulate the evolution of these plasma profiles over time for both (1) presetting actuator trajectories that guide the plasma into the desired state for a given experiment and (2) making finer adjustments to the actuators in real-time using model predictive control. In either case, the basic need is to predict how profiles will evolve into the future given the present state alongside a ‘proposal’ for how actuators will be set into the future. By predicting the evolution of the plasma profiles given a variety of ‘proposals’ for how the control knobs might be set, a user or algorithm chooses the settings that will take the profiles to the most desirable state.

A host of workflows can be considered for such plasma profile evolution. The traditional approach is integrated

modeling frameworks that iteratively calculate flux coefficients (the dynamics of which occur on turbulent timescales) and the plasma equilibrium (which is established on MHD timescales), then use this information alongside heat/particle deposition profiles, spatial boundary conditions, sawtooth/NTM simulations, and other experimental assumptions to integrate diffusion-like equations on much longer transport timescales [1, 2].

For example, the JETTO [3] transport code has been used to predict T_e , T_i , n_e , and rotation on JET scenarios [4]. It calls and passes information between a variety of modules, e.g. QuaLiKiz to calculate transport coefficients and the PENCIL code [5] to calculate beam deposition. TRANSP [6] is another popular transport code. With help from OMFIT [7] in writing namelists and submitting jobs, TRANSP has been used to predict T_e and T_i on DIII-D scenarios [8] with GLF23 [9] or TGLF [10] to calculate transport coefficients, and NUBEAM [11] for beam deposition.

For realtime applications, the control-oriented transport code RAPTOR has been used to predict T_e , T_i , and n_e profiles with QLKNN [12, 13] (a neural network approximation for QuaLiKiz) for JET scenarios [14]. That particular workflow so far just used actuator source profiles from previous runs. However, much progress has been made for calculating actuator source profiles in realtime while maintaining accuracy,

⁴ The two authors contributed equally to this paper.

* Author to whom any correspondence should be addressed.

e.g. for injected neutral beams [15, 16] and electron cyclotron waves [17].

In all of the above mentioned work, however, some assumptions or experimental data on uncontrollable quantities from within the prediction window are still given as input, including the Z_{eff} profile (or multiple impurity ion densities) and the spatial boundary condition on plasma profiles (or transport coefficients). OMFIT's STEP [18, 19] workflow seeks consistency among these quantities via core-pedestal and impurity transport coupling. Though not built for time-dependent predictions, this steady-state workflow is an important proof of concept for future work in this area. It uses TGLF-NN (a faster, neural network regression of TGLF) to calculate transport coefficients, STRAHL [20] to calculate impurity transport (used to estimate the shape of the Z_{eff} profile), and EPED1-NN (a faster, neural network regression of [21]) to calculate the boundary conditions on kinetic profiles. For time-dependent predictions, JETTO has recently been used to simulate all profiles plus multiple impurities via SANCO [22], though it still uses an ad hoc model to determine transport coefficients near the plasma edge [23].

We propose a fundamentally different approach to profile evolution that may prove useful for both offline scenario optimization and real-time model-predictive control. Rather than using integrated modeling with approximate submodules, we directly approximate the profile at a future timestep given the present profiles, present zero-dimensional plasma parameters, and future information on the actuator settings. Rather than requiring a sequence of models and approximations/assumptions with their compounding inaccuracies and often time-consuming computations, we seek a simple and fast ($\lesssim 100$ ms) input–output mapping between what physicists control (the actuators) and the metric that physicists care about (the evolution of plasma profiles). This work approximates such a mapping using a neural network trained on processed experimental data.

Machine learning trained exclusively on processed experimental data has already proven useful for a variety of fusion tasks. Event prediction, such as for disruptions and instabilities, is a particularly well-explored area. Various shallow learning algorithms have been applied, such as support vector machines [24], discriminant analysis [25], classification and regression trees [26], and random forests [27, 28]. Neural networks also have a long history in event prediction, employed by researchers at TEXT [29], DIII-D [30], ASDEX [31], JET [32], JT-60U [33], and ADITYA [34]. More recently, a disruption predictor combining recurrent and convolutional neural networks was developed, which helped inspire the architecture for the present work [35]. Several of these algorithms were tested on tokamaks in real-time, demonstrating that machine learning not only can provide state-of-the-art accuracy but also can be useful for control.

The model we present here is a first attempt at fully data-driven profile prediction. Due to the difficulties obtaining a large database of reliable profiles, it does not yet predict ion temperature and ion density profiles, instead predicting $iota \equiv 1/q$, n_e , T_e , rotation, and pressure. For the sake of simplicity, it

also works only during current flat-top, and considers a somewhat limited set of possible scenarios and control knobs (as described in section 2.1).

2. Machine learning model

The time evolution of the plasma state can be generically described by a differential equation of the form:

$$\frac{dx}{dt} = f(x, u), \quad (1)$$

where x is a vector of the plasma state, consisting of the values of the profiles suitably discretized in space, and u is a vector of control inputs. This can be put in a discrete time form:

$$x_{t+1} = x_t + f(x_t, u_t), \quad (2)$$

where t is a time index. In this paper, we attempt to approximate the function f using a neural network.

2.1. Data processing

Data is loaded and partially processed within the OMFIT framework [7]. The rest of the processing, along with the model training, is done on 1 NVIDIA V100 GPU and 8 IBM POWER9 CPU cores on Princeton University's traverse computing cluster. This code is publicly available⁵.

DIII-D shots from the 2013 through 2018 campaigns are collected from the MDS + database [36]. Shots with a pulse length less than 2 s, a normalized plasma beta less than 1, or a non-standard tokamak divertor topology are excluded from the start. We exclude all data after a disruption occurs. For simplicity in this first iteration of the model, we chose to use just four actuators. We therefore also exclude shots using electron cyclotron heating and 3D coil operation (e.g. used for resonant magnetic perturbations). We also do use gas control, but we read the density target for a low-level feedback controller rather than the gas flow rates themselves; therefore we only include samples when the feedback controller is in use, as opposed to when operators set the gas flow rates directly.

All signals are then put on the same 50 ms time base by averaging all signal values available between the present time and 50 ms prior. If no data is available in the window, the most recent value is floated. Only shots in which all signals are available are included. For this first iteration of the model, we only include data during plasma current flat-top.

The model considers data in 50 ms non-overlapping windows. This is a spacing large enough to smooth over most variations in modulated signals like the injected power, but small enough that significant changes in the profiles are not missed. Each window corresponds to a single 'timestep'. There are three fundamental types of data we consider: plasma profiles, plasma parameters, and actuators. The plasma state is determined by the plasma profiles and the parameters. Profiles are one-dimensional signals like temperature and density. Parameters are zero-dimensional signals like average density, plasma

⁵ <https://github.com/jabbate7/plasma-profile-predictor>

Table 1. Signals used in profile prediction neural network. ‘Source’ denotes the algorithm or MDS + pointname from which the data was obtained.

	Name	Source	Units
Profiles	Electron density	Zipfit	10^{19} m^{-3}
	Electron temperature	Zipfit	keV
	Ion rotation	Zipfit	kHz
	Rotational transform ^a (ι)	EFIT01	—
	Plasma pressure	EFIT01	Pa
Actuators	Injected power	pinj	MW
	Injected torque	tinj	N·m
	Target current	iptipp	A
	Target density	dstdenp	10^{19} m^{-3}
Parameters	Top triangularity	EFIT01	—
	Bottom triangularity	EFIT01	—
	Plasma elongation	EFIT01	—
	Plasma volume	EFIT01	m^3
	Normalized internal inductance	EFIT01	—
	Line avg. density	dssdenest	10^{19} m^{-3}

^aWhile the safety factor q is more common in tokamak physics, due to the singularity at the separatrix, the rotational transform $\iota = 1/q$ was found to present fewer numerical difficulties.

volume, and triangularity. Actuators are knobs that are set by the operator, including injected power and target current. A full list of signals used by the algorithm is shown in table 1. We also plot the distribution over training samples of values for various signals in figure 1.

Although ion temperature and density are important parameters for fusion, we do not include them in this first iteration of the model since the signals are often less reliable and therefore would require more careful analysis and fitting methods. We also might achieve better predictions and more complete control by replacing the total beam power and torque with the lower level individual beam voltages, perveances, and energies, but for simplicity we focus on only the total power and torque. This choice is still relevant for control, as we can use total power and torque as a proxy for individual beam settings via the existing DIII-D ‘VEP’ controller [37]. Similarly, as a proxy for individual gas flow rates, we use the target for the existing DIII-D line-averaged density controller (see [38] and references therein for details on the density controller).

The model takes in actuator and plasma parameter signals 6 timesteps (i.e. 300 ms) into the past. It takes in the profiles at the present timestep. And it takes in a ‘proposal’ for the actuators at each of 4 timesteps (200 ms) into the future. Throughout the paper, we will call this full set of information a ‘sample’. With the sample as input, the algorithm predicts the change in each of the profiles 200 ms (4 timesteps) into the future. This 200 ms prediction window was chosen based on the typical energy confinement time (τ_E) at DIII-D (see figure 2), and was empirically found to be a period over which the profiles change noticeably while not so long that the future state cannot be reliably predicted. The lookback of 300 ms (6 timesteps) was determined by trial and error, and also by rough estimates of correlation times between actuator inputs and profile response.

Temperature, density and rotation profiles are from DIII-D’s automatic profile fitting code Zipfit, which does some

basic normalization then runs a variety of fits (including tension splines and a modified hyperbolic tangent plus inverse Zernike polynomial method). Zipfit returns the fit with the lowest $|\chi_{\text{reduced}}^2| \propto \sum_i \left(\frac{(y_i - \hat{y}(x_i))^2}{y_{i,\text{error}}^2} \right)$ for y_i the i th measurement (calculated from diagnostics), $y_{i,\text{error}}$ the error bar of the i th measurement, and $\hat{y}(x_i)$ the fit value at the point of the i th measurement [39]. Pressure and rotational transform profiles are automatically generated with the DIII-D equilibrium code EFIT [40]. Zipfit profiles are downsampled onto 33 equally spaced points in normalized toroidal flux coordinates, denoted by ρ , where $\rho = 0$ is the magnetic axis and $\rho = 1$ is the last closed flux surface. EFIT profiles are similarly downsampled but onto a basis of normalized poloidal flux denoted by ψ . This downsampling of data that was already fit is done to reduce the size of the input data. Note that EFIT and Zipfit are not completely causal calculations because they average diagnostic constraints from the future. However, this future information is at most 20 ms into the future, much less than the 200 ms prediction window and also less than half of the energy confinement time. Nonetheless, in the future we plan a more standardized fit to diagnostics that will be fully causal.

Each signal was normalized by subtracting out the median value and dividing by the inter-quartile range. This was found to be more robust against outliers than the more common method of subtracting the mean and dividing by the standard deviation.

2.2. Model architecture

A sketch of the model architecture is shown in figure 3. Given the spatially distributed information contained in the plasma profiles, we take inspiration from image classification models and use a sequence of convolutional layers to attempt to capture local effects in the profiles, such as gradients and transport coefficients. Using multiple convolutional layers with varying

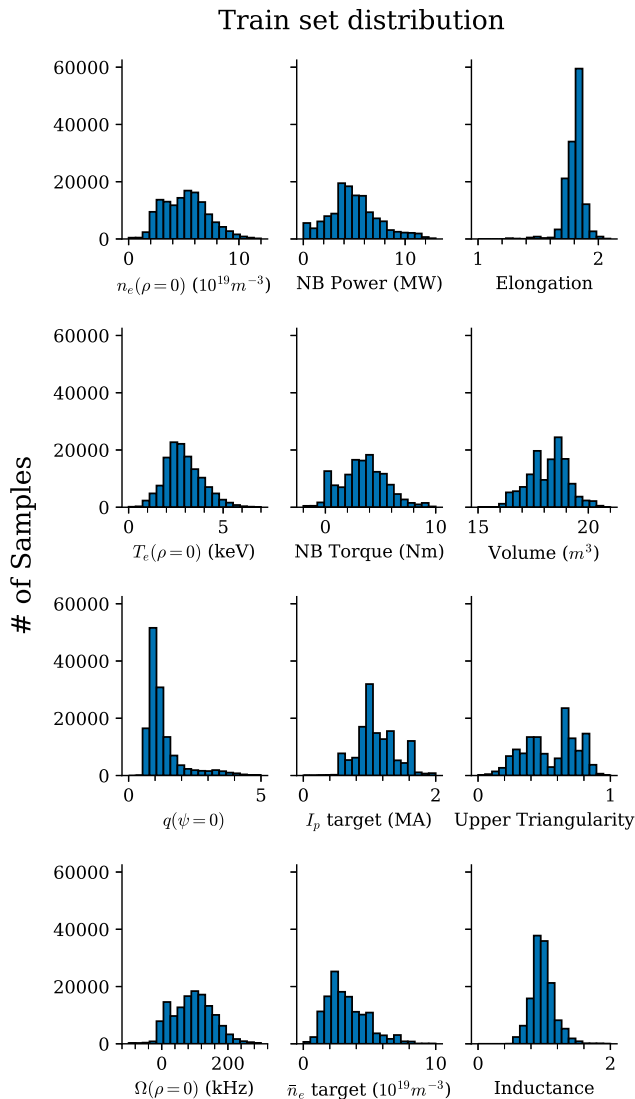


Figure 1. Histogram of various input signals for all training samples. The left column shows the core-most value of electron density, electron temperature, q , and rotation. The middle column shows the four actuators. The right column shows a few of the parameters.

kernel sizes has been shown to help the model learn information over a wide range of spatial scales [41, 42]. Multiple profiles are also processed at the same time, with each profile treated as a ‘channel’ of a 1D image (cf, RGB channels for a 2D image). This allows the model to learn correlations between the different profiles. Each layer expands the number of channels and the final output of the initial convolutional layers is an array of size $n \times m$, where n is the number of spatial points (33) and m is a hyperparameter to be tuned (denoted as ‘channels’, discussed further in section 2.4). Each channel of this array can be considered a ‘feature profile’ that is a nonlinear combination of the various input profiles.

The parameter and actuator data is fed through a series of fully connected and long-short term memory (LSTM) layers [43]. The fully connected layers attempt to measure the effect of the given actuator or parameter at each spatial point in the plasma, while the LSTM layers look over the time history of the signals to account for the fact that in many cases it

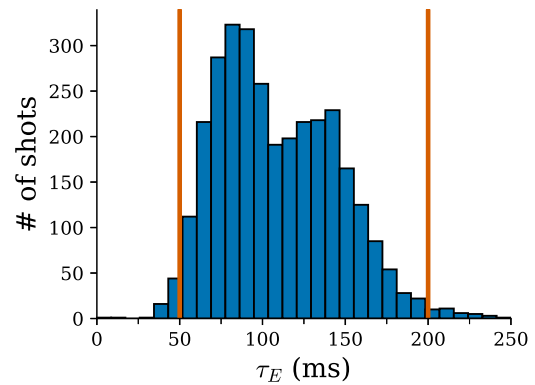


Figure 2. Histogram of median energy confinement times for the shots included in the training set. The vast majority of confinement times lie above 50 ms (the left red line) and below 200 ms (the right red line), hence the choice to average over 50 ms windows and predict 200 ms into the future.

takes time for a given actuator to propagate its effect through the plasma. The final output from this section of the model is again an array of size $n \times m$, which is then added to the ‘feature profiles’ output by the initial convolutional layers.

This combination of profile and actuator information is then fed through a second sequence of convolutional layers, with a unique output path for each predicted profile. Each of these output paths reduce the m ‘feature profiles’ down to a single output profile, which is the model’s estimate of the given profile 200 ms in the future.

All layers in the network use the ReLU activation function, which has good performance and ease of training [44, 45]. The one exception is the LSTM layers, which use a hard sigmoid function for their recurrent activation. This bounds the output, which helps improve convergence during training [46].

2.3. Model training

The goal of a neural network is to approximate a function by optimizing the parameters of a model (described above). The neural network can be described by a function of the form

$$f(\mathbf{x}, \mathbf{u}, \boldsymbol{\theta}), \quad (3)$$

where \mathbf{x} and \mathbf{u} are the state and control, and $\boldsymbol{\theta}$ are the weights to be learned through training. Training consists of attempting to find the weights $\boldsymbol{\theta}^*$ that minimize a cost function. In our case, the cost function is a mean squared error. The optimization problem can then be expressed as

$$\boldsymbol{\theta}^* = \arg \min_{\boldsymbol{\theta}} \frac{1}{N} \sum_i^N \|\mathbf{x}_{t+1}^i - f(\mathbf{x}_t^i, \mathbf{u}_t^i, \boldsymbol{\theta})\|^2 + \lambda \|\boldsymbol{\theta}\|^2, \quad (4)$$

where i is an index that ranges over all N samples in the training dataset, and λ is a regularization parameter that can be tuned to prevent overfitting.

We use Keras [47] with a backend of tensorflow to manage the traditional neural net training workflow. Over the training set, the package computes the steepest-descent step in the loss function over the model-weight space and saves it. After trial-and-error, we chose a batch size of 128. So after every 128th

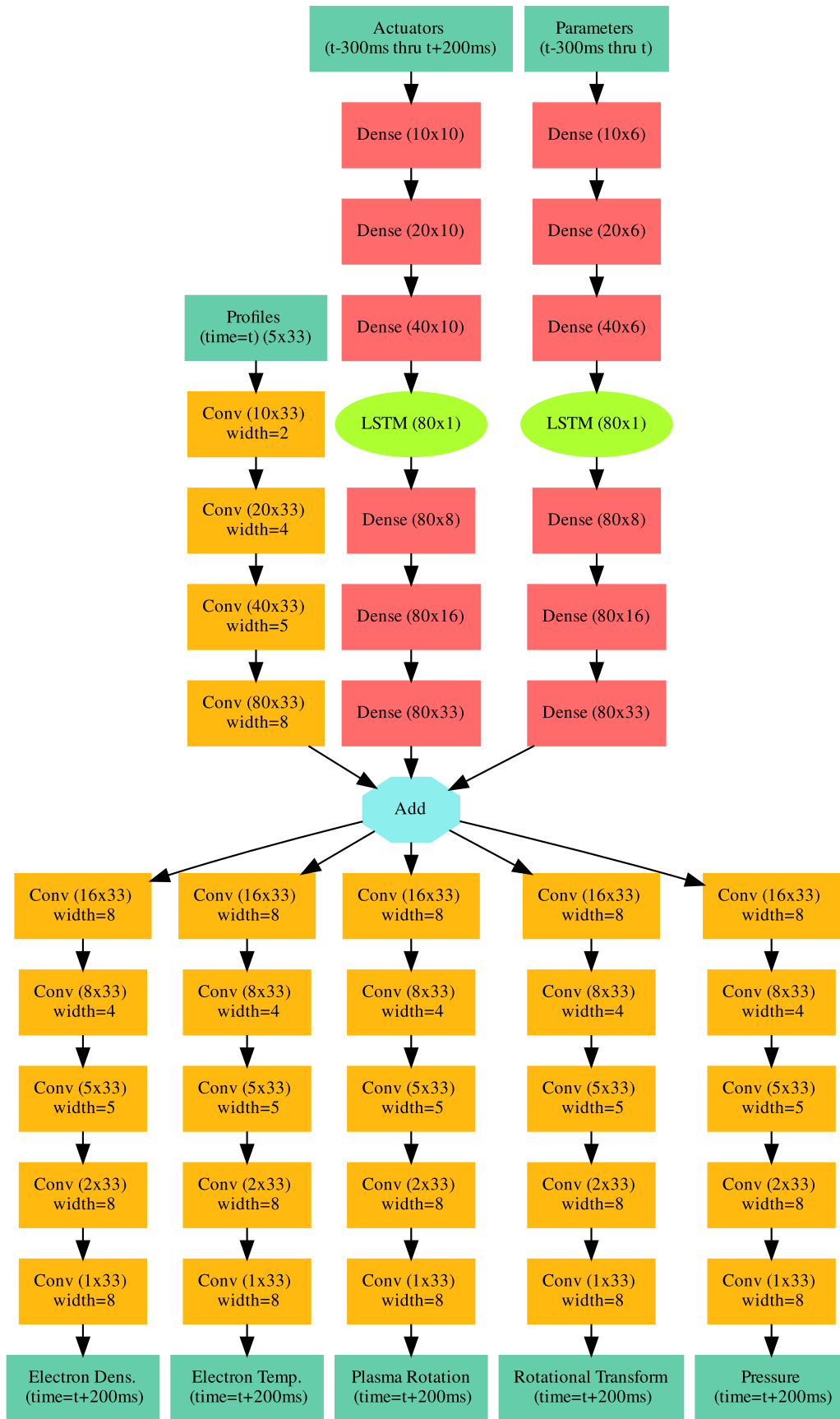


Figure 3. Model architecture showing the algorithm’s process of combining parameter, actuator, and profile data into a set of ‘feature profiles’, and then ultimately outputting predictions for each signal based on that block of information. The activation function used in all layers is ReLU.

Table 2. Hyperparameters varied during model development, and the resulting loss values averaged over all models. The hyperparameters selected for the final model are bolded. In all cases, the selection with the minimum loss (the chosen model) also had the lowest average loss, indicating that the choice of hyperparameter is robust.

Architecture:	Convolutional		Fully Connected	
Avg. loss:	0.080		0.572	
Channels:	32	64	128	192
Avg. loss:	0.137	0.080	0.037	0.051
Shape params:	No		Yes	
Avg. loss:	0.083		0.040	
Sample weighting:	None		Variations	
Avg. Loss:	0.173		0.101	

randomly selected sample, the package does a pseudo-average (we use the adaptive gradient optimization algorithm ADA-GRAD [48]) of the previous 128 samples and then updates the model weights and recomputes the derivatives to be used for computing the steepest-descent paths for the next batch of 128 samples. We repeat this process over the entire dataset until either the validation loss stops improving for more than eight consecutive steps ('early stopping') or 200 times ('epochs'), whichever comes first. This entire process takes about 5 ho of wall-clock time per model.

Shots in our database were split into 3 groups based on the last digit of the shot number. Shots with a number ending in 0 are reserved for a test set, accounting for 389 shots containing 15 132 samples that were not seen by any models during training or hyperparameter tuning. Shots ending in a randomly chosen number (other than 0) were used for the validation set and hyperparameter tuning, accounting for 15 772 samples from 415 shots. The remaining 121 242 samples from 3117 shots were used for training.

2.4. Model tuning

Another key choice in machine learning models is the hyperparameters, such as layer type, hidden layer size, type of activation function, and data normalization/weighting scheme. We follow the traditional machine learning workflow. For a variety of possible models (defined by their hyperparameter values), the training set is used to stochastically optimize the model weights. We then compute the loss function for each of the models over the validation set and choose the model which yields the lowest loss.

Initial random-hyperparameter searches demonstrated clear dominance of some hyperparameters over others. For example, we tried a variety of loss functions (ReLU [49], ELU [50], SELU [51], sigmoid, tanh [52]) and ReLU was found to perform best overall. We then used a fixed-grid hyperparameter tuning process to arrive at the final hyperparameter values. Explanations of the hyperparameters in this grid search are discussed below, and summarized in table 2 and figure 4.

The first hyperparameter we considered is the type of hidden layer used. We considered both convolutional and fully connected layers. It was found that the convolutional layers systematically performed better than fully connected, which might be physically understood by recalling that convolutional

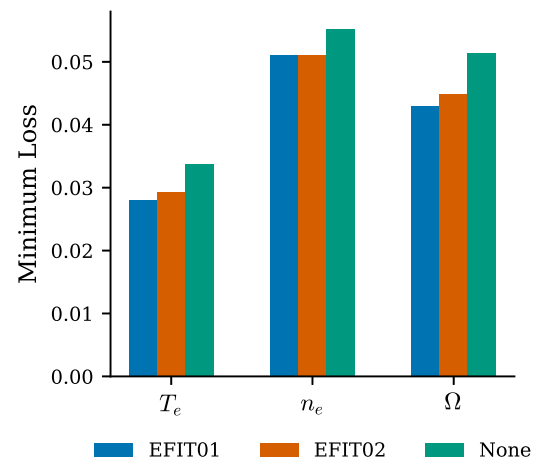


Figure 4. Comparison of minimum validation loss for models using pressure and rotational transform profiles from EFIT01, EFIT02, and without. In all cases, having information about pressure and rotational transform improved the predictions for temperature, density, and plasma rotation, and inputting EFIT01 data is slightly better than inputting EFIT02 data.

layers help couple local information, which resembles the local gradients that drive transport equations.

A related hyperparameter is the maximum size of the hidden layers, as given by the number of output channels in the middle of the model. It was found that a maximum hidden layer size of 128 channels provided the best results, with smaller models generally not able to fit the data well. With 128 channels, the model had approximately 10 million parameters, which in some cases led to overfitting, so L2 regularization was used with a magnitude of $\lambda = 10^{-4}$. Larger models tended to overfit even with much higher levels of regularization.

We also found that supplying the model with parameters to describe the plasma shape improved performance. We chose to use scalar shape parameters rather than the full 2D flux shapes to keep the model size manageable, though future work may include more detailed shape information.

Another hyperparameter we considered was weighting the importance of samples differently during training. It was observed that many of the shots in the training set are fairly similar, and across many timesteps the profiles can be roughly constant. One common method to mitigate the effects of an unbalanced training set is weighting the samples relative to

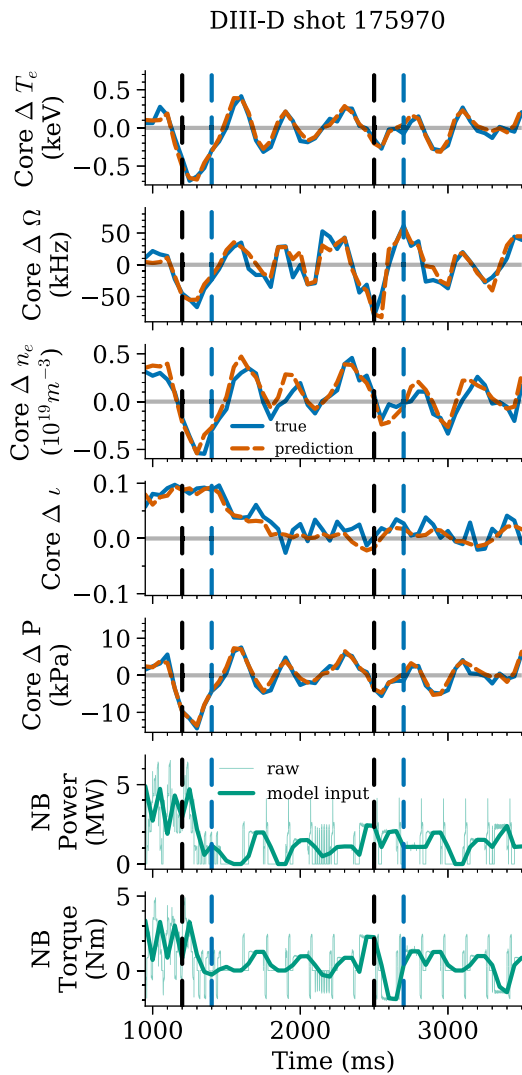


Figure 5. Model prediction vs experimental measurement of core (averaged over ρ/ψ from 0 to 0.3) changes in various signals between the indicated timesteps and 200 ms beyond it. The neutral beam power and torque, whose fluctuations drive the dynamics of the shot, are also plotted. The dashed black lines indicate the timesteps plotted in figures 6 and 7. The blue dashed line on the neutral beam plots are 200 ms ahead of the black dashed line, so depict the final timestep of future data that informs the prediction happening at the black dashed line.

their frequency. This is easiest to implement for classification tasks with a finite number of classes, where the number of members of each class can serve as the weighting parameter. This can be difficult for regression problems, where the number of classes is effectively infinite, so we opted for a simpler approach of weighting the samples by the total variation of the actuators in the 300 ms lookback window. To validate performance, we turned this sample weighting off. We found that weighting the samples in this fashion did improve performance on average and led to the best performing models when combined with the other choices of hyperparameters. We suggest that this weighting effectively instructed the model to focus more on samples with large variations, which would help it to learn the dominant physics driving transport.

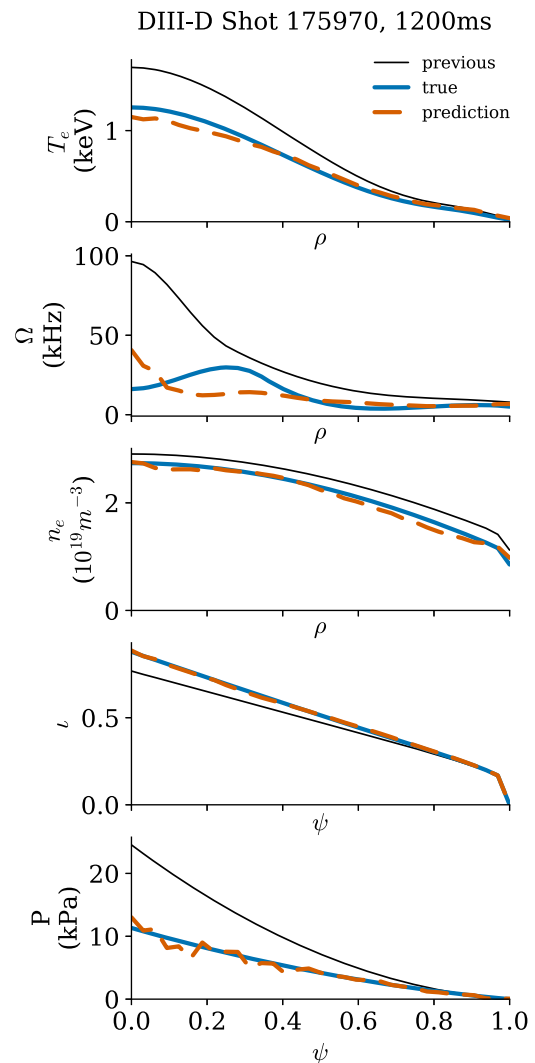


Figure 6. Model prediction vs experimental measurement at the left time slice in figure 5. The gray curve shows the true experimental profile at time t , the orange dashed curve shows the model prediction for the profile at time $t + 200$ ms, and the blue curve shows the true experimental profile at $t + 200$ ms. Note this prediction is made given a decreasing future neutral beam power and torque, hence the decrease in temperature and rotation.

The final consideration was which type of EFIT equilibrium reconstruction to use for the pressure and rotational transform profiles. EFIT01 is a magnetics only reconstruction of the equilibrium, and so is generally only accurate for estimating global parameters. EFIT02 uses motional Stark effect data to constrain the safety factor profile and so is generally a more accurate estimate of the equilibrium. For each of (1) EFIT01 pressure/rotational transform profiles alongside kinetic profiles as profile inputs (2) EFIT02 pressure/rotational transform profiles alongside kinetic profiles as profile inputs and (3) only kinetic profiles as profile inputs, a variety of different models (over a similar grid of hyperparameters as shown in table 2) were trained to predict the kinetic profiles only. For each of the three cases, we used the minimum validation loss across models (i.e. the loss corresponding to the hyperparameter-

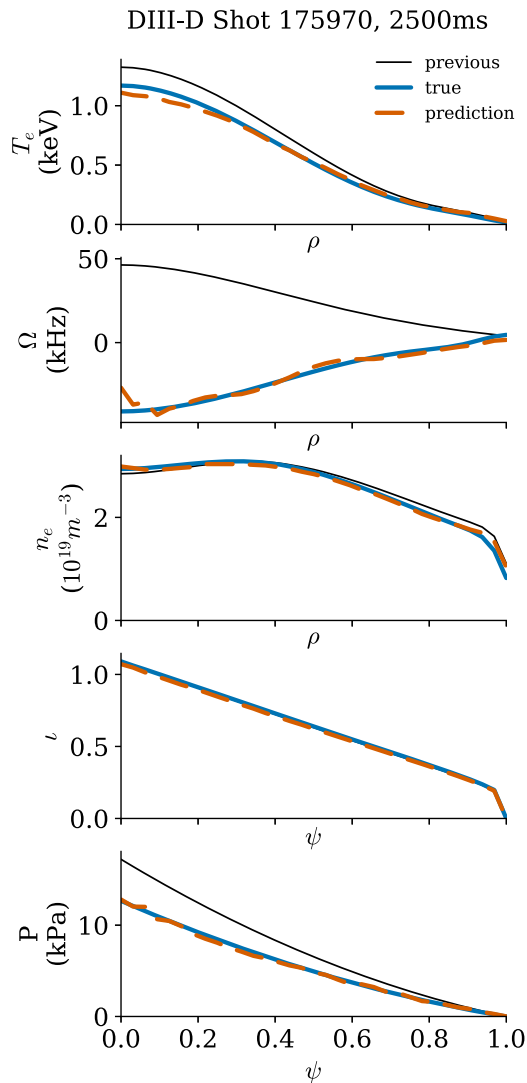


Figure 7. Model prediction vs experimental measurement at the right time slice in figure 5. The future neutral beam power is only slightly decreasing while the neutral beam torque is decreasing a lot, hence the slight decrease in temperature but major decrease in rotation.

tuned model) as a measure of how helpful the profile inputs are in predicting the kinetic profiles. As shown in figure 4, including any type of magnetic profiles results in better predictions for temperature, density, and plasma rotation. However, despite profiles from EFIT01 generally being inaccurate, using EFIT01 data results in slightly better predictions overall than EFIT02. This could be due to EFIT01 data being available for slightly more training samples (121 242 in the training set for EFIT01 compared to 106 720 for EFIT02, an 11% difference). It could also be due to the higher fidelity EFIT02 data containing more detail than is strictly necessary, making the relevant information harder to extract. In any case, future work is currently underway to build a large database of kinetically constrained equilibria using CAKE [53] which will be used to train future models. We also hope to more systematically evaluate the sensitivity of model performance to different data fitting and filtering methods.

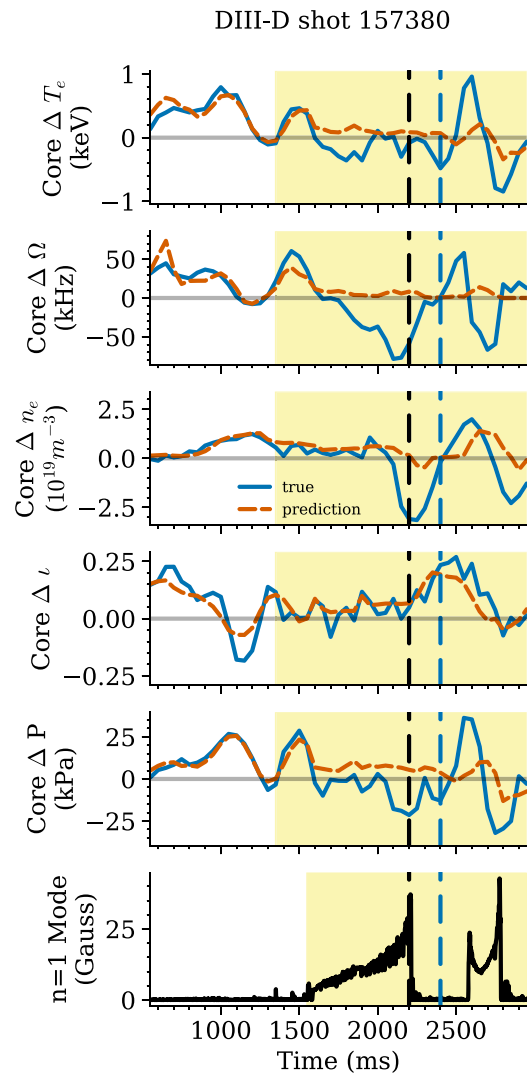


Figure 8. Model prediction for a shot with a growing tearing mode starting at 1500 ms (highlighted in yellow and quantitatively measured by the RMS strength of the $n = 1$ magnetic fluctuation, depicted in the bottom plot). Predictions are mostly accurate until the mode begins to impact the dynamics.

3. Model performance

To measure model performance, we will compare model predictions of changes in profiles to the experimental truth over samples from the never-before-seen test set (shots ending in 0). Computing the prediction for a single sample takes a few tens of milliseconds on a standard modern computer.

We start by looking at a few individual experiments. In figure 5, we examine the model's ability to predict profiles during the beam-modulated shot 175970. We can also look at model performance for predicting full profiles at the timeslices indicated by figure 5's black dashed lines. These profile predictions are for 1200 ms in figure 6 and 2500 ms in figure 7. Note that we are plotting the raw output of the model to give the clearest picture of what the model returns, although the output is sometimes not smooth. These profiles could be used as-is, e.g. in a controller that only needs the overall shape and direction of change over time. For other applications, the profiles

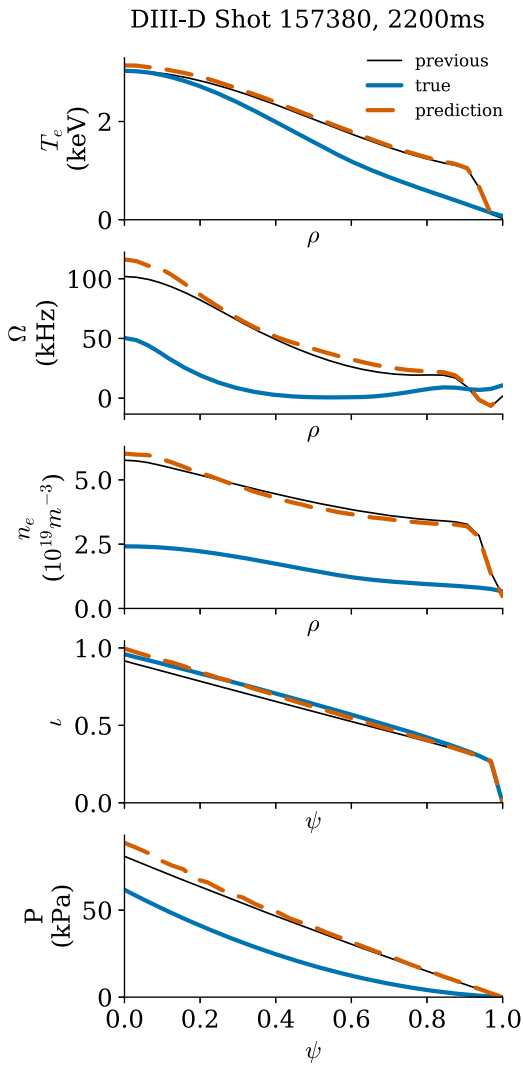


Figure 9. Full profile predictions for the timeslice indicated in figure 8. The dynamics are governed by the tearing mode, which is information unknown to the model.

could be smoothed as a post-processing step.

The predictions in the first timeslice (figure 6) are relatively accurate for all profiles except rotation, which has the correct direction and magnitude of change but the wrong shape. This may be because we are using the total injected torque, which neglects that each individual beam deposits particles at a different angle and location. As previously mentioned, an upgrade to this model might take individual beam powers and torques as input. Nonetheless, for the second timeslice in figure 7, we see better agreement in the rotation, suggesting the approximation of total torque is valid enough for some cases.

Predictions for shot 175970 are relatively good. We also selected a relatively bad example: shot 157380, whose time trace is shown in figure 8 and whose time slice at 2200 ms is shown in figure 9. Despite predictions being bad overall, we saw the accuracy is actually quite good until 1500 ms (prior to yellow highlight in the time trace). We plotted the $n = 1$ mode amplitude (bottom panel of time trace) and realized that predictions begin to deteriorate right as a tearing mode begins to grow. The present model is not given signals known to forecast

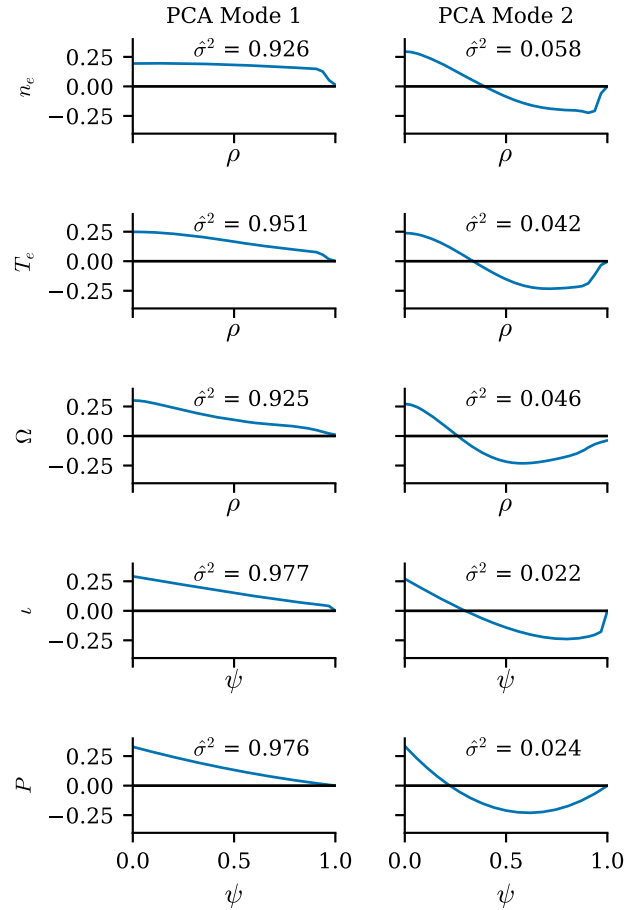


Figure 10. First two PCA modes (columns) for each profile (row). $\hat{\sigma}^2$ denotes the explained variance ratio for the given mode. The explained variance ratio is a measure of how much of the variance in the full dataset is accounted for in a given PCA mode, normalized by the total variance in the dataset. For all profiles, the first two PCA modes are sufficient to capture more than 90% of the variance in that profile.

tearing modes, such as the amplitude of the $n = 1$ and $n = 2$ modes [28], so this may explain the bad predictions. Including these mode amplitudes and similar signals in the future could improve performance on shots like this one whose dynamics are governed by instabilities.

In addition to individual shot analysis, we can also compare model predictions to true predictions for a few selected points or moments of the profiles. Instead of tracking individual point predictions as we did for the core in the timetraces of individual shots, we opt here to track the principle component analysis (PCA) coefficients of the profiles so we can encapsulate more information in a single plot and get a sense for how well the model predicts both bulk changes and shape changes.

PCA works by finding an orthogonal set of basis vectors for a given dataset, with the desirable feature that the first PCA mode captures the maximum possible variance in the dataset, while the second mode captures the most possible variance while being orthogonal to the first mode, and in general the i th mode captures the maximum possible variance while remaining orthogonal to all modes $j < i$. Using the PCA modes as

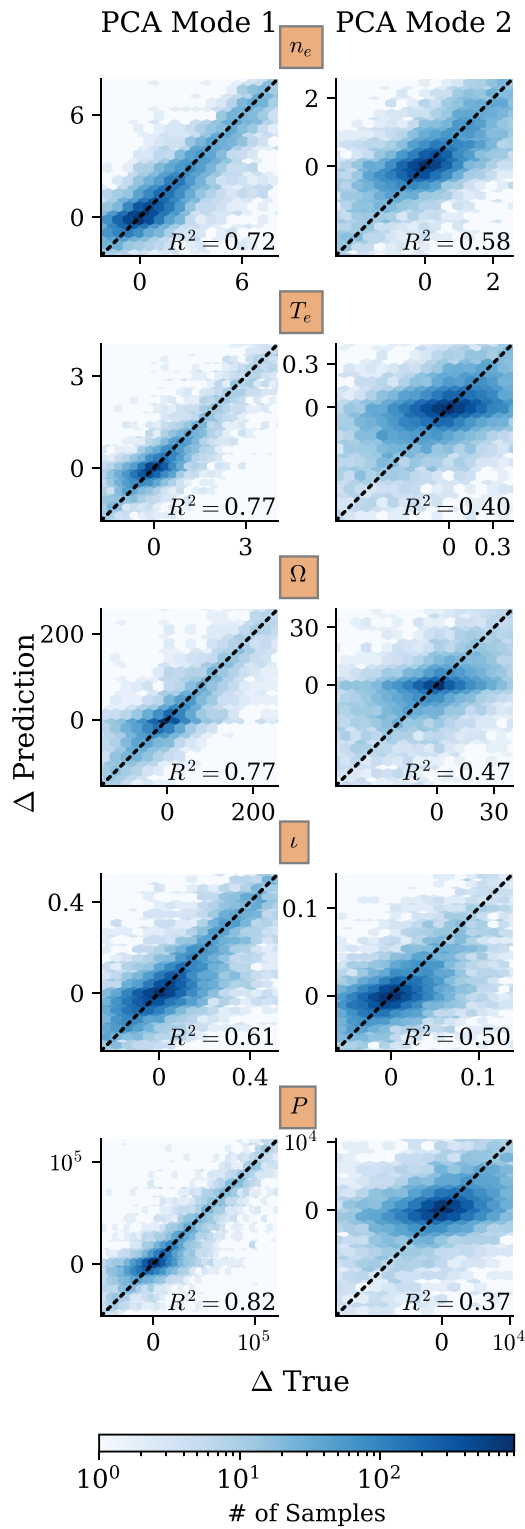


Figure 11. Comparison of the predicted changes over 200 ms in PCA coefficient 1 (left column) and 2 (right column) to the true changes in the PCA coefficients, for all samples in the test set. Rows correspond to the five profiles. R^2 accuracy is shown at bottom right of each subplot, and is 1 for perfect accuracy. All axes are linearly scaled, but the number of samples is logarithmic.

basis functions, we can decompose a given profile as

$$f(\psi) = \sum_i a_i v_i(\psi) \quad (5)$$

where v_i are the PCA modes, and a_i are weighting coefficients. Given that the modes capture the maximum possible variance, only a few terms are usually needed to describe a profile very accurately. Illustrations of the two most dominant modes (v_1 and v_2) for each profile are shown in figure 10, along with their explained variance, normalized to 1 (higher explained variance corresponds to greater importance).

We can examine how well the a_i of the predicted profiles match the a_i of the true profiles. In figure 11 we compare the predicted change in the first two PCA coefficients to the true change, for all test samples. The dotted black line is drawn at 45 degrees and denotes perfect accuracy where the prediction matches the true value. Points above the line indicate the model is over-predicting the coefficient, and points below indicate the model is under-predicting the coefficient. The R^2 value at the bottom right of each subplot is the statistical ‘coefficient of determination’ for predicting the true change in the coefficient is exactly equal to the predicted change in the coefficient. If the model is perfect, this value would be 1.

We see that alignment of samples with the black line for the first PCA coefficient is relatively good, all profiles having an R^2 value above 0.6. Performance is worse but still much better than blind guessing for the second PCA coefficient, meaning the model is learning not just the bulk direction of change, but also about the higher-order shaping. The degradation of performance between the first and second PCA coefficient is to be expected as the second mode represents significantly less information.

4. Discussion and conclusion

We demonstrated that a machine learning model can directly approximate both the bulk and shape of profiles 200 ms into the future, given only experimental data at the present timestep and a ‘proposal’ for actuators into the future. Given that this model runs in tens of milliseconds and is very simple to use, it is a potentially useful tool for operators and physicists when planning plasma scenarios. Additionally, we have already successfully implemented a model-predictive control algorithm on DIII-D using a reduced version of the model that runs in hundreds of microseconds and reads only real-time information. A paper on this work is forthcoming.

An important caveat remains. Shots tend to be reloaded from previous experiments when testing new physics, so that despite having thousands of shots many are very similar. In general a neural net finds the easiest way to match its target, and so could be learning to identify the relatively small number of discharge patterns rather than recognizing physics of transport and source deposition. Whether the neural net is

learning useful patterns or simply memorizing past discharge evolution is difficult to answer. One imperfect methodology for future work would be to compare the neural net's predicted response to each actuator with that of a theoretical model. This would allow independent and flexible variation of the inputs. Another mechanism will be our planned experiments allowing the model to control the plasma in real-time through a variety of unseen scenarios. Finally, having a larger database of shots where semi-random changes to actuators are made during flat-top would be a helpful supplement to the existing DIII-D database. In addition to helping improve data-driven models, this would also help develop and validate physics-based models.

There are a few techniques to make the model perform even better, as mentioned throughout the paper. First, some information is still missing. For example, $n = 1$ and $n = 2$ mode amplitudes, and perhaps fluctuation (from ECE or BES) and radial electric field measurements may help. More challenging would be including wall-condition information (such as how recently the machine was boronized). Second, it would be desirable in the future to predict more profiles (such as ion temperature and density), account for more scenarios (such as current ramp-up and non-standard topologies), and use more actuators (such as shape control, ECH, and individual beam powers and torques rather than just the totals).

Acknowledgments

The authors thank Carlos Paz-Soldan for describing how to identify non-normal coil operation, Brian Grierson and Orso Menighini for help understanding predictive TRANSP, Sterling Smith and David Eldon for help reading in data, Oak Nelson for help with OMFIT and DIII-D tools, summer intern Alex Liu for helping with sensitivity analysis and sanity checking, Phil Snyder for a useful discussion about how to compare physics models and neural nets, and previous summer intern Jalal Butt for useful discussions. 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 DC-AC02-09CH11466, DE-SC0015480, DE-SC0015878, DE-AR0001166, DE-FC02-04ER54698, and Field Work Proposal No. 1903.

Disclaimer

This report is prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency

thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

ORCID iDs

J. Abbate  <https://orcid.org/0000-0002-5463-6552>

R. Conlin  <https://orcid.org/0000-0001-8366-2111>

E. Kolemen  <https://orcid.org/0000-0003-4212-3247>

References

- [1] Ongena J.P.H.E., Voitsekrovitch I., Evrard M. and McCune D. 2012 *Fusion Sci. Technol.* **61** 180–9
- [2] Mantica P., Angioni C., Bonanomi N., Citrin J., Grierson B.A., Koechl F., Mariani A. and Staebler G.M. 2019 *Plasma Phys. Control. Fusion* **62** 014021
- [3] Cenacchi G. et al 1988 JETTO A free boundary plasma transport code *Technical Report vol 84* Italy
- [4] Citrin J. et al 2017 *Plasma Phys. Control. Fusion* **59** 124005
- [5] Challis C.D., Cordey J.G., Hammén H., Stubberfield P.M., Christiansen J.P., Lazzaro E., Muir D.G., Stork D. and Thompson E. 1989 *Nucl. Fusion* **29** 563–70
- [6] Hawryluk R. 1979 An empirical approach to tokamak transport *Course on Physics Plasmas Close to Thermonuclear Conditions* (17 February 2021) (<https://transp.pppf.gov/files/Hawryluk.pdf>)
- [7] Menighini O. et al 2015 *Nucl. Fusion* **55** 083008
- [8] Grierson B.A. et al 2018 *Fusion Sci. Technol.* **74** 101–15
- [9] Waltz R.E. et al 1997 A gyro-Landau-fluid transport model *Phys. Plasmas* **4** 2482
- [10] Waltz R.E. et al 2013 *Phys. Plasmas* **20** 012508
- [11] Goldston R.J., McCune D.C., Towner H.H., Davis S.L., Hawryluk R.J. and Schmidt G.L. 1981 *J. Comput. Phys.* **43** 61–78
- [12] van De Plassche K.L. et al 202 Fast modeling of turbulent transport in fusion plasmas using neural networks *Phys. Plasmas* **27** 1 17 022310
- [13] van de Plassche K.L., Citrin J., Bourdelle C., Camenen Y., Casson F.J., Dagnelie V.I., Felici F., Ho A. and Van Mulders S. 2020 *Phys. Plasmas* **27** 022310
- [14] Felici F., Citrin J., Teplukhina A.A., Redondo J., Bourdelle C., Imbeaux F. and Sauter O. 2018 *Nucl. Fusion* **58** 096006
- [15] Boyer M. et al 2019 *Nucl. Fusion* **59** 056008
- [16] Weiland M., Bilato R., Dux R., Geiger B., Lebschy A., Felici F., Fischer R., Rittich D. and van Zeeland M. 2018 *Nucl. Fusion* **58** 082032
- [17] Reich M., Bilato R., Mszanowski U., Poli E., Rapson C., Stober J., Volpe F. and Zille R. 2015 *Fusion Eng. Des.* **100** 73–80
- [18] Menighini O. et al 2017 *Nucl. Fusion* **57** 086034
- [19] Menighini O. et al 2020 *Nucl. Fusion* **61** 026006
- [20] Dux R. and Peeters A.G. 2000 *Nucl. Fusion* **40** 1721–9
- [21] Snyder P.B. et al 2012 *Phys. Plasmas* **19** 056115
- [22] Taroni A. and Cenacchi G. 1994 Jetto a free boundary plasma transport code (https://inis.iaea.org/search/search.aspx?orig_q=RN:19097143) (International Nuclear Information System)
- [23] Breton S. et al 2018 *Nucl. Fusion* **58** 096003
- [24] Rattá G.A., Vega J., Murari A., Vagliasindi G., Johnson M.F. and de Vries P.C. 2010 *Nucl. Fusion* **50** 025005
- [25] Zhang Y., Pautasso G., Kardaun O., Tardini G. and Zhang X.D. 2011 *Nucl. Fusion* **51** 063039
- [26] Murari A., Vega J., Rattá G.A., Vagliasindi G., Johnson M.F. and Hong S.H. 2009 *Nucl. Fusion* **49** 055028
- [27] Rea C. et al 2019 *Nucl. Fusion* **59** 059601

- [28] Fu Y. *et al* 2020 *Phys. Plasmas* **27** 022501
- [29] Hernandez J.V., Vannucci A., Tajima T., Lin Z., Horton W. and McCool S.C. 1996 *Nucl. Fusion* **36** 1009
- [30] Wroblewski D., Jahns G.L. and Leuer J.A. 1997 *Nucl. Fusion* **37** 725
- [31] Pautasso G. *et al* 2001 *J. Nucl. Mater.* **290–293** 1045–51
- [32] Cannas B., Fanni A., Sonato P. and Zedda M.K. 2007 *Nucl. Fusion* **47** 1559
- [33] Yoshino R. 2003 *Nucl. Fusion* **43** 1771
- [34] Sengupta A. and Ranjan P. 2000 *Nucl. Fusion* **40** 1993
- [35] Kates-Harbeck J., Svyatkovskiy A. and Tang W. 2019 *Nature* **568** 526–31
- [36] Stillerman J.A., Fredian T.W., Klare K.A. and Manduchi G. 1997 *Rev. Sci. Instrum.* **68** 939–42
- [37] Boyer M.D. *et al* 2019 *Nucl. Fusion* **59** 059601
- [38] Laggner F.M. *et al* 2020 *Nucl. Fusion* **60** 076004
- [39] Smith S. Recent improvements to automatic profile fitting (ZIP-FIT) (2010) (Accessed: 20 November 2020)
- [40] Lao L.L., John H.E.S., Peng Q., Ferron J.R., Strait E.J., Taylor T.S., Meyer W.H., Zhang C. and You K.I. 2005 *Fusion Sci. Technol.* **48** 968–77
- [41] Ronneberger O., Fischer P. and Brox T. 2015 U-net: Convolutional networks for biomedical image segmentation *Int. Conf. on Medical Image Computing and Computer-Assisted Intervention Munich* 234–41 (https://link.springer.com/chapter/10.1007/978-3-319-24574-4_28)
- [42] Krizhevsky A., Sutskever I., Hinton G. 2012 ImageNet classification with deep convolutional neural networks *Advances in Neural Information Processing Systems* Lake Tahoe 1097–105
- [43] Gers F.A. *et al* 1999 Learning to forget: continual prediction with LSTM *International Conference on Artificial Neural Networks* 1999 (17 February 2021), Edinburgh (<https://ieeexplore.ieee.org/document/818041/>)
- [44] Glorot X. *et al* 2011 Deep sparse rectifier neural networks *Proc. 14th Int. Conf. on Artificial Intelligence and Statistics* Ft. Lauderdale pp 315–23
- [45] Nair V., Hinton G. 2010 Rectified linear units improve restricted Boltzmann machines *Proceedings of the 27th Int. Conf. on Machine Learning Haifa* 807–14
- [46] Nwankpa C. *et al* 2018 arXiv:1811.03378
- [47] Chollet F. *et al* 2015 Keras (<https://keras.io>)
- [48] Duchi J.C. *et al* 2011 *J. Mach. Learn. Res.* **12** 2121–59
- [49] Li Y., Yuan Y. 2017 Convergence analysis of two-layer neural networks with relu activation *Advances in Neural Information Processing Systems Long Beach* 30 597–607
- [50] Clevert D.A., Unterthiner T. and Hochreiter S. 2016 Fast and accurate deep network learning by exponential linear units (ELUs) *Int. Conf. on Learning Representations* San Juan 1–14 arXiv:1511.07289
- [51] Klambauer G. *et al* 2017 *Advances in Neural Information Processing Systems* (December 2017) 972–81 *NeurIPS Proceedings* (17 February 2021)
- [52] Karlik B. and Olgac A. V. 2011 Performance analysis of various activation functions in generalized MLP architectures of neural networks *Int. J. Artif. Intell. Expert Syst.* **1** –22
- [53] Xing Z. *et al* 2020 *Fusion Eng. Des.* **163** 112163