CCFE
CULHAM CENTRE FOR FUSION ENERGY

UK Atomic Energy Authority

UKAEA-CCFE-PR(20)122

K.L. van de Plassche, J. Citrin, C. Bourdelle, Y. Camenen, F. J. Casson, V.I. Dagnelie, F. Felici, A. Ho, JET Contributors

Fast modelling of turbulent transport in fusion plasmas using neural networks

Fast modelling of turbulent transport in fusion plasmas using neural networks

K.L. van de Plassche, J. Citrin, C. Bourdelle, Y. Camenen, F. J. Casson, V.I. Dagnelie, F. Felici, A. Ho, JET Contributors

This is a preprint of a paper submitted for publication in Physics of Plasmas

Fast modelling of turbulent transport in fusion plasmas using neural networks

K.L. van de Plassche,[1, a)] J. Citrin,[1] C. Bourdelle,[2] Y. Camenen,[3] F. J. Casson,[4] V.I. Dagnelie,[1, 5] F. Felici,[6] A. Ho,[1] and JET Contributors[7]

[1)] *DIFFER, PO Box 6336, 5600 HH Eindhoven, The Netherlands*
[2)] *CEA, IRFM, F-13108 Saint-Paul-lez-Durance, France*
[3)] *CNRS, Aix-Marseille Univ., PIIM UMR7345, Marseille, France*
[4)] *CCFE, Culham Science Centre, Abingdon, UK*
[5)] *ITP, Utrecht University, Princetonplein 5, 3584 CC Utrecht, The Netherlands*
[6)] *EPFL-SPC, CH-1015, Lausanne, Switzerland*
[7)] *See the author list of E. Joffrin et al. accepted for publication in Nuclear Fusion Special issue 2019, https://doi.org/10.1088/1741-4326/ab2276*

(Dated: 6 September 2019)

We present an ultrafast neural network (NN) turbulent tokamak transport model, QLKNN, for heat and particle fluxes. QLKNN is a surrogate model based on a database of $3 \cdot 10^8$ flux calculations of the quasilinear gyrokinetic transport model QuaLiKiz. To ensure accurate reproduction of the underlying model, we include known features of the physical system by choosing specific training targets and using a customized cost function in our training pipeline. We have coupled QLKNN to the tokamak modelling framework JINTRAC and rapid control-oriented tokamak transport solver RAPTOR. We demonstrate and validate the coupled frameworks through application to three JET shots covering a representative spread of H-mode operating space, predicting turbulent transport in the plasma core region ($0.2 < \rho_{N,tor} < 0.85$). QLKNN is able to accurately reproduce QuaLiKiz-predicted kinetic profiles ($T_{i,e}$ and n_e) but orders of magnitude faster, from 7 days on 16 cores (JINTRAC-QuaLiKiz) to 20 minutes on 2 cores (JINTRAC-QLKNN) and 90 seconds on 1 core (RAPTOR-QLKNN). The discrepancy between QLKNN and QuaLiKiz is only on the order 1%-10% in rotationless cases. The impact of rotation on turbulent fluxes is included in QLKNN through a new flux scaling rule in postprocessing, based on a set of linear gyrokinetic simulations. This difference from the native QuaLiKiz rotation rule results in slightly larger (3%-15%) differences in the final kinetic profiles for cases including rotation. Dynamic behaviour is also well captured by QLKNN, with differences of only 4%-10% compared to full QuaLiKiz observed at mid-radius, for a study of density buildup following the LH transition. Deployment of neural network surrogate models in multi-physics integrated tokamak modelling is a promising route towards enabling accurate and fast tokamak scenario optimization, Uncertainty Quantification, and control-oriented applications.

I. INTRODUCTION

Accurate prediction of tokamak core plasma temperature and density is essential for interpretation of current-day fusion experiments, designing future devices, and optimization of plasma scenarios. Time-evolved tokamak simulation on discharge timescales is typically carried out within a 'integrated modelling' approach[1], where multiple models representing various physics calculations are coupled together within a single code or workflow. An essential component of integrated models is the prediction of turbulent transport fluxes, particularly in the tokamak core where transport is often dominated by plasma microinstabilities[2,3]. However, calculating these fluxes using first-principle-based nonlinear gyrokinetic models within an integrated modelling framework – while feasible[4] – is too computationally expensive for routine simulation.

Reduced order turbulence models have thus been developed for increased tractability. They remain first-principle-based yet computationally cheaper through invoking the quasilinear approximation. Quasilinear turbulence models like QuaLiKiz[5–7] and TGLF[8] are valid in extensive parameter regimes in the tokamak core. These models can predict turbulent transport fluxes approximately 6 orders of magnitude faster than δf local nonlinear codes. For QuaLiKiz, this means around 10-100 seconds on a single CPU, depending on the physics fidelity used in the simulation. The speed has enabled routine runs of QuaLiKiz coupled to integrated modelling suites such as JINTRAC[9,10], recently leading to numerous successful validation exercises against JET[5,11,12] and AUG[13] discharges. However, these simulations still can take days to run, parallelized on 16 cores. This sets limits on large-scale model validation and theory-based optimization of fusion experiments, as well as for model-based real-time control applications.

To further accelerate integrated modelling workflows we apply feed-forward neural networks (FFNNs) as a surrogate model, reproducing the underlying turbulent transport model within tens of microseconds. The concept rests on taking advantage of the fast evaluation time of the reduced tokamak turbulence models (e.g. QuaLiKiz), and applying them for generating large training sets then used for neural network regression. The neural networks can then be used as a drop-in replacement inside the integrated model, removing one of the main computational bottlenecks.

Similar development of neural network surrogates for physics models applied within tokamak integrated modelling has been carried out for: the TGLF quasilinear turbulent transport model[14], the EPED pedestal confinement model[14] and the neutral beam heating code NUBEAM[15–17]. This paper represents the state-of-the-art of the QuaLiKiz neural network sur-

[a)] k.l.vandeplassche@differ.nl

rogate model, far beyond our original proof-of-principle[18,19]. The input dimensionality is increased from 4 to 10, leading to increased surrogate model fidelity. Furthermore, particle transport is now included.

Other novel aspects include the large amount of samples in the training dataset (discussed in section III), and the focus on including physics-based features in the training pipeline (discussed in sections IV and V). To properly introduce the applied methodology, we summarize neural network techniques in section II. Finally, we show the application of the neural network surrogate model within integrated modelling frameworks in section VII.

II. NEURAL NETWORKS

Neural networks are universal approximators and hence a powerful tool for regression[20]. In this work we apply fully connected feed-forward neural networks to a supervised regression problem, in which we reproduce the input-output mapping of the QuaLiKiz code. The basic building block of a FFNN is the *neuron*, which takes multiple inputs, multiplies each input with a weight w_i, sums the results, adds a bias b and applies a non-linear *activation function* f, as shown in figure 1. In a FFNN neurons are distributed into layers, with

$$y = f\left(\sum_{i=0}^{n} w_i x_i + b\right)$$

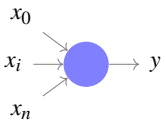

FIG. 1. Schematic and mathematical representation of a neuron.

each neuron in a layer taking the output of each neuron in previous layer as input. Most FFNNs have at least an *input layer* in this case taking the physical input *features* described in Section III, a *hidden layer* capturing the to-be learned hidden relationships and an output layer, combining the learned relationships into a *target*. A FFNN with a single hidden layer is able to reproduce any sufficiently smooth input-output mapping up to arbitrary error[21], but in practice training a network with at least two hidden layers has better convergence properties. The equation for a two-hidden-layer neural network is shown in equation 2. We use the notation of Ref.[22], w^l_{jk} for the weight of the connection from the k^{th} neuron in the $(l-1)^{\text{th}}$ layer to the j^{th} neuron in the l^{th} layer. Then, b^l_j is the bias of the j^{th} neuron in the l^{th} layer. Then the *activation* (i.e. "input of") the j^{th} neuron in the l^{th} layer with activation function f is simply:

$$a^l_j = f\left(\sum_k w^l_{jk} a^{l-1}_k + b^l_j\right) \tag{1}$$

For example we show the explicit formula for a 2-hidden layer neural network with N-dimensional input x_{in} and M-dimensional output y in Equation 2. The output layer has a linear activation function which is simply the identity function $f(x) = I(x) = x$, as is usual for regression problems. We also assume each hidden layer has the same non-linear activation function σ. A schematic representation of this neural network can be found in Figure 2.

$$y = a^3_1 = \sum_i^M w^3_i \sigma\left(\sum_j w^2_{ij}\sigma\left(\sum_k^N w^1_{jk} x_{in,k} + b^1_j\right) + b^2_i\right) + b^3_o \tag{2}$$

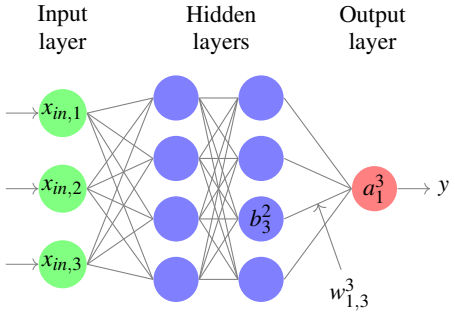

FIG. 2. A schematic representation of a two-hidden-layer feed-forward neural network.

The weights and biases of the network are determined by minimizing some cost function C, called *training*. Assuming we have A input-output mapping *samples*, we collect these in an $M \times A$ *input matrix* and $N \times A$ *output matrix*. Before training the full dataset is generally split in a *training set*, which is used to update the weights and biases, a *validation set* which is used to check generalization of the neural network model every *epoch* and a *test set* which is never seen during training, and is used to check generalization across any tunable parameters related to the training process described later. The weights and biases are updated using an optimizer, usually a variant of (mini-batch) gradient descent[23]. For mini-batch gradient descent the training set is further split in batches of size B, which is itself an hyperparameter to be tweaked. A small batch size will generally be slower to converge, but the resulting model has a better generalizing properties[24]. A common choice for measure-of-goodness and regularizing term for regression tasks are the mean square error and L2-regularization respectively. Now we can write down a general formula for the cost function, extended in this work in Section IV B, where y_i is the network prediction for a single sample, and $\hat{y}_i$ the real value in the dataset:

$$C = C_{good} + \lambda_{regu} C_{regu} \tag{3}$$

$$C = \frac{1}{B}\sum_i^B (\hat{y}_i - y_i)^2 + \lambda_{regu} \|W\|_2^2 \tag{4}$$

Where $\|W\|_2$ denotes the matrix l2-norm of all the weights combined. The derivative of the cost function with respect to its tunable parameters can be analytically determined using the chain-rule in what is called *backpropagation*. This can then be used in the update of the gradient descent, generally of the form:

$$\theta_{n+1} = \theta_n - \gamma \nabla C(\theta_n) \tag{5}$$

where θ are the tunable parameters (w and b) and γ is the step size or *learning rate*, another hyperparameter to be optimized. The training algorithm needs an initial guess θ_0 to start training, which is in our case a random Gaussian distribution with mean 0 and standard deviation 1 for all weights and biases. The weights and biases are updated every batch B. After the optimizer has seen the full training set, this is called an epoch. The resulting neural network is then used to determine the loss against the full validation set, which is used to determine convergence. If convergence is reached, the training is stopped and the neural network saved. If not, all samples are re-shuffled and looped over until convergence is reached. In this work we use *early stopping* to determine convergence. Early stopping sets a bound on the amount of epochs the validation loss is allowed to increase, a hyperparameter called *patience*. Early stopping prevents overfitting and gives a robust stopping criterion.

This method of training is quick, even for a large amount of parameters, as $\nabla C(\theta_n)$ is analytical and efficient to calculate. It is thus also quick to calculate the derivatives of the final trained neural network with respect to its inputs $\mathrm{d}y/\mathrm{d}x$. This is highly useful for our application, when the neural network turbulent surrogate models are integrated into implicit PDE solvers (solving the transport equations) and/or used for trajectory control applications.

III. DATASET GENERATION

We use the quasilinear gyrokinetic transport model QuaLiKiz to generate a large database of turbulent transport model calculations. QuaLiKiz solves the linear gyrokinetic dispersion relation in the electrostatic limit in $s-\alpha$ geometry. By assuming a shifted Gaussian ansatz for the mode eigenfunctions in the strong ballooning limit, strongly trapped and passing particles, and a small Mach number, the calculation is greatly simplified leading to increased calculation speed ($\times 10^3$) beyond standard linear gyrokinetics. The linear responses at the eigenfrequencies are then used to set the transport fluxes (heat, particle, and momentum), in conjunction with a *saturation rule* for the electrostatic potential amplitudes and spectral shape, tuned to non-linear gyrokinetic simulations both at ion-scales and electron-scales[6,25].

The input space of the full QuaLiKiz code ($\sim$15 dimensions for typical simulations) is too large to cover with a brute-force hypercube scan. We thus constrain the training set dimensionality to the subset most significantly impacting turbulent transport within the framework of QuaLiKiz approximations. These input dimensions include the logarithmic ion temperature gradient (R/L_{T_i}), electron temperature gradient (R/L_{T_e}), density gradient (R/L_n), ion-electron temperature ratio (T_i/T_e), safety factor (q), magnetic shear ($\hat{s}$), local inverse aspect ratio (r/R), collisionality (ν^*), and effective charge (Z_{eff}), with a carbon impurity and deuterium main ion. Notable simplifications are excluding plasma rotation ($\gamma_{E\times B} = v_{par} = v_{perp} = 0$), assuming equal density gradient for the two ion species, and no Shafranov shift. This significantly extends the previous proof-of-principle 4D neural network QuaLiKiz regression[18]. These nine inputs are taken as the feature space of the neural network. The impact of rotation, important for accurate tokamak plasma simulation, is taken into account through a new separate model in post-processing, as described in Section VI.

A database consisting of 3×10^8 QuaLiKiz input-flux relations was generated with HPC resources on the Edison supercomputer at NERSC, using 1.3 MCPUh. The database spans ion scales ($k_\theta \rho_s \leq 2$) and electron scales ($k_\theta \rho_s > 2$) and contains contributions to transport fluxes and coefficients q (heat), Γ (total particle), D (particle diffusivity), and V (particle convection) per species. The input space was chosen as a rectangular, non-uniform 9-dimensional grid. The bounds cover dimensionless parameter regimes typically encountered in the core of standard aspect-ratio present-day tokamaks, and future devices such as ITER and DEMO. We chose the spacing of the grid to be a higher density around typical threshold zones (e.g. $-\frac{R}{T_e}\frac{\mathrm{d}T_e}{\mathrm{d}r} \approx 5$) and zones of high non-monoticity (e.g. $\hat{s} \approx 0.7$) based on previous extensive experience with application of QuaLiKiz within integrated modelling and standalone. See Table I for the bounds of the generated dataset.

To aid with successful neural network regression, as discussed in the subsequent sections, QuaLiKiz was modified to additionally output fluxes and transport coefficients arising solely from individual classes of modes, i.e. ITG, TEM, ETG. Mode identification is determined by mode number (ion or electron scale) and mode frequency (ion or electron direction). The ETG electron heat flux is defined as the q_e arising from the spectrum $k_\theta \rho_s > 2$. To separate ITG and TEM fluxes, the saturation rule was evaluated twice for electron modes and ion modes separately, at ion-scales. This can lead to inconsistencies compared to combining all ion-scale modes together in the saturation rule. However, in practice, the difference between summing the separate ITG and TEM fluxes together (in cases where they coexist in the spectrum) compared to their self-consistent total evaluation in the saturation rule, is typically less than 20%. To further extend the general applicability of the neural networks, we use a form of GyroBohm normalization for all transport coefficients in this work, as defined in Equations 6-11

$$c_{GB} \equiv \frac{\sqrt{A_{i,0} m_p} T_e^{1.5}}{q_e^2 B_0^2 a} \tag{6}$$

$$\Gamma_{GB} \equiv \frac{a}{n_s c_{GB}} \Gamma_{SI} \tag{7}$$

$$D_{GB} \equiv \frac{1}{c_{GB}} D_{SI} \tag{8}$$

$$V_{GB} \equiv \frac{a}{c_{GB}} V_{SI} \tag{9}$$

$$q_{GB} \equiv \frac{a}{n_s T_s c_{GB}} q_{SI} \tag{10}$$

$$\chi_{GB} \equiv \frac{1}{c_{GB}} \chi_{SI} \tag{11}$$

a and R are the midplane-averaged minor and major radii of the last-closed-flux-surface. Unless noted otherwise, all radial derivatives are against the midplane-averaged minor radius $r \equiv r_{minor}$. For convenience, we define the normalized

TABLE I. 9D hyperrectangle bounds and number of points of the QuaLiKiz neural network training set. Each input is non-uniformly distributed in space, with a finer resolution in experimentally more relevant regimes.

variable	# points	min	max
$k_\theta \rho_s$	18	0.1	36
R/L_{T_e}	12	0	14
R/L_{T_i}	12	0	14
R/L_n	12	-5	6
q	10	0.66	15
$\hat{s}$	10	-1	5
r/R	8	0.03	0.33
T_i/T_e	7	0.25	2.5
ν^*	6	1×10^{-5}	1
Z_{eff}	5	1	3
Total	3×10^8	≈ 1.3 MCPUh	

length scales:

$$L_{T_{i,e}} \equiv -T_{i,e}\left(\frac{\mathrm{d}T_{i,e}}{\mathrm{d}r}\right)^{-1} \tag{12}$$

$$L_n \equiv -n\left(\frac{\mathrm{d}n}{\mathrm{d}r}\right)^{-1} \tag{13}$$

the normalized collision frequency:

$$\nu^* \equiv \nu_e^* \equiv \nu_e \tau_{bounce} \tag{14}$$

$$\nu_e \equiv 917.4 Z_{eff}(10^{-19} n_e)\Lambda_e (10^3 T_e)^{-1.5} \tag{15}$$

$$\Lambda_e \equiv 15.2 - 0.5\ln(10^{-20} n_e) + \ln(10^3 T_e) \tag{16}$$

$$\tau_{bounce} \equiv \frac{qR}{(\frac{r}{R})^{1.5}\sqrt{\frac{q_e}{m_e}T_e}} \tag{17}$$

where q_e and m_e are the electron charge and mass respectively, and finally the effective ion charge Z_{eff}:

$$Z_{eff} \equiv \frac{\sum_i n_i Z_i^2}{\sum_i n_i Z_i} = \frac{\sum_i n_i Z_i^2}{n_e} \tag{18}$$

IV. PHYSICS-BASED NEURAL NETWORK TRAINING

Regularized neural networks provide a smooth regression of supplied training data. It does not assume any features of the underlying mapping. Physics-informed features can be directly implemented into the training methodology to significantly improve the fidelity of the surrogate transport model. For our application, we desire the following features:

- sharp flux discontinuities at critical (temperature) gradients of the underlying instabilities
- identical critical (temperature) gradient for all transport channels driven by a single (TEM/ITG) instability

This was found essential for consistent results in integrated modelling.

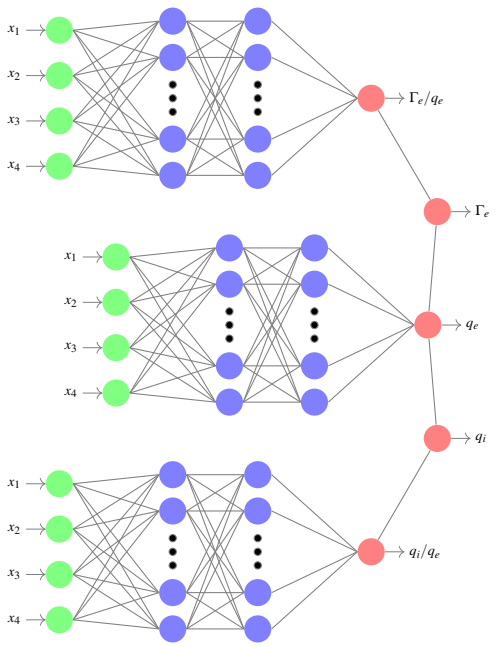

FIG. 3. Schematic overview of the Γ_e and q_e predicted by a combined leading flux and ratio-predicting neural network. Three separate FFNNs, one predicting the leading flux q_e and two ratio-predicting network predicting Γ_e/q_e and q_i/q_e are combined to a network ensemble that predicts Γ_e, q_e, and q_i.

A. Training targets

The identical critical thresholds for all transport channels was forced by a careful choice of training targets. The transport coefficients were separated into a *leading flux* and *flux ratios*. For example, for TEM fluxes, the leading flux is the electron heat flux q_e, resulting in the flux ratios q_i/q_e and Γ_e/q_e. Networks are then trained on the leading flux and flux ratios separately, resulting in a leading flux network, and flux ratio networks. In the transport model implementation, the flux ratio predictions and leading flux predictions are multiplied together to re-obtain the original transport fluxes q_i and Γ_e. This procedure is sketched in Figure 3. The fact that the leading flux is zero in the stable region (below the critical threshold), guarantees that the thresholds of all transport channels are identical. Increased smoothness and quality in the regression is achieved by removing training set outliers through data filtering (see section IV C).

Splitting the training targets by mode (ITG, TEM, ETG) was found important for obtaining flux ratio regressions of sufficient quality. Flux ratio network training for total fluxes (i.e, corresponding to the original QuaLiKiz output, as opposed to each of the separated ITG, TEM, ETG flux outputs) was unable to converge to a result of sufficient quality for a robust surrogate turbulence model, even after extensive hyperparameter scans. This is likely due to sharp discontinuities present in the flux ratios when not separating the fluxes. This is apparent in a TEM-ITG transition, for example in a scan of R/L_{T_i} as shown in Figure 4. The boundary be-

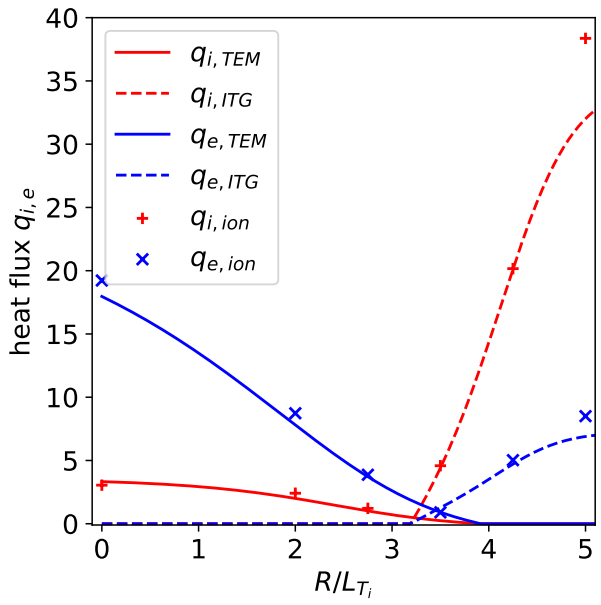

FIG. 4. The QuaLiKiz predicted total heat fluxes for electrons q_e and ions q_i for multiple values of $\frac{R}{L_{T_i}}$, while keeping the other input parameters constant (pluses and crosses). We also show neural networks fit with the methodology described in IV. Important to note is the capture of sharp transport characteristics around $-\frac{R}{T_i}\frac{dT_i}{dr} \approx 3.1$. Note the excellent quality of regression throughout. The discrepancy for the highest $\frac{R}{L_{T_i}}$ point is due to it being filtered out of the training set due to non-experimentally-relevant high flux, see section IV C

tween ITG and TEM regimes for this specific parameter set is $-\frac{R}{T_i}\frac{dT_i}{dr} \approx 3.1$. Above this value (ITG regime), $q_i/q_e > 1$. Below this value (TEM regime), $q_i/q_e \ll 1$. The transition between these regimes is extremely sharp, a feature challenging to capture by a regularized neural network. Instead, we use the mode-specific fluxes calculated by QuaLiKiz described in Section III, where the mode-specific flux ratios within the ITG and TEM regimes shows significantly less structure. Neural networks are fit for each mode separately. Then, in the transport model implementation the per-mode fluxes are added together in postprocessing using an unweighted sum. Fitting the separate modes results in clearer thresholds without transition regions, enabling the use of the modified cost function in IV B, resulting in a sharper transition at the threshold. As seen in figure 4, the neural networks fits (solid and dashed lines) from the combined ITG + TEM networks accurately reproduces the non-trivial structure of the ITG-TEM transition.

B. Customized cost function

Training a neural network means optimizing the weights and biases of the network to minimize a cost function C, which typically compares for each set of inputs, the neural network output to desired targets - in our case the QuaLiKiz input-output mapping. Typically the cost function consists of a measure of goodness-of-fit, and a regularizing term, as already shown in Eq. 3. We have customized the cost function for our application beyond this standard implementation, to impose prior physics knowledge of the mapping structure into the system. This prior knowledge consists of: sharp instability thresholds, zero flux in the region where no instabilities are predicted, and identical transport flux thresholds for all transport channels. This last point has been treated through the leading-flux and flux-ratio paradigm introduced in section IV A. We now summarize the other two.

The sharpness of the critical threshold is achieved by only including the unstable points (where instabilities are predicted) in the measure of goodness-of-fit C_{good} for the leading flux regression. Otherwise, if including the zero flux points explicitly, then due to regularization some smoothing at the discontinuous critical threshold region is inevitable, leading to a loss of accuracy in the regression. By only including the unstable points, the leading flux neural network predictions are free to extrapolate to negative fluxes below the critical threshold, which are then clipped to zero in the transport model implementation, leading to the desired sharp critical threshold behaviour for all transport channels.

We then wish to avoid any possible FFNN extrapolation to spurious non-zero fluxes in the stable region below critical threshold. This is done by controlling the allowed range of extrapolation in the stable region. We add an additional penalty term C_{stab} in the cost function for the leading flux regression, for samples predicted to be stable by QuaLiKiz. This penalty term punishes positive FFNN predictions in ostensibly stable regions, while remaining zero for negative FFNN predictions in the stable region (which are then subsequently clipped to zero).

The customized cost function is summarized in Eq.19. The free parameters λ_{regu}, λ_{stab}, and C_{stab}, as well as other hyperparameters like network topology, are then optimized using a simple grid search. To test generalization, the dataset is split in a test set of 5% never seen during training, and a validation set of 5% used during training to avoid overfitting on training data. The remainder is used as training set. So, for each network prediction NN_i relating to a QuaLiKiz calculation QLK_i we have for all n samples and k weights:

$$C = C_{good} + \lambda_{regu} C_{regu} + \lambda_{stab} C_{stab} \tag{19}$$

$$C_{good} = \begin{cases} \frac{1}{n}\sum_{i=1}^{n} (QLK_i - NN_i)^2, & \text{if } QLK_i \neq 0 \\ 0, & \text{if } QLK_i = 0 \end{cases} \tag{20}$$

$$C_{regu} = \sum_{i=1}^{k} w_i^2 \tag{21}$$

$$C_{stab} = \begin{cases} 0, & \text{if } QLK_i \neq 0 \\ \frac{1}{n}\sum_{i=1}^{n} NN_i - c_{stab}, & \text{if } QLK_i = 0 \end{cases} \tag{22}$$

C. Training set filtering

Inaccurate data in the training set can have a deleterious impact on the neural network training by overly biasing the regression towards an inaccurate representation. Such inaccuracies can arise due to unexplored corners in parameter space present in the QuaLiKiz scan, outside the commonly used (and experimentally relevant) parameter regimes of the code. While several code improvements were already made for some of these regimes on a case by case basis, surveyal by eye of the entire dataset was not feasible due to data size. Therefore a conservative approach was taken in filtering the training set to remove untrusted QuaLiKiz flux calculations. As the dataset is too large for memory, dask framework[26] was used to allow for general out-of-core processing of arbitrarily large array-like structures. For the networks trained in this work, data points were deemed invalid and not included in training, according to the following heuristic criteria:

- Difference between total particle flux and derived particle flux from diffusion and convection transport coefficients is more than 50%, i.e. $\left|\frac{\Gamma_s-(-D_s dn_s/dr+V_s n_s)}{\Gamma_s}\right| < 0.5$
- Difference between unweighted sum of ITG + TEM mode contributions, and self-consistent total flux calculation, was more than 50%
- Ambipolarity was violated by more than 50%
- Any transport coefficient is non-zero but predicted to be smaller than 10^{-4} in GyroBohm (GB) units
- Outliers in the flux-ratio distributions were removed by visual examination of the data histograms and determining cut-off points corresponding to tails of the distributions. These were determined as:

$$0.05 < q_{e,ITG}/q_{i,ITG} < 1.5 \tag{23}$$
$$0.02 < \Gamma_{e,ITG}/q_{i,ITG} < 0.6 \tag{24}$$
$$0.05 < q_{i,TEM}/q_{e,TEM} < 2.0 \tag{25}$$
$$0.03 < \Gamma_{e,TEM}/q_{e,TEM} < 0.8 \tag{26}$$

  Where s is the electron e or ion i species, and x is the transport coefficient of interest, e.g. Γ or q.

In addition, to restrict the training set to a more experimentally relevant regime, all points with either total ion or electron energy fluxes larger than 33 (in GB units) were removed, which is far beyond typically encountered in core plasmas.

D. Measures of goodness

Performance indicators are a critical tool for differentiating the quality of different neural networks, trained with different hyperparameters, to assess the optimal networks to use in our application. In contrast to classical regression tasks, the final loss is not the key performance indicator of the trained network. Instead, for our application, how well the trained neural network performs as a transport model within integrated transport modelling is the most important. However, using the integrated model directly in the training pipeline is cumbersome and computationally expensive. Instead, we define metrics relating to the aforementioned capture of known physical features and use these in conjunction with the classical test loss to judge the quality of the trained networks after training. To do this we take 1D slices in the main driving gradient (for each mode) from the full dataset and let the network predict over the range of this slice. The main driving gradients are taken to be the electron temperature gradient for TEM and ETG, and the ion temperature gradient for ITG. The full dataset contains 2.4×10^7 such slices, but taking 5% was sufficient to statistically differentiate networks with different hyperparameters. We first define for each slice:

- The *neural network critical gradient* $c_{NN,crit}$. This is the location where the neural network leading flux predictions cross from positive to negative fluxes corresponding to the transition from unstable to stable QuaLiKiz regions.
- The *spurious stable prediction* c_{spur}. This is first encountered point in the QuaLiKiz stable region, when descending from high gradient to low gradient, where the neural network predictions spuriously transitions from negative flux (clipped to zero in the transport code implementation) to positive flux
- The *QuaLiKiz critical gradient proxy* $c_{QLK,crit}$. This is taken as the midpoint between the gradient slice gridpoints corresponding to the transition from zero to positive fluxes in the original QuaLiKiz data

Using these quantities, we found the following measures to be important:

- The percentage of slices where QuaLiKiz predicts a threshold, and QLKNN does not
- The percentage of slices where QLKNN predicts spurious flux in the stable region
- The mean absolute distance between the QuaLiKiz and QLKNN thresholds $\frac{1}{n}\sum_i^n |c_{NN,crit} - c_{QLK,crit}|$
- The smoothness in the unstable zone as defined from the second derivative with respect to the driving gradient: $\frac{1}{n}\sum_i^n |\frac{\partial^2 x}{\partial (R/LT_s)^2}|$, if $R/LT_s > c_{NN,crit}$. This strongly depends on the regularisation hyperparameter.
- The relative distance of spurious stable flux prediction and predicted threshold $\frac{c_{NN,crit}-c_{spur}}{c_{NN,crit}}$

An overview of these distances is shown in Figure 5. A trained network never has an absolute minimum in all these metrics simultaneously, so instead a trade-off is made. In this work we have not attempted to unify these metrics in a single value. Instead, the metrics are used as guidance to select a small number of networks that are then tested inside the integrated model. The metrics for the final implemented networks can be

TABLE II. An overview of the measures of goodness as described in Section IV D: The percentage of slices with no threshold (No thresh frac) and without spurious flux predictions (No spurious frac), the absolute threshold mismatch (Abs thresh mismatch), relative spurious flux prediction distance (Rel spurious dist), and smoothness in the unstable zone (Unstab zone smooth). These quantities are shown for the three leading flux networks $q_{e,ETG}$, $q_{e,TEM}$, and $q_{i,ITG}$. These statistics were taken on a reduced 7D dataset, fixing $Z_{eff} = 1$ and $\nu^* = 1e-3$. No attempt is made to combine these measures of goodness into a single final value, nor is currently known what the upper and lower bounds are. However, as shown later in this work, these values were found sufficient for good model performance in integrated modelling.

	No thresh frac [%]	No spurious frac [%]	Abs thresh mismatch	Rel spurious dist [%]	Unstab zone smooth
$q_{e,ETG}$	3.3	97.7	-0.38	-0.44	0.017
$q_{e,TEM}$	14.3	98.6	-0.31	-0.70	0.008
$q_{i,ITG}$	4.2	99.2	-0.26	-0.52	0.0300

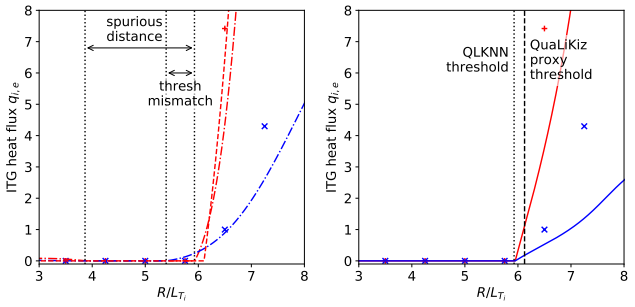

FIG. 5. Predictions of the ITG driven heat flux for the ions (red) and electrons (blue). We show three types of networks, networks trained using a standard RMS error on both the stable and unstable points (dash-dot, left) and only on the unstable points (dashed, left). These networks show clear mismatch between transport channels and QuaLiKiz prediction, as well as prediction of fluxes in the stable region. The physics-based neural network (right) have no mismatch between transport channels, a sharper threshold closer to the QuaLiKiz prediction, as well as no prediction of flux in the stable region.

found in Table II. All these metrics are identical for the leading flux network and their associated ratio network because of the choice of training targets described in Section IV A. Finding minimum required values of the metrics is outside the scope of this work, but we note the low percentage of stable flux predictions for all networks, and the low values for threshold mismatch. Because of the low percentage of stable flux predictions, the relative spurious distance is thought to be of less importance, while the smoothness was assumed to be sufficient by visual inspection of many neural network predictions on random slices. Finally, while the measures of goodness for the TEM networks are not as good as the others, we found it encouraging enough to implement them in the later-described transport models. However, for future work improving these networks specifically would be beneficial.

V. TRAINING PIPELINE

The networks were trained using the TensorFlow[27] framework. TensorFlow is an open source framework allowing various machine learning algorithms to be run efficiently on heterogeneous machines, both for CPU and for GPU architectures. The framework can be used to train neural networks out-of-the-box, but as a general framework care has to be taken that the use-case it is applied to matches the expectations of the framework. In this work, we have identified and worked around two limitations of the TensorFlow framework at time of writing. Firstly, TensorFlow is most commonly used for deep learning. In deep learning, the amount of training samples versus the size of the network, and thus its evaluation speed, is relatively small. In this work, the networks are shallow and the amount of training samples large. As such, we have implemented a simple but factor two quicker shuffling algorithm using numpy[28]. This results in 1.25x (CPU) to 2x (Tesla P100 GPU) reduction of training time. Secondly, TensorFlow uses its own proprietary format to save the trained neural network weights and biases to disk. This would mean any integrated framework would need to depend on TensorFlow/python to use the neural network predictions. This is inconvenient and non-performant for many integrated frameworks, especially if they are in MATLAB (RAPTOR) and Fortran (JINTRAC). Instead, we wrote a lightweight communication format using JSON between TensorFlow, and an implementation in Fortran with wrappers for Python and MATLAB. Using these MKL accelerated native Fortran functions a network with 3 layers and 128 neurons can be evaluated within 60 µs on a single core or 100 ms if the derivatives of the neural network output with respect to the neural network inputs are also evaluated. These wrappers are freely available at GitLab[29].

Neural network training involves optimizing training hyperparameters. While many algorithms to optimize hyperparameters exist, the authors are not aware of a commonly used readily-available framework to do this. As such, we have written a thin wrapper around TensorFlow, using a PostgreSQL database and Spotify's Luigi framework[30] to interact with supercompute job schedulers and train, validate and analyse networks trained with the QLKNN training framework. This allows the user to set up simple hypergrid hyperparameter scans. For the dataset used in this work we have found optimal hyperparameters which work well for a dataset of reduced 7D space (fixing $Z_{eff} = 1$ and $\nu^* = 1e-3$), also work sufficiently well for networks trained on the full 9D space. Additionally, hyperparameters optimal for ITG neural networks were found to work well for networks for the other modes in exploratory hyperparameter scans. Using this property, we have done wide scans of the following hyperparameters for $q_{i,ITG}$, resulting in over 1000 trained neural networks:

- number of layers
- nodes per layer
- goodness of fit
  - RMSE

- regularization
  - L2
  - early stopping patience
- loss function weights
  - cost L2 scale λ_{regu}
  - cost stable positive scale λ_{stab}
  - cost stable positive barrier C_{stab}
- Validation fraction (the test fraction was kept constant)

This resulted in the following optimal hyperparameters: RMSE as measure of goodness, L2 and early stopping with 15 patience as regularization and overfitting prevention. The networks were trained with the ADAM algorithm[31]. Then, for each leading flux a small scan over number of layers, nodes per layer, λ_{regu}, and λ_{stab} was carried out. This resulted in optimal settings of 3 layers, 128 nodes, 1e-5 and 1e-3 respectively. Finally, all networks were trained for the 9D set with a small hyperparameter scan around the found optimal 7D value, giving the same optimal value. The final networks are freely available on GitLab[32].

VI. ROTATION RULE

To save computation time, the dataset was ran without rotation. Beyond adding additional dimensions in training set input-space with associated cost, running QuaLiKiz with rotation takes $\times 4$ more computation time due to a loss of symmetry in the internal integration routines. However, since the impact of rotation on confinement can be critical, particularly in high performance H-modes, we implemented a new flux suppression rule in postprocessing. This rule is based on a new set of linear GENE[33] scans around the GA-Standard case, coupled to a methodology to assess the impact of rotation on linear growth-rates in spite of the Floquet fluctuations[34,35]. These scans consisted of toroidal rotation scans for various q, $\varepsilon \equiv \frac{r}{R}$, and $\hat{s}$, capturing both the effects of $E \times B$ stabilisation and Parallel Velocity Gradient (PVG) destabilization. The rule scales all ion-scale fluxes with a tuned function $f_{rot}(q, \hat{s}, \varepsilon)$. It depends also on the rotationless maximum ion-scale growth rate γ_0, which is predicted by an additional neural network based on the HPC-generated QuaLiKiz database, and the normalized $E \times B$ shearing rate $\gamma_{E \times B}$ defined in Equation 27.

$$\gamma_{E \times B} \equiv -\frac{\mathrm{d}v_{perp}}{\mathrm{d}r} \frac{R}{c_{ref}} \tag{27}$$

$$c_{ref} \equiv \sqrt{\frac{T_{ref}}{m_p}} \tag{28}$$

Where v_{perp} is the perpendicular velocity, T_{ref} is a reference temperature of 1 keV, and m_p is the proton mass. The TEM/ITG ion i and electron e transport coefficient x is then scaled with f_{rot} as described in Equation 29.

$$f_{rotrule} = c_1 q + c_2 \hat{s} + c_3/\varepsilon - c_4 \tag{29}$$

$$f_{rot} = \max(1 + f_{rotrule}\gamma_E/\gamma_0, 0) \tag{30}$$

$$x_{i/e,ITG/TEM} = f_{rot} * x_{i/e,ITG/TEM} \tag{31}$$

Where the values of the constants were determined to be $c_1 = 0.13$, $c_2 = 0.41$, $c_3 = 0.09$, and $c_4 = 1.65$. Using this rule, we are able to capture partially the effect of rotation on transport in a computationally quick way.

VII. APPLICATION IN TRANSPORT CODES

Tokamak transport, here confined to the 1D (radial) evolution of plasma density, temperature, and angular momentum in the plasma core, is governed by a highly non-linear coupled system of partial differential equations. Generally this system is too complicated to solve fully explicitly coupled from first principles with direct numerical simulation, so assumptions have to be made to improve tractability. Timescale separation between transport and turbulent process timescales allows the system of equations to be decoupled in mathematically and computationally decoupled *modules*. This is illustrated in the 1D energy equation (in cylindrical coordinates for simplicity) shown in Equation 32. Analogues exist for the magnetic flux diffusion equation, density equation and momentum equation. The transport coefficients q_s and sources $Q_s(r,t)$ are typically calculated by physics models, under the assumption that the process timescale is much less than the PDE timestep. In this study we focus on the energy and density transport, as these are the coefficients calculated by QLKNN.

$$\frac{3}{2}\frac{(\partial n_s T_s)}{\partial t} + \frac{1}{r}\frac{(\partial r q_s)}{\partial r} = Q_s(r,t) \tag{32}$$

We have implemented QLKNN as a transport module inside JINTRAC and RAPTOR[19]. In the current implementation QLKNN provides the main ion flux $q_{i,1}$, electron heat flux q_e and electron particle flux Γ_e. For multiple ion species we assume the same GyroBohm heat flux for each ion. This involves a multiplication by ion density, and hence leads to negligible impurity heat flux as expected. Contrary to RAPTOR which evolves electron density directly, JINTRAC solves the ion density equations for particle transport. Since the current version of QLKNN contains only Γ_e, we thus assume for each ion $\Gamma_i = \frac{n_i}{n_e Z_i}\Gamma_e$, maintaining ambipolarity. Finally, for numerical stability, we use either an effective diffusion D_{eff} or convection V_{eff}, derived from the total particle flux, depending on the flux direction and density gradient. V_{eff} is used for up-gradient particle transport and for low density gradients $\left(\left|\frac{R}{L_n}\right| < 0.1\right)$. Future work will aim to improve on these assumptions by neural network fits on species dependent D_i and V_i directly, which is important for multiple-isotope fuelling and impurity transport applications.

Neural networks do not extrapolate well outside their training set boundaries. In this work, this is trivial to detect, as the training set was a bounded regular hyperrectangle. We

chose to clip the inputs to the input layer of the neural network within the bounds of the hyperrectangle, with a margin of 5% on all sides. Alternative approaches for are also possible, such as training multiple neural networks to form a "committee", where extrapolation is detected from increased variance of the committee predictions in zones with sparse or non-existent data. This increase in variance arises from different local minima of the weight optimizations due to random initialisation. This is more suitable for training sets which are not pre-selected hyperrectangles, such as the training derived from experimental databases[14]. We chose not to implement this here due to the additional calculation times involved, and the trivial structure of our training set.

A. QLKNN simulation results within integrated modelling

We now compare QLKNN simulations to full QuaLiKiz within integrated modelling, for a representative set of 3 JET H-mode discharges. The correspondence between QuaLiKiz and the experimental profiles will not be discussed here, and on this point we refer the reader to the citations where the original JINTRAC-QuaLiKiz simulations were carried out for each of the cases. We focus on the correspondence between QuaLiKiz and QLKNN, as well as between the implementations within JINTRAC and RAPTOR.

To judge the quality of the neural network regression and the impact of the made assumptions, we first show a comparison of QLKNN and QuaLiKiz on the high performance JET baseline #92436[11] within both JINTRAC and RAPTOR integrated modelling. The JINTRAC-RAPTOR comparison further acts as a benchmark exercise for correct coupling of QLKNN within the code suites. These simulations correspond to an averaged 500 ms time-window during discharge flattop. A Gaussian Process Regression fit is performed on the kinetic profile data, and the distribution average is used as initial condition[11]. The current, temperature and density profiles are then evolved over multiple energy confinement times until the temperature and density profiles are in stationary-state, and compared to the experimental fits. As QLKNN is only applicable for turbulent transport in the tokamak core, we evolve temperature and density only inside $\rho_{N,tor} = 0.85$, and include a proxy transport coefficient for sawtooth-induced transport in the core for all simulations in this work.[11]. Appendix A contains a full overview of the used settings. Furthermore, JINTRAC and RAPTOR contain a different treatment of magnetic geometry. To reduce the impact of these differences and instead compare QLKNN itself as close as possible for the JINTRAC-RAPTOR benchmark, we force the geometric normalization constants a and R, as well as the q and $\hat{s}$ profile to be identical to the final condition of the JINTRAC simulations at the QLKNN model input. We show three simulations to investigate QLKNN model performance in different levels of increasing physics fidelity.

In Figure 6 we show in the left column a simulation closest to the QLKNN assumptions, namely a single Carbon impurity species and no rotation. We then increase physics fidelity in the center column by using a more realistic mix of impurity isotopes, and inclusion of the ad-hoc electromagnetic stabilisation rule[11] and solving the magnetic equilibrium self-consistently with ESCO. These settings are described in detail in[11] where the original JINTRAC-QuaLiKiz simulations were carried out. Finally in the right column we include the effect of rotation in the outer-half radius of the plasma[5,11] using the QuaLiKiz native impact-of-rotation prediction for QuaLiKiz, and for QLKNN the new QLKNN-rotation-rule as described in section VI. The parameter of merit for QLKNN performance is the relative root mean square (RRMS) difference of the predicted profiles compared to the original QuaLiKiz runs. See Equation 33, where the summation is over JINTRAC simulation radial grid points. The RRMS between JINTRAC-QLKNN and JINTRAC-QuaLiKiz are shown in Table III.

$$RRMS \equiv \sqrt{\frac{\sum_{i=0}^{n}(QLK_i - \hat{NN}_i)^2}{\sum_{i=0}^{n} QLK_i^2}} \tag{33}$$

As is clear from the left and middle columns in Figure 6, corresponding to the first two table rows, the final kinetic profiles predicted by JINTRAC-QLKNN and JINTRAC-QuaLiKiz agree very well. The maximum discrepancy is on the order 1-10%. Importantly, this small hit in accuracy comes with momentous speed increases, especially for the full-physics case, from around 7 days of walltime on 16 cores for JINTRAC-QLKNN to around 20 minutes of walltime on two cores for JINTRAC-QLKNN. At this simulation speed, QLKNN itself was no longer the bottleneck in the JINTRAC-QLKNN simulations, and increasing the amount of cores further had little effect. The largest difference between QLKNN and QuaLiKiz can be found in the full-physics case, shown in the rightmost column and last row of III. This discrepancy is is mainly caused by the different treatment of rotation, which is expected. While the inclusion of rotation did lead to an increase in n_e and T_i in the QLKNN simulations, for this case the degree of stabilisation is less than in QuaLiKiz itself.

The RAPTOR-QLKNN simulations also match the JINTRAC-QuaLiKiz simulations well, in the 2-10% range. As the QLKNN model in both JINTRAC and RAPTOR were identical, further differences are attributed to differences in the numerical schemes implemented in RAPTOR and JINTRAC, which will be investigated in future work. We emphasize that the RAPTOR-QLKNN simulation took only 90 s on a single core using 0.05 s time steps. This 5 order of magnitude simulation speedup, at similar accuracy, compared to the original JINTRAC-QuaLiKiz simulations, is the key result of this work. Furthermore, in these RAPTOR simulations, QLKNN was still the bottleneck, so extending QLKNN inside RAPTOR using MPI parallelism is currently under investigation for yet further speedup.

Next we show the general applicability of QLKNN in two more JET shots. The first is the high collisionality baseline JET H-mode scenario #73342[18,36], where the simulation corresponds to a stationary-state during flattop, and the GPR fit time-window was taken to be 500 ms. The second case is high performance JET hybrid scenario #92398, subject to DT extrapolation in upcoming campaigns[37]. To demonstrate the capabilities of JINTRAC-QLKNN for dynamic evolution, this

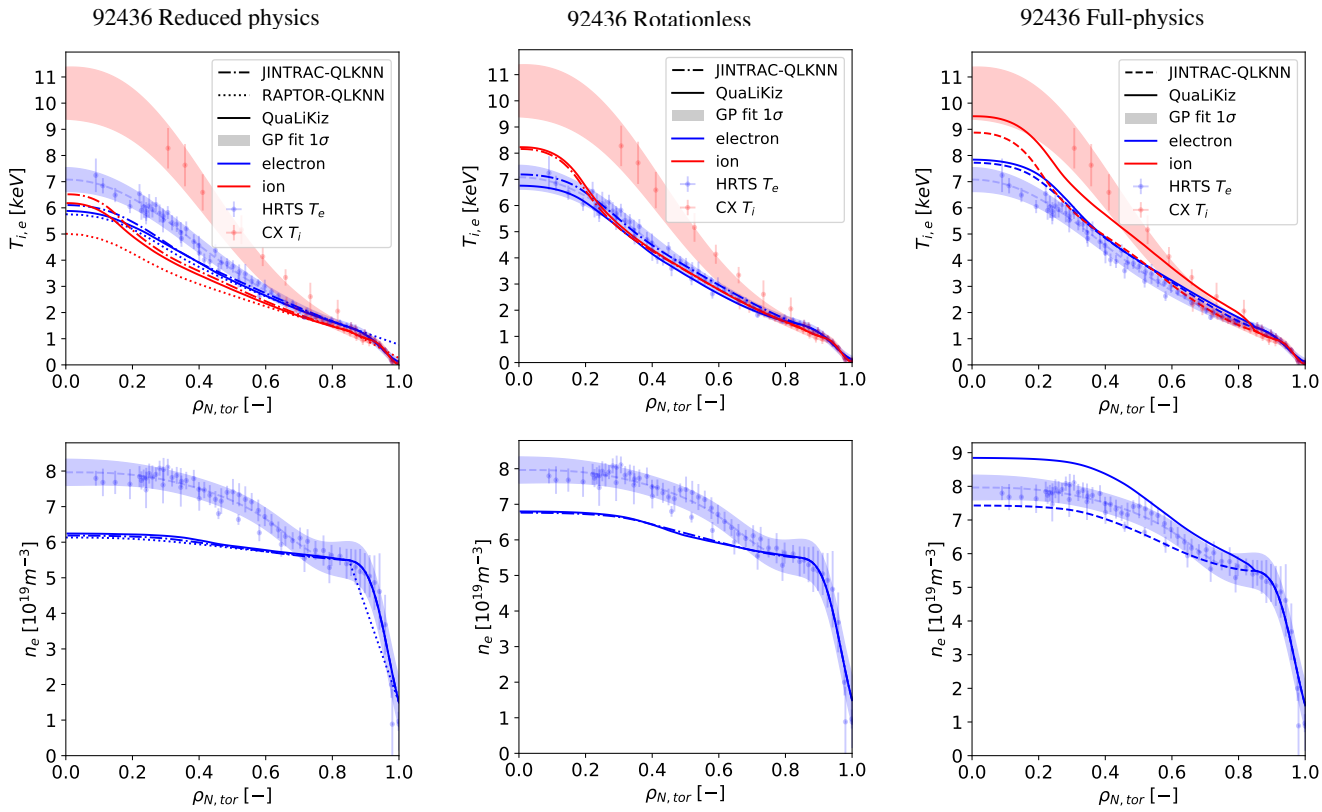

FIG. 6. The final kinetic profiles of the JINTRAC-QuaLiKiz (solid) and JINTRAC-QLKNN (dash-dot) simulations of JET shot #92436. Shown are the final temperatures for the ions (top, red) and electrons (top, blue) as well as the final electron density (bottom, blue). From left to right we show three cases of increasing physics fidelity: a reduced physics case, a more complete but rotationless case, and finally a case with rotation. Note the excellent agreement between QLKNN and QuaLiKiz in all figures, although a larger discrepancy was found for the case with rotation. This is expected, as the treatment of rotation is different in QuaLiKiz and QLKNN.

TABLE III. A comparison of the final kinetic profiles of JINTRIC-QuaLiKiz and JINTRAC-QLKNN using the relative root mean square profile difference (RRMS). Shown are a simulation close to QLKNN assumptions (Reduced physics), a simulation with realistic isotope mix and magnetic geometry, but without rotation (Rotationless), and the same simulation with rotation (Full-physics), all for JET shot #92436. The average was taken in the simulation region $0.2 < \rho_{N,tor} < 0.85$. The differences between QLKNN and QuaLiKiz are small, especially for the rotationless simulations. The different treatment of rotation results in larger differences for the full-physics case.

Simulation	RRMS [%]		
	T_i	T_e	n_e
Reduced physics (JINTRAC)	3.9	2.3	0.9
Reduced physics (RAPTOR)	9.8	3.0	1.5
Rotationless	1.8	8.4	0.5
Full-physics	2.6	15	14

discharge was simulated during the density buildup following the LH transition. The GPR fits for each snapshot during the evolution was taken to be 50 ms. Both cases were re-run with JINTRAC-QuaLiKiz for this paper, with interpretive impurities, meaning that the impurity profiles were constrained to match the main ion profile peaking.

TABLE IV. The final kinetic profile differences between JINTRIC-QuaLiKiz and JINTRAC-QLKNN using the relative root mean square profile difference (RRMS) for the simulations of JET shot #73342 and #92398. We show simulations without rotation (Rotationless), as well as with rotation (Full-physics), similar to III. Again, the differences between QLKNN and QuaLiKiz are small for the rotationless case, and larger for the full-physics case.

Simulation	RRMS [%]		
	T_i	T_e	n_e
73342 rotationless	0.3	1.5	0.9
73342 full-physics	3.1	3.8	2.8
92398 rotationless	12	10	7
92398 full-physics	13	10	9.9

The final kinetic profiles and comparison between QuaLiKiz and QLKNN for #73342 are shown in Figure 7 and Table IV, and for #92398 in Figure 8. For #92398 we also show the temperature and density temporal evolution at three radial locations in Figure 9.

Again we note the excellent agreement between JINTRAC-QuaLiKiz and JINTRAC-QLKNN. The rotationless #73342 case matches excellently within 2%. #92398 matches less

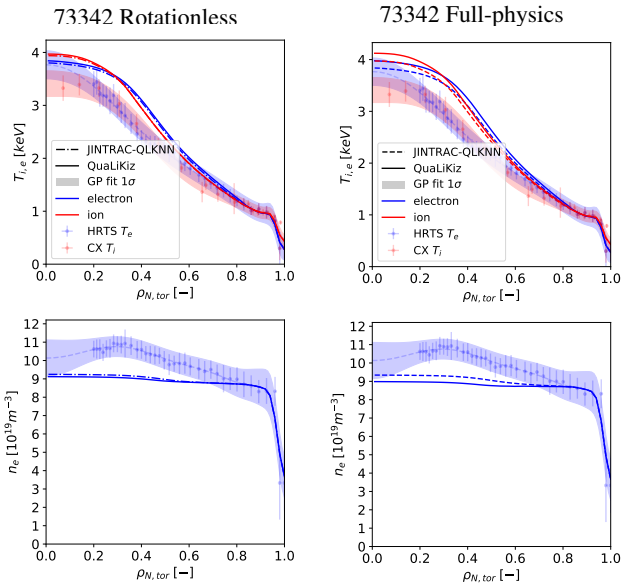

FIG. 7. The final kinetic profiles of the JINTRAC-QuaLiKiz (solid) and JINTRAC-QLKNN (dash-dot) simulations of JET shot #73342. Shown are the final temperatures for the ions (top, red) and electrons (top, blue) as well as the final electron density (bottom, blue). Both cases were run with interpretive impurities without rotation (left) and with rotation (right). The QLKNN predictions lie close to the QuaLiKiz ones, in the order of 4% at maximum, which show the generality of applying QLKNN as quicker surrogate for the full QuaLiKiz model.

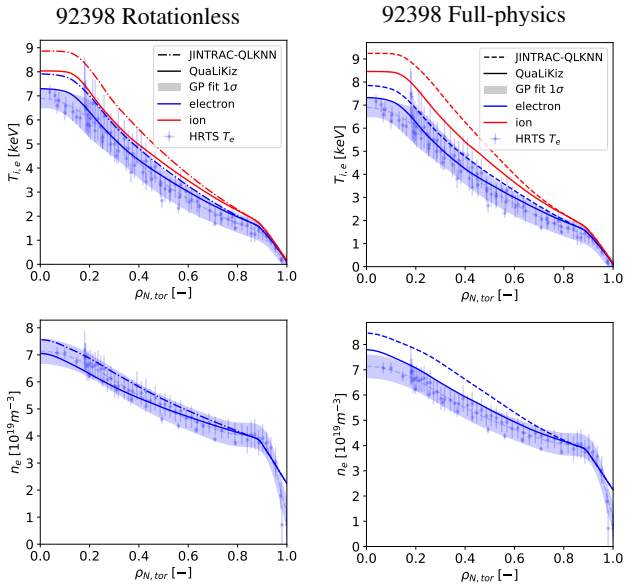

FIG. 8. The final kinetic profiles of the JINTRAC-QuaLiKiz (solid) and JINTRAC-QLKNN (dash-dot) simulations of JET shot #92398. Shown are the final temperatures for the ions (top, red) and electrons (top, blue) as well as the final electron density (bottom, blue). Both cases were run with interpretive impurities without rotation (left) and with rotation (right). Here the disagreement between QuaLiKiz and QLKNN is larger than previous cases, but still within 13%. Future improvements to the QLKNN model are expected to lower these differences, but this result shows that even in this state the QLKNN model can be used for quick explicative studies.

well in comparison, around 10%. While still good, we expect the match to improve by expanding the QLKNN input dimensionality, most notably a better capture of different impurity species and Shafranov shift, which are both planned in future work. Note that the better agreement between JINTRAC-QuaLiKiz and JINTRAC-QLKNN in the #92436 case for the rotationless case compared to #92398 may simply be coincidental, as the impact of the input dimensions not included in QLKNN can 'cancel out'. Next generations of QLKNN will have further increased input dimensionality.

For the cases with rotation, the impact on #73342 is small, simply due to low rotation in this high-density case. For #92398, the agreement between the native QuaLiKiz and QLKNN rotation rules is excellent, both boosting n_e and T_i significantly, and by the same magnitude. Note that T_e is barely impacted by rotation, since the T_e profile is predicted to be clamped by ETG turbulence for this discharge, both in the original QuaLiKiz and the QLKNN simulations.

The dynamic behaviour of QLKNN for #92398 is shown in Figure 9. The match between JINTRAC-QuaLiKiz and JINTRAC-QLKNN is excellent, most notably the density build-up in the lower plot, staying within a discrepancy of 4% at mid-radius for the whole duration. However, the small differences between the two models compound from the outer-radius inward and over multiple timesteps, resulting in the relatively larger but still acceptable discrepancy for the final condition in Figure 8. While factor 4 less than compared to #92436, the speed-up gained in the #92398 simulation is still very significant, from 11 hours on 16 cores to 5 wallminutes on 2 cores.

VIII. CONCLUSIONS AND OUTLOOK

We have shown a method to train physics-based neural networks as turbulent transport models, which we applied to generate a surrogate model for the fast quasilinear gyrokinetic transport model QuaLiKiz. Utilizing HPC, we generated a large dataset of $3 \cdot 10^8$ flux calculations, which was used as training set for fully connected feed forward neural networks for regression. Prior physics knowledge of the underlying model features was incorporated by using a customized cost function, choosing appropriate training targets, and looking beyond traditional measures of goodness. This surrogate model, QLKNN, has been integrated into two integrated modelling suites, JINTRAC and RAPTOR. We applied the JINTRAC-QLKNN ensemble to carry out predictive dynamic simulations of core transport in three JET shots, covering a representative spread of H-mode operating space. We have also shown one similar simulation using RAPTOR-QLKNN, in good agreement with JINTRAC-QLKNN. This benchmark was important for verifying the implementation of QLKNN in both code suites. Furthermore, the RAPTOR-QLKNN simulation is significantly faster than JINTRAC-QLKNN by 2 orders of magnitude. This was due primarily to larger timesteps

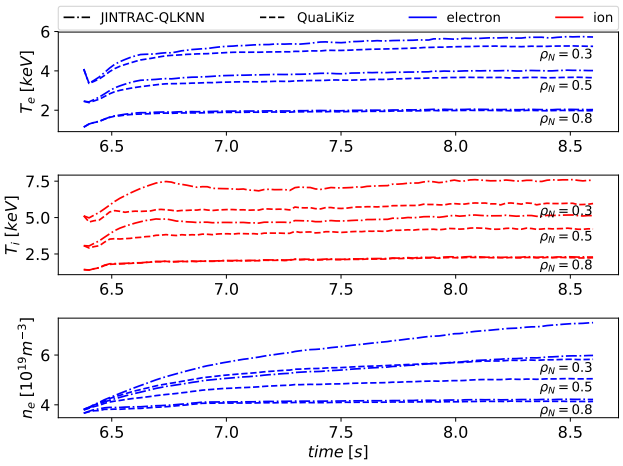

FIG. 9. A time-dependent JINTRAC-QLKNN simulation without rotation of JET #92398. Note the density buildup that is very well captured by QLKNN. The RRMS differences at $\rho = 0.5$ for the full time-evolution are $T_e = 8\%$, $T_i = 9\%$, and $n_e = 4\%$

in RAPTOR's implicit PDE solver, which is greatly facilitated by the analytical input-output derivatives available in the neural network transport model. The steady-state and dynamic kinetic profiles match those of the full QuaLiKiz simulations closely, while being up to five orders of magnitude faster to run.

The largest discrepancy between QLKNN and QuaLiKiz is caused by the different rotation rules employed between QLKNN and QuaLiKiz. The rotationless cases studied in this work showed differences from 1%-10% in the final kinetic profiles. The rotation cases studied showed mildly larger differences ranging from 3%-15%. The rotation discrepacy was more prevalent for the #92436 case studied. An improved treatment of rotation will be part of future work, for example by implementing the quench rule on the individual growth rates in the spectrum before evaluating the saturation rule, thus capturing spectral shifts.

Future work will improve the QLKNN model by extending to larger input space, focusing on the impurity density gradient, and multiple-ion transport important for multiple-isotope fuelling applications and impurity transport. Additionally, using a robust method to fit a large amount of experimental kinetic profiles[11], one can base a training set on experimental data, instead of the hyperrectangle methodology described here. This allows for more input dimensions to be used, as well as including rotation by using the native QuaLiKiz rotation model, instead of a rotation rule as described here. There are also other techniques to include physics information in neural networks. The late fusion method can be used to include functional information in the network architecture itself, for example by constraining the mapping to a critical gradient model, and has already been successfully used in a proof-of-principle QuaLiKiz surrogate model[38].

Beyond the model improvements, work can now commence on extensive experimental validation of QLKNN predictions, as well as using QLKNN for scenario optimisation and design. As shown in this work, physics-based neural network surrogate models can enable first-principle dynamic transport simulations at unprecedented speeds, opening up new avenues for tokamak scenario optimization and realtime control applications.

ACKNOWLEDGMENTS

This work has been carried out within the framework of the EUROfusion Consortium and has received funding from the Euratom research and training programme 2014-2018 and 2019-2020 under grant agreement No 633053. The views and opinions expressed herein do not necessarily reflect those of the European Commission. This research used resources of the National Energy Research Scientific Computing Center (NERSC), a U.S. Department of Energy Office of Science User Facility operated under Contract No. DE-AC02-05CH11231. We are grateful to F. Jenko for assistance with computational resources.

Appendix A: Appendixes

[1] F. M. Poli, Physics of Plasmas **25**, 055602 (2018).
[2] E. J. Doyle, W. A. Houlberg, Y. Kamada, V. Mukhovatov, T. H. Osborne, A. Polevoi, G. Bateman, J. W. Connor, J. G. Cordey, T. Fujita, X. Garbet, T. S. Hahm, L. D. Horton, A. E. Hubbard, F. Imbeaux, F. Jenko, J. E. Kinsey, Y. Kishimoto, J. Li, T. C. Luce, Y. Martin, M. Ossipenko, V. Parail, A. Peeters, T. L. Rhodes, J. E. Rice, C. M. Roach, V. Rozhansky, F. Ryter, G. Saibene, R. Sartori, A. C. C. Sips, J. A. Snipes, M. Sugihara, E. J. Synakowski, H. Takenaga, T. Takizuka, K. Thomsen, M. R. Wade, H. R. Wilson, ITPA Transport Physics Topical Group, ITPA Confinement Database and Modelling Topical Group, and ITPA Pedestal and Edge Topical Group, Nuclear Fusion **47**, S18 (2007).
[3] W. Horton, B. Hu, J. Q. Dong, and P. Zhu, New Journal of Physics **5**, 14 (2003).
[4] M. Barnes, I. G. Abel, W. Dorland, T. Görler, G. W. Hammett, and F. Jenko, Physics of Plasmas **17**, 056109 (2010).
[5] J. Citrin, C. Bourdelle, F. J. Casson, C. Angioni, N. Bonanomi, Y. Camenen, X. Garbet, L. Garzotti, T. Görler, O. Gürcan, F. Koechl, F. Imbeaux, O. Linder, K. van de Plassche, P. Strand, and G. Szepesi, Plasma Physics and Controlled Fusion **59**, 124005 (2017).
[6] C. Bourdelle, J. Citrin, B. Baiocchi, A. Casati, P. Cottier, X. Garbet, F. Imbeaux, and JET Contributors, Plasma Physics and Controlled Fusion **58**, 014036 (2016).
[7] *QuaLiKiz homepage*.
[8] G. M. Staebler, J. E. Kinsey, and R. E. Waltz, Physics of Plasmas **14**, 055909 (2007).
[9] G. Cenacchi and A. Taroni, JET-IR , 84 (1988), eNEA-RT-TIB–88-5.
[10] M. Romanelli, G. Corrigan, V. Parail, S. Wiesen, R. Ambrosino, P. da Silva, A. Belo, L. Garzotti, D. Harting, F. Köchl, T. Koskela, A. Lauro-Taroni, C. Marchett, A. Mattei, E. Militello-asp, M. F. F. Nave, S. Pamela, A. Salmi, P. Strand, and G. Szepesi, Plasma and Fusion research **9**, 3403023 (2014).
[11] A. Ho, J. Citrin, F. Auriemma, C. Bourdelle, F. J. Casson, H.-T. Kim, P. Manas, G. Szepesi, and H. Weisen, Nuclear Fusion **59**, 056007 (2019).
[12] S. Breton, F. J. Casson, C. Bourdelle, J. Citrin, Y. Baranov, Y. Camenen, C. Challis, G. Corrigan, J. Garcia, L. Garzotti, S. Henderson, F. Koechl, E. Militello-Asp, M. O'Mullane, T. Pütterich, M. Sertoli, and M. Valisa, Nuclear Fusion **58**, 096003 (2018).
[13] O. Linder, J. Citrin, G. M. D. Hogeweij, C. Angioni, C. Bourdelle, F. J. Casson, E. Fable, A. Ho, F. Koechl, and M. Sertoli, Nuclear Fusion **59**, 016003 (2018).

TABLE V. Lorum ipsum With additional BgB tranport (ion particle = 1) lower limits for thermal (10)

Field name/option	Value/setting			
Shot number	73342 (both)	92398 (both)	92436 (full)	92436 (NN comp)
JAMS version	v121218	v080817	v121218	
Number of grid points	51	101	101	101
Start time a (s)	60.75	46.3779	50	50
End time a (s)	62.75	48.6	52	52
Min. time step (s)	1e-13	1.0e-08	1.0e-13	1e-13
Max. time step (s)	1e-3	1.0e-03	1.0e-03	1e-3
Ion (1) mass (u)	2	2	2	2
Simulation boundary	0.85	0.85	0.85	0.85
Equilibrium	EFIT	ESCO	ESCO	EFIT
Equilibrium boundary	-	0.995	0.998	-
Toroidal field	-	2.798	2.8	-
Plasma current	2.5e6	2.2e6	2.9e6	2.9e6
Neoclassical transport model	NCLASS	NCLASS	NCLASS	NCLASS
Bootstrap current	yes	yes	yes	yes
BgB transport factor (χ)	0.08	0.03	0.08	0.08
BgB transport factor (mom)	1	3	1.25	1.25
Particle tansport min	10	1	10	10
Impurities	Interpretive	Interpretive	SANCO	SANCO

TABLE VI. 73342

Additional transport			
Shape	Gaussian	Gaussian	Gaussian
Centre	0	0	0
Height (cm 2 s -1)	4e4	2e4	2e4
Width	0.25	0.25	0.25

TABLE VII. 92398

Additional transport			
Shape	-	Gaussian	-
Centre	-	0	-
Height (cm 2 s -1)	-	0.15	-
Width	-	1e3	-

TABLE VIII. 92436

Additional transport			
Shape	Gaussian	Gaussian	Gaussian
Centre	0	0	0
Height (cm 2 s -1)	2e4	1e4	1e4
Width	0.212	0.212	0.212

TABLE IX. impurities

	species 1			species 2			species 3		
simulation	m	c	ss	m	c	ss	m	c	ss
73342	12	6	4						
92398	58	28	28						
92436 (full-physics)	9	4	4	58	28	28	184	74	74
92436 (reduced-physics)	12	6	6						

[14] O. Meneghini, S. P. Smith, P. B. Snyder, G. M. Staebler, J. Candy, E. Belli, L. Lao, M. Kostuk, T. Luce, T. Luda, J. M. Park, and F. Poli, Nuclear Fusion **57**, 086034 (2017).

[15] R. J. Goldston, D. C. McCune, H. H. Towner, S. L. Davis, R. J. Hawryluk, and G. L. Schmidt, Journal of Computational Physics **43**, 61 (1981).

[16] A. Pankin, D. McCune, R. Andre, G. Bateman, and A. Kritz, Computer Physics Communications **159**, 157 (2004).

[17] M. D. Boyer, S. Kaye, and K. Erickson, Nuclear Fusion **59**, 056008 (2019).

[18] J. Citrin, S. Breton, F. Felici, F. Imbeaux, T. Aniel, J. F. Artaud, B. Baiocchi, C. Bourdelle, Y. Camenen, and J. Garcia, Nuclear Fusion **55**, 092001 (2015).

[19] F. Felici, J. Citrin, A. A. Teplukhina, J. Redondo, C. Bourdelle, F. Imbeaux, and O. Sauter, Nuclear Fusion **58**, 096006 (2018).

[20] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed. (Prentice Hall PTR, Upper Saddle River, NJ, USA, 1998).

[21] G. Cybenko, Mathematics of Control, Signals and Systems **2**, 303 (1989).

[22] M. Nielsen, "Neural networks and deep learning," (2019).

[23] L. Bottou, F. E. Curtis, and J. Nocedal, arXiv e-prints , arXiv:1606.04838 (2016), arXiv:1606.04838 [stat.ML].

[24] D. Masters and C. Luschi, CoRR **abs/1804.07612** (2018), arXiv:1804.07612.

[25] J. Citrin, H. Arnichand, J. Bernardo, C. Bourdelle, X. Garbet, F. Jenko, S. Hacquin, M. J. Pueschel, and R. Sabot, Plasma Physics and Controlled Fusion **59**, 064010 (2017).

[26] D. D. Team, *Dask: Library for dynamic task scheduling* (2016).

[27] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," (2015), software available from tensorflow.org.

[28] S. van der Walt, S. C. Colbert, and G. Varoquaux, Computing in Science Engineering **13**, 22 (2011).

[29] *QLKNN-fortran repository* ().

[30] *Luigi repository*.

[31] D. P. Kingma and J. Ba, CoRR **abs/1412.6980** (2014).

[32] *QLKNN JSON networks repository* ().

[33] F. Jenko, Computer Physics Communications **125**, 196 (2000).

[34] V. I. Dagnelie, *Dynamics of linear ITG modes with flow shear in ballooning space*, Master's thesis, Utrecht University (2017).

[35] V. I. Dagnelie, J. Citrin, F. Jenko, M. J. Pueschel, T. Görler, D. Told, and H. Doerk, Physics of Plasmas **26**, 012502 (2019), https://doi.org/10.1063/1.5030416.
[36] B. Baiocchi, C. Bourdelle, C. Angioni, F. Imbeaux, A. Loarte, and M. Maslov, Nuclear Fusion **55**, 123001 (2015).
[37] F. J. Casson, H. Patten, C. Bourdelle, S. Breton, J. Citrin, F. Köchl, C. Angioni, Y. Baranov, R. Bilato, E. A. Belli, C. D. Challis, G. Corrigan, A. Czarnecka, O. Ficker, L. Garzotti, M. Goniche, J. P. Graves, T. Johnson, K. Kirov, P. J. Knight, E. A. Lerche, M. J. Mantsinen, J. Mlynář, M. Sertoli, M. Valisa, and J. E. T. Contributors, IAEA book of abstacts (2018).
[38] D. Schaefer, *Hybrid Neural Networks in Nuclear Fusion Transport Modelling*, Master's thesis, Faculty of Physics, LMU (2019).