Modelling Turbulent Transport For The Tokamak L-Mode Edge using Neural Networks

Camille Bouvy
Department of Data Science and Artificial Intelligence
Maastricht University
Maastricht, The Netherlands

***Abstract*—Current day fusion experiments require accurate computational models predicting the heat and particle transport in tokamaks. However, these models require substantial time and computation resources. To accelerate predictions, fast surrogate models based on neural network regression of quasilinear turbulent tokamak transport codes have been developed. The regression reproduces the input-output space with high speed and accuracy. The current state-of-the-art neural network surrogates are the QLKNN-hyper-10D developed at DIFFER and QLKNN-jetexp-15D, which both have shown good fidelity when implemented into the integrated models JINTRAC and RAPTOR. However, there is a transition region towards the edge region of the tokamak plasma that exhibits different dynamics than the tokamak plasma core. As such, this region can be considered in a different class of parameter regime. This project presents a neural network model trained on dataset devoted to this edge parameter space. The database covers a wide range of realistic tokamak edge parameters. The trained and optimised surrogate model was able to reproduce the QuaLiKiz outputs ostensibly well. It was also evaluated using measures of goodness based on physics features of the underlying model. Computing the discrepancy in integrated modelling between QLKNN and QuaLiKiz is out of the scope of this project and is left for future work.**

I. Introduction

With a growing global population and an urgent need to reduce fossil fuel usage, researchers have been looking for a technology capable of generating large quantities of clean and sustainable energy. Fusion power could satisfy this demand, and become a safe and sustainable part of the world's future energy supply. It indeed offers the prospect of an almost inexhaustible source of energy that is low carbon, has very little waste, and is safe, with no risk of meltdown unlike in nuclear fission. Nuclear fusion is the reaction powering the Sun and the stars. This physical process occurs when two light nuclei join together to make one heavy nucleus [1]. If those light interacting nuclei belong to elements with low atomic numbers, substantial amounts of energy are released. For this process to occur, high temperatures (at least 150,000,000 degrees Celsius) and pressure are required. Nowadays, the leading approach to fusion energy is the tokamak. This is a large machine that heats gas until it fuses into plasma, and transforms the energy released into electricity. It uses a powerful magnetic field to confine the plasma [2]. Unfortunately, due to technological and engineering challenges, nuclear fusion has not left the experimental stage for over half a century. One of the major issues facing its development is how demanding and expensive research experiments are. A way to tackle this deficiency is by creating computational models that emulate the results seen in current fusion experiments. This allows for the interpretation of present-day experiments, optimization and design of future experiments, and if fast enough can be applied for real-time control.

Thus, accurate computational models predicting the plasma temperature, density and rotation in tokamaks are required. In order to model all the fusion dynamics, many models representing various physics phenomena are integrated together into a single modelling framework. An essential component of these integrated models is the prediction of turbulent fluxes. Calculating those fluxes is most commonly done using nonlinear gyrokinetic models. These model the behaviour of plasma particles within high magnetic fields using theory taking advantage of spatiotemporal scale separation. For example, the typical instability timescales are much longer than gyration times of particles in the magnetic field. Unfortunately, they exhibit a computational time too expensive for routine simulation of tokamak discharge evolution.

Reduced order turbulence models have thus been developed for increased tractability. For instance, QuaLiKiz, a Quasilinear turbulence model developed at DIFFER, invokes the quasilinear approximation. This allows the prediction of turbulent fluxes approximately six orders of magnitude faster than nonlinear codes [3]. However, when integrated into modelling suites such as JINTRAC, it can still take a few days to run.

To further accelerate integrated modelling workflows, fast surrogate models based on neural network (NN) regression of quasilinear turbulent tokamak transport codes have been developed. The regression reproduces the original model with high speed and accuracy within the confines of the parameter space of their original training set. The current state-of-the-art NN surrogates are the lattice-input-space-based QLKNN-hyper-10D [4], based QuaLiKiz, and the JET-experiment-input-space-based QLKNN-jetexp-15D [5].

Both QLKNN variants are valid for the bulk of the core region of a tokamak plasma, consisting of closed nested magnetic flux surfaces. However, there is a transition region towards the edge region of a tokamak, as depicted in Figure 1. This "near-edge" region is in H (high-confinement) mode at high input power, typically a narrow region of suppressed transport and steep gradients called the "pedestal". The top of

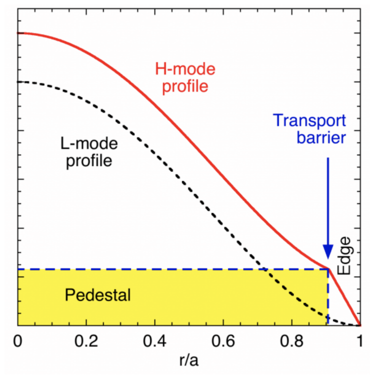


Fig. 1. Plot of the plasma pressure given the plasma radius, showing the pedestal and pressure radial profiles for both H-Mode and L-Mode [6]

the pedestal is then the internal boundary condition for core plasma tokamak simulation. Nevertheless, in "L-mode" at low input power, this near-edge region has relatively low temperatures, high logarithmic gradients, and high collisionality. As such, it can be considered in a different class of parameter regime compared to the rest of the tokamak plasma core.

This project focuses on generating a QuaLiKiz training set for NN regression, for L-mode edge parameter space. Due to constraints that can be applied on the input space, such as equilibration of ion and electron temperatures and gradients, the training set size is much smaller than those used for the inner-core variants of QLKNN. Similar work has been carried out in the past at DIFFER [7], but due to more recent improvements made in the QuaLiKiz collisionality model and numerics (see Appendix A), this is an opportune moment to revisit this work.

The main research question addressed in this work was: "How accurately can neural networks predict the QuaLiKiz turbulent transport model for the L-mode "near-edge" region?". The first step of this research was to create the NN training set containing the parameters relevant for the L-mode edge region. Next, the most optimal hyperparameters were selected and the NN surrogate model was generated. It was finally benchmarked against the original QuaLiKiz results, and its performance was assessed using a variety of customized metrics. Future work would benefit from combining this model with the inner core NN turbulence model, and testing the performance of the two pasted NN in tokamak simulation in order to further evaluate the model's accuracy.

The sections III, IV, V outline the different methods used to conduct this work. Section VI details how the model is evaluated. Section VII describes the experiments carried out and their results, and the performance of the final model is analysed in Section VIII. Finally, a summary is provided in Section IX and suggestions are made on potential future work.

II. Related work and alternative approaches

Machine learning has found multiple applications in nuclear fusion research. Neural networks have been used to predict plasma disruptions [8], reconstruct magnetic configurations [9] and also combined with Support Vector Machines (SVM) to classify L-H transitions onsets [10]. Most related to the work of this paper are other neural network surrogates for physics models applied within tokamak integrated modeling [4], [11], [12]. Nevertheless, neural networks are not the only machine learning technique finding use in nuclear fusion. Among others, one can use Random Forests to predict disruptions [13], linear regression to estimate tokamak boundary [14], Bayesian analysis to reconstruct plasma profiles [15] and Gaussian processes combined with Genetic Algorithms to validate plasma transport code [16]. In more recent work, Gaussian Process Regression has also been applied to plasma turbulent transport model validation via integrated modelling [17].

In regards to surrogate models, many other techniques than Artificial Neural Networks have been developed over the past years. Best known are Polynomial Surface Response Models (PRSM) [18], Radial Basis Functions (RBF) [19] and Support Vector Regression (SVR) [20]. Each of those models suits various types of problems differently. Polynomials surrogates, such as PRSM, have been shown to be less performant on unstructured or natural data. Models like RBFs on the other hand use a weighted sum of simple functions, enabling them to emulate more complex design. Naturally, one has to keep in mind that in most cases, increased accuracy is closely related to a lengthier computational time. SVRs can be seen as an extension to RBFs methods modelled after the Support Vector Machine theory [21]. This method is an elegant way of producing predictions from large sets of noisy data, hence applying it as a surrogate for QuaLiKiz could be an interesting approach for future work. In this project, Neural Networks were used as they are very reliable and well-established models.

III. Training set generation

In order for the neural network to accurately emulate the quasilinear gyrokinetic QuaLiKiz model, it has to learn from input-output mappings. To that effect, the QuaLiKiz code (version 2.8.1) is executed with a constrained input space corresponding to the dimensions most significantly impacting turbulent transport in the L-Mode. A more detailed description of QuaLiKiz can be found in Appendix A. The selected input parameters for dataset the are the following:

- Normalized logarithmic ion temperature gradient (R/L_{Ti})
- Normalized logarithmic electron temperature Gradient (R/L_{Te})
- Normalized logarithmic density gradient (R/L_{ne})
- Magnetic pitch angle/ safety factor (q)
- Magnetic pitch angle shear ($\hat{s}$)
- Radial coordinate (x)
- Dilution (n_{main_ion}/n_e)
- Collisionality (v^*)
- Ion-electron temperature ratio (T_i/T_e)

Plasma rotation has been excluded as it is less significant in L-Mode. The input space consists of a non-uniform rectangle

Parameter	Num. points	Min value	Max value
$k_\theta \rho_s$	16	0.1	45
R/L_{Ti}	17	0	150
R/L_{Te}	17	0	150
R/L_{ne}	14	0	110
q	10	2	30
$\hat{s}$	12	1	40
x	1	0.95	0.95
n_{main_ion}/n_e	5	0.1	0.5
$v*$	8	0.1	3
T_i/T_e	1	1	1

TABLE I
DATASET POINTS OF THE HYPERRECTANGLE AND THEIR RANGE

grid with the chosen bounds given in Table I. These were picked based on previous experience, observations and physical knowledge to cover a wide range of realistic tokamak core parameter. In this regard, it was chosen to only include a single data point for both x and T_i/T_e. Reason for this is that, for the radial coordinate x, the difference in the trapped electron fraction in the L-mode edge region (of $x = [0.9, 1.0]$) only has a small impact on the modes, so taking only a single point for x is sufficient. For the ion-electron temperature ratio on the other hand, a single point taking value 1 was chosen as the ion-electron heat exchange at the relatively high collisionality is very high (so it can be assumed that ions and electrons have the same temperature). In a few cases, T_i/T_e can go up to 1.5. Hence, it would be interesting to investigate more T_i/T_e points with value 1.25 and 1.5. This is however out of the scope of this project and is left for future work.

The QuaLiKiz outputs investigated are ion and electron heat flux and electron particle flux in three different turbulence modes. Those examined turbulent modes are Ion Thermal Gradient (ITG), Trapped Electron Mode (TEM) and Electron Thermal Gradient (ETG) modes, as they are considered as causing the bulk of the turbulent transport. Other modes may contribute to transport under the simulated conditions, but they are not implemented within QualiKiz, and are thus outside the scope of this work. All outputs are dimensionless (Gyro-Bohm units or GB) to allow generalisation to various tokamaks.

The resulting database was generated using HPC resources on the Marconi supercomputer, using $\simeq$ 250000 CPUh. It consisted of 4197 netCDF files, that were then concatenated in one big file. As the QuaLiKiz run is a scan over a few parameters, all arrays were re-casted to an orthogonal base. This last step improves the effectiveness of dataset queries. Next, this hypercube dataset was converted to HDF5 (pandas) and all missing values were deleted to obtain a NN training readable format.

Erroneous data in the training set can diminish the neural network accuracy. While QuaLiKiz has been continuously improved over the past years, some inaccuracies can remain in the output. This is particularly true in this case, as QuaLiKiz generally predicts the tokamak plasma core and that this dataset is based on the edge. In order to ensure that the dataset is kept clean from errors, the untrusted QuaLiKiz outputs are filtered out. These include:

- Points where ambipolarity is violated by more than 50%
- Points with very tiny transport ($\leq 1 \times 10^{-4}$ GB units)
- Points where the sum of the fluxes calculated in separate modes is over 50% more than the total flux

Moreover, points with negative fluxes were set to zero and undefined values (NaN or Not a Number) were removed. Typically, points where QuaLiKiz breaks down and underestimates the flux for large temperature gradients are also filtered out. Unfortunately, due to an OutOfMemory Error in the filtering code, this was not carried out in this project. Finally, the dataset is split into a test, validation and training set.

IV. NEURAL NETWORK

A. *Feed Forward Neural Networks*

A neural network is a type of machine learning algorithm modelled after the human brain [22]. Its goal is to recognize underlying relationships in a set of data using a number of simple, highly interconnected processing elements: neurons. Each neuron is represented by a node, having an associated weight and bias. Neural networks are comprised of multiple node layers, namely an input layer, one or more hidden layers and an output layer.

The NN architecture used in this study was a fully connected feed-forward NN (FFNN). In those networks, each node is connected to all nodes in the next layer, but are not connected to any in the current or previous layers. Each node in a layer takes the output of each node in the previous layer as input, and then sends a linear combination of the inputs and biases to the next layer. If the output of any individual node is above a specified threshold value, that node is activated and sends data to the next layer of the network. Otherwise, no data is passed along to the next layer of the network. For information passed forward from a node i within a layer to a node j within the next layer, this gives the following expression:

$$y_j = f\left(\sum_i w_{ij} y_i + b_j\right) \tag{1}$$

where y_j represents the output of neuron j, w and b represent the neurons weights and biases, and f is the activation function. There are various existing activation functions ($tanh$, $relu$, $sigmoid$,...) Each have their advantages and drawbacks, and the function choice typically depends on the application.

The accuracy of the network is improved by optimizing the parameters of the network (weights and biases). This is done by minimizing a *cost function* which is calculated using the difference between the outputs produced by the network and the target outputs from the training set. The exact cost function used in this work is depicted in Section IV-B. The optimization algorithm used to update the weights and biases is mini-batch gradient descent and is further described in Appendix B.

Before starting the training process, the dataset is split into different subsets: a training set used to update the weights and biases, a validation set used to check generalization of the neural network model after every step of the optimizer algorithm, and a test set which is just used to estimate the network's performance after training.

B. Neural Network Adaptation to physics constraints

Regularized neural networks do not assume any features of the underlying mapping. However, plasma behaviour has physics-informed features that need to be respected, namely:

1) Sharp flux "critical threshold" behaviour and zero flux in the region where no instabilities are predicted
2) Identical threshold for all transport channels

In order for those features to hold, some changes are made to the usual NN training methodology.

First of all, the cost function is adapted to force a sharp critical threshold and to ensure that the flux is zero in regions where no instabilities are predicted by QuaLiKiz. Typically the cost function consists of a measure of goodness-of-fit a regularizing term. The first term compares for each set of inputs, the neural network output to desired targets (in this case the QuaLiKiz input-output mappings), and the second one prevents overfitting. In order to force the sharp critical threshold, a third term is added to the cost function. This term punishes positive FFNN predictions in ostensibly stable regions (predicted stable by QuaLiKiz) while remaining zero for negative FFNN predictions in the stable region (which are then subsequently clipped to zero). Putting all three terms together gives the following cost function:

$$C = C_{good} + \lambda_{regu} C_{regu} + \lambda_{stab} C_{stab} \tag{2}$$

$$C_{good} = \begin{cases} \frac{1}{n}\sum_{i=1}^{n}(QLK_i - NN_i)^2 & QLK_i \neq 0 \\ 0 & QLK_i = 0 \end{cases} \tag{3}$$

$$C_{regu} = \sum_{i=1}^{k}(w_i^2) \tag{4}$$

$$C_{stab} = \begin{cases} 0 & QLK_i \neq 0 \\ \frac{1}{n}\sum_{i=1}^{n}(NN_i - c_{stab}) & QLK_i = 0 \end{cases} \tag{5}$$

where NN_i are the network predictions, QLK_i are the QuaLiKiz calculation, n and k are the amount samples and weights and λ_{regu}, λ_{stab} and c_{stab} are optimized hyperparameters. The quadratic criterion provided in Equ. 3 was employed as it is the benchmark criterion for cost functions.

Second, the training process is modified so that all transport channels (ions and electrons) have the same threshold. The training targets are split by mode (ITG, TEM, ETG) and a single network is trained for each. This allows for flux ratio regressions of sufficient quality. For each mode, a network predicts the leading flux, and others predict the flux ratio (e.g. for TEM the leading flux is q_e and the flux ratios are Γ_e/q_e and q_i/q_e). The output of the per-mode predicting networks is then added together in the transport model implementation in post-processing using an unweighted sum.

V. Grid Search

When training a neural network, there are many parameters that can be tuned to give a good performance, commonly called hyperparameters. In contrast to the weight in bias, they are kept constant during the training process. In order to find the most optimal values for hyperparameters, an algorithm called Grid Search can be used. This method performs an exhaustive search over the hyperparameters in the desired search range. The network is then trained with each possible combination of these points and the best value is picked by evaluating the network's performance [23]. In this work, the evaluation is done using the metrics described in VIII.

Grid Search suffers from the curse of dimensionality. Therefore, the more hyperparameters considered in the search the less efficient the algorithm. However, as explained in Section VII-B, not many hyperparameters have to be optimized in this work, making Grid Search suitable for this work. If more dimensions were to be investigated, an algorithm such as Random Search might be more adequate.

VI. Evaluation Metrics

Modelling of turbulent transport in fusion plasma is a task that can not be accurately evaluated using classical regression assessment methods. The state-of-the-art approach to measure the performance of trained neural networks in this application is to apply them in integrated transport modelling. However, the integrated model is computationally expensive, and using it in the hyperparameter optimization process would be quite tedious. There is currently no objective and quantitative measures of goodness for trained networks in this work, but some metrics can give insight to determine the performance of the model.

The main error metric used to assess regression models is the Root Mean Squared Error (RMSE), given by

$$RMSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{6}$$

where y_i is the target output and $\hat{y}_i$ is the predicted one. This metric is thus an absolute measure of the goodness for the fit, evaluating how close the prediction is to the target value.

Moreover, as mentioned in Section IV-B, the model needs to capture certain physics features in order to be valid. Therefore, its performance can also be evaluated by analysing the following measures:

- The percentage of slices where the model predicts a threshold
- The percentage of slices where the model predicts fluxes in an ostensibly stable region (depicted in Figure 2)
- The ”unsteadiness” (when the curve is not smooth) of the prediction after the threshold

Naturally, one would desire to achieve high results in the first measure, and minimize the two others. All those metrics are investigated together in order to determine an optimal neural network. Unfortunately, the odds that a neural network performs best for all those metrics are very low. A compromise thus has to be made when selecting the hyperparameters. Future work would profit from the investigation of a single measure of goodness uniting all those metrics.

VII. Experiments and Results

This section describes the results of the training set generation and hyperparameter optimization. For the sake of time, all

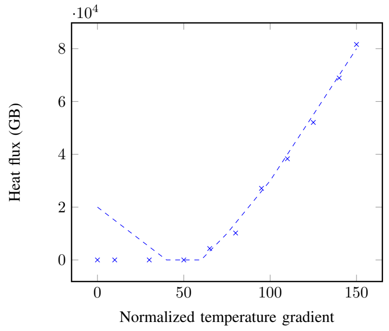


Fig. 2. Example of a neural network prediction exhibiting a ”pop-back” behaviour (i.e. predicting fluxes in the stable region) on the left of the figure

neural networks described in the following section are trained on a single target, namely the ion heat flux q_i in ITG mode (efiITG_GB).

A. *Final training set*

The QuaLiKiz output dataset obtained contains 5×10^7 input-flux relations before filtering. The percentage of the dataset removed by each filter is listed below. The indicated percentages are with respect to the total dataset size before the filtering.

- Points where ambipolarity is violated by more than 50%: 0.1%
- Points with very tiny transport predicted ($\leq 1 \times 10^{-4}$ GB): 0.005%
- Points where ITG/TEM/ETG heat flux is 50 % greater than total flux: 2.43%

The dataset after filtering consists of 4.97×10^6 input-flux relations, with 18.07% of the fluxes being set to zero (as they were predicted to be negative). Ultimately, $\simeq$ 92% of the original expanded dataset serve as training set for the NN. Within this training set, none of the input-flux relations is completely stable (i.e. exhibits no ITG, TEM, or ETG instabilities), the same goes for ITG and ETG dominant modes. Moreover, only 1.11% exhibit dominant TEM modes. The remaining input-flux relations yield a combination of TEM and the other two instabilities, with 1.21% being a combination of TEM and ETG modes, 60.99% being a combination of TEM and ITG modes, and finally the remaining exhibiting all three instabilities.

An overview of the variable mean for each of the different fluxes (given in GB units) is provided in Table II. As a point of comparison, the mean values of the TEM, ITG and ETG heat fluxes of the core database were respectively 3.47, 9.33 and 3.54. This significant difference is explained by the fact that at low temperatures, the GyroBohm flux increases, as the GB scaling factor has $T^{2.5}$ at the denominator. Those fluxes in SI (W/m^2) are therefore still reasonable in this regime. Furthermore, one can notice that some mean values are considerably different from the corresponding median values. This is justified by the fact that there are a few points at very high fluxes (e.g. at q_e = 1500GB - as a matter of comparison,

Mode	Variable	Mean (in GB)	Median (in GB)
TEM	q_e	503.71	345.91
	q_i/q_e	0.96	0.54
	Γ_e/q_e	0.23	0.22
ITG	q_i	465.53	241.05
	q_e/q_i	3.42	0.56
	Γ_e/q_i	0.32	0.25
ETG	q_e	168.00	0.00

TABLE II
QuaLiKiz Outputs

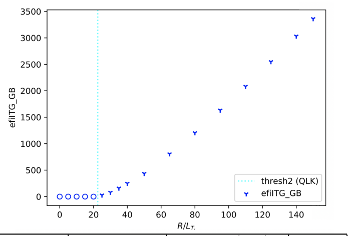
heat fluxes of 100GB have a higher temperature than the centre of the Sun) which behave erratically. These represent the plasma disruptions. Part of those points could be filtered out by the aforementioned temperature gradient breakdown filter. Nevertheless, some outliers would still remain in the dataset. The neural network might thus give defective predictions at high fluxes.

Ultimately, a standard slice of QuaLiKiz output resembles the one depicted in Figure 3. Typically, the predicted fluxes are stable (i.e. zero) until they hit the threshold. At this stage, they start to increase smoothly. Some of the data slices however still encompass some anomalies. For instance, Figure 4 displays a data slice exhibiting a ”sawtooth” behaviour. This irregularity is most likely originated by a problem in the numerics in this high R/L_T, R/L_n regime. Similarly, several slices at high R/L_n start predicting instabilities from the outset and therefore do not possess a threshold (see Figure 5). Future work would benefit from the investigation and filtering of these erroneous data slices.

B. *Hyperparameter Optimization*

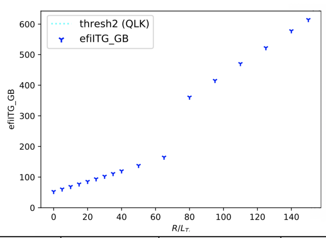
Most of the hyperparameter values used for this NN were validated in previous work. For instance, the values for the learning rate (used for Mini-Batch Gradient Descent), λ_{regu} and λ_{stab} were already optimized in previous work concerning NN emulating QuaLiKiz in L-Mode Edge [7] and can be reused here. The early stop patience, c_{stab} and Optimizer can keep the default values used in the current state-of-the-art NN surrogate, as they achieve satisfying results [4]. Likewise, the activation function used for all neurons is the default one, $f(x) = tanh(x)$. Selecting this function as activation can result in slower convergence, but it leads to a smoother training rate with less variability between networks.

The number of hidden layers, nodes per layer, λ_{L2} and batch size hyperparameters were optimized specifically for this work. The reason for this is that the regularization term and batch size have the most influence on the smoothness of the output and the training time respectively. Likewise, the number of hidden layers and neurons in each have a great influence on the model's complexity. As mentioned in Section V, this optimization was done using grid search. In order to accelerate the process, two grid searches were performed, one of them taking the λ_{L2} and batch size and the other the node and layer numbers. This can be done as these hyperparameter pairs are greatly inter-correlated, but do not have a high influence on the other pair. The investigated values for each of the



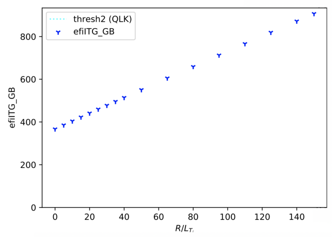
smag	dilution	$log_{10}(v*)$	q
6.0	0.65	1.0	3.0

Fig. 3. Slice of the QuaLiKiz output for the parameters given above, where the dashed line represents the threshold



smag	dilution	$log_{10}(v*)$	q
1.0	1.0	2.5	20.0

Fig. 4. Slice of the QuaLiKiz output for the parameters given above, where the datapoints exhibit a ”sawtooth” behaviour



smag	dilution	$log_{10}(v*)$	q
1.0	1.0	0.66	2.0

Fig. 5. Slice of the QuaLiKiz output for the parameters given above, where no threshold is predicted

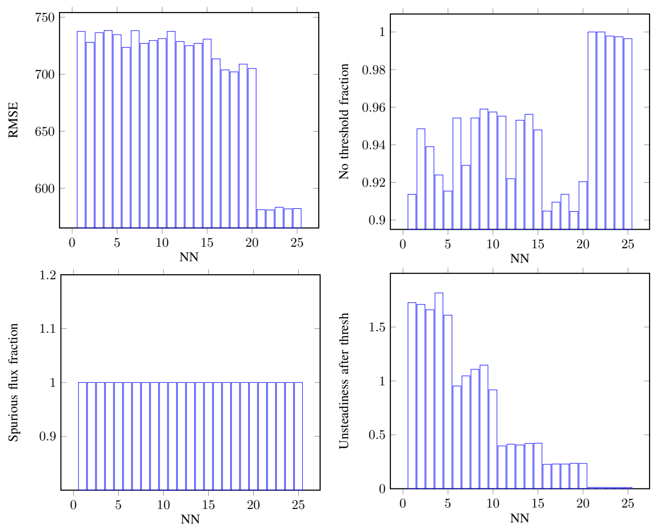


Fig. 6. Performance of the networks of the grid search for L2 regularisation and batch size. The metrics shown are RMSE (top left), percentage of slices with no threshold (top right), percentage of slices without spurious flux predictions (bottom right) and unsteadiness in the unstable zone (bottom right). The plotted networks each have a different combination of the two hyperparameters values, with L2 regularisation term increasing.

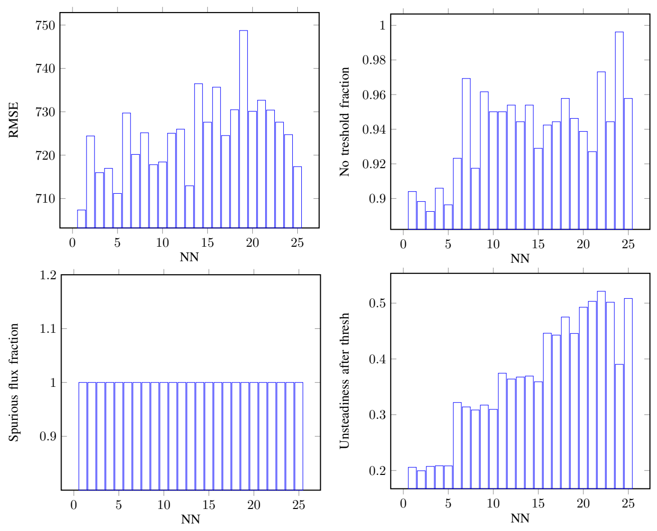


Fig. 7. Performance of the networks of the grid search for hidden layer number and node number. The metrics shown are RMSE (top left), percentage of slices with no threshold (top right), percentage of slices without spurious flux predictions (bottom right) and unsteadiness in the unstable zone (bottom right). The plotted networks each have a different combination of the two hyperparameters values, with an increasing layer number.

hyperparameters are five points spread around the optima of the state-of-the-art NN. They have bounds respectively [1, 5], [50, 175], [5×10^{-6}, 0.1] and [100, 10000] for the number of hidden layers, node number, λ_{L2} and batch size. The first grid search was performed using the current state-of-the-art NN default structure (i.e. 3 hidden layers with 128 nodes each).

Figure 6 and 7 display the results of this process, with the NNs representing all the possible values combination for the investigated hyperparameters. The first noticeable aspect that one could observe is that for both grid searches, the fraction of spurious flux is always one, meaning that none of the networks predicts spurious fluxes before the threshold. Although this is an indication of good performance for the models, it should be noted that, as revealed by the threshold fraction plots, models do not predict any thresholds in most cases.

Another conspicuous aspect in Figure 6 is that the NNs from 20 to 25 have very low RMSE and very low unsteadiness. This can be explained by the fact that those models have the highest regularization term (λ_{L2} = 0.1). That is, they tend to underfit the data, and the prediction model resembles a classic linear one (see Figure 8). Consequently, all those models also never predict a threshold (as reflected in the no threshold fraction, where all of them take value one).

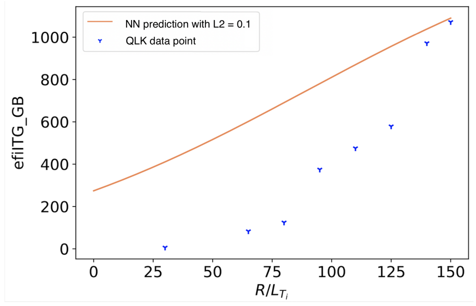


Fig. 8. Slice of neural network prediction of the ITG-driven transport flux q_i as a function of the ion temperature gradient R/L_{Ti} with λ_{L2} = 0.1 and batch size = 10000

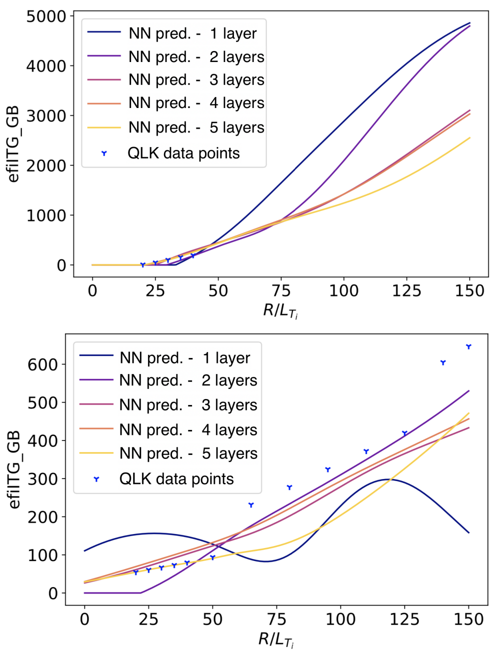


Fig. 9. Two slices of neural network prediction of the ITG-driven transport flux q_i as a function of the ion temperature gradient R/L_{Ti} with different number of hidden layers (where each layer has 128 neurons)

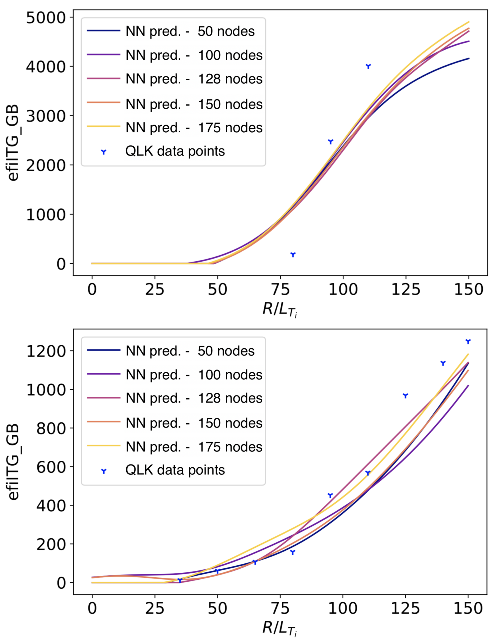


Fig. 10. Two slices of neural network prediction of the ITG-driven transport flux q_i as a function of the ion temperature gradient R/L_{Ti} with different number of nodes per hidden layer (with two hidden layers)

Parameter	**Optimized value**
Number of layers	2
Neurons per layer	128
λ_{regu}	0.1
λ_{stab}	1
λ_{L2}	0.0005
Batch size	5000
Early stopping patience	15
c_{stab}	-5
Optimizer	Adam
Learning rate	0.01

TABLE III
OPTIMIZED HYPERPARAMETER VALUES USED TO TRAIN THE NEURAL NETWORK

Other models appear to obtain better results. For instance, NN 12 has a relatively low unsteadiness after the threshold, while getting one of the lowest scores for the no threshold fraction metric (although the NNs all perform relatively similarly for this measure). Consequently, the hyperparameter values inherent in NN 12 (λ_{L2} = 0.0005 and batch size = 5000) were selected and the second grid search was performed using these.

Figure 7 shows the results for the grid search performed on the neural network's architecture. Looking at those plots, it seems like all models have relatively similar performance. The most distinctive feature is the unsteadiness after the threshold, which is lower for networks comprising of less hidden layers. This is attributable to the fact that increasing the number of hidden layers increases the model's complexity. That is, networks with a single hidden layer might fit the data less well, as they would more likely exhibit a linear behaviour. As the chosen metrics are not very revealing with regards to which model performs best, the networks are further investigated by visual inspection.

The first parameter to be examined is the layer number. Figure 9 depicts two slices predicted by neural networks with different hidden layer numbers (the neuron number per layer is 128, the default value of the current state-of-the-art model). Generally, the networks have relatively similar performance, as shown by the top plot of Figure 9. However, when the data exhibits irregularities (akin to the ones outlined in Section VII-A), the network with two hidden layers seems to generalize best (see the bottom picture in Figure 9). That is, the number of layers chosen for the final model is two.

Next, the number of neurons per layer is tested. Predictions of neural networks with two hidden layers and a various number of neurons per layers are shown in Figure 10. From these plots, one can infer that the number of neurons per layer does not have a strong effect on the model's performance. The networks with 128 and 150 neurons per layer however seem to have a slightly sharper threshold. Considering that networks with fewer nodes have lower computational time, the selected value for the number of nodes per layer is 128.

The final hyperparameter values used in this work are presented in Table III.

VIII. MODEL EVALUATION

In this section, an attempt to validate the QLKNN-Edge model by comparing its predictions to the original QuaLiKiz model is discussed. The numerical validity of the NN was evaluated by comparing the predicted results to QuaLiKiz outputs over parameter scans. This is depicted in Section VIII-A. The performance of the NN within integrated modelling frameworks should have been tested using the integrated model JINTRAC. However, due to time constraints, this was

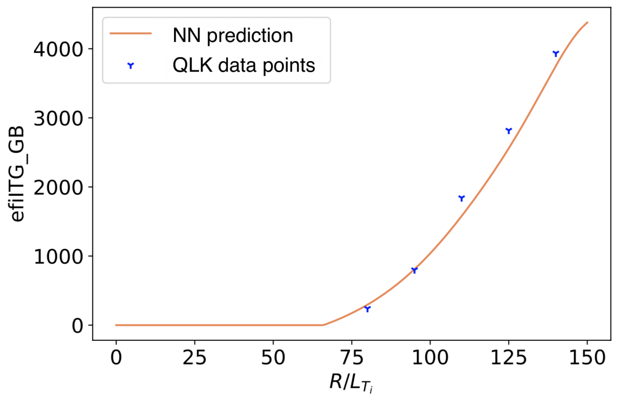


Fig. 11. Comparison of the ITG-driven transport flux q_i as a function of the ion temperature gradient R/L_{Ti} predicted by the final neural network (line) and QuaLiKiz (dots)

Metric	Value
RMSE	525855.937
No threshold fraction	0.917
Spurious flux fraction	1.0
Unsteadiness after threshold	0.308

TABLE IV
OVERVIEW OF THE MEASURES OF GOODNESS OBTAINED FOR THE FINAL NEURAL NETWORK

not carried out in this project and is left for future work. Section VIII-B details the framework that would have been used to carry out this work.

A. *Simulation results compared to QuaLiKiz*

Figure 11 compares the NN surrogate model to the original QuaLiKiz evaluations over a parameter scan of the ion temperature gradient R/L_{Ti} (with the other parameters being fixed). Looking at this plot, one can infer that the model fits the data points pretty well and exhibits the desired threshold behaviour. It should nevertheless be noted that the dataset contains a significant amount of irregular slices, which adversely affects some of the network's predictions. The relatively high threshold prediction error for instance is attributable to the erroneous data, as most QuaLiKiz slices also do not predict a threshold.

The values for the measures of goodness described in Section VI of the final neural network are outlined in Table IV.

B. *Simulation results within integrated modelling*

In order to test the model's performance when applied in transport code, it is classically implemented as a transport module inside an integrated model. A commonly used intergated modelling framework is JINTRAC. For this model to be integrated into JINTRAC, it has to be applied alongside the inner core NN turbulence model so that the full Tokamak region is predicted. The NN trained on the edge mode is valid for the region of the radius (ρ) from 0.9 to 0.98. Since the integrated model JINTRAC is pretty robust, it is good enough to place the two neural networks alongside without using any smoothing technique, such that the core NN predicts regions of the radius from ≈ 0.2 to 0.9, and that the edge NN predicts the regions of the radius from 0.9 to 0.98. The lower end of the radius or Magnetohydrodynamic (MHD) transport is predicted using a proxy-sawtooth transport patch. The results of the simulation would have then been validated against the original QuaLiKiz model within integrated modelling, JINTRAC-QuaLiKiz outputs.

IX. Conclusion

Within this work, a QuaLiKiz dataset for the L-mode edge parameter space was built, and was used as a training set for fully connected feed-forward neural networks. Prior physics knowledge of the underlying model features was incorporated by using customized measures of goodness. By utilising a grid search method, the optimal hyperparameters were selected in order to get the best performance.

The research question addressed in this work was: "How accurately can neural networks predict the QuaLiKiz turbulent transport model for the L-mode "near-edge" region?". Ordinarily, the performance within integrated modelling is investigated. Due to time constraints, this was not carried out in this work. However, the model's accuracy can also be assessed using the aforementioned metrics. Through the optimisation process, a threshold prediction error of 91%, spurious flux error of 0% and unsteadiness in the unstable region of 0.308 were achieved. Furthermore, the model was validated by visual inspection.

Future work is foreseen in re-training the neural networks on a dataset comprising fewer anomalies and irregularities. The first step towards a cleaner dataset would be to apply the temperature gradient breakdown filter. Additionally, it was observed that leaving the undefined values (NaN) in the dataset can result in a better overall performance of the model. A longer-term goal would be to include all networks for all fluxes (as in this work only the ion heat flux in ITG mode was investigated) and implementing these networks in integrated models such as JINTRAC. This would allow to assess the model's performance more accurately and with less reliance on visual judgement. Finally, it would be worthwhile to generate another QuaLiKiz dataset including more data points (for instance including $T_i/T_e = 1.25$ and 1.5), and benchmark neural networks trained on each dataset to one another.

X. Acknowledgements

This thesis was prepared in partial fulfilment of the requirements for the Degree of Bachelor of Science in Data Science and Artificial Intelligence, Maastricht University. The project was pursued in collaboration with DIFFER, the Dutch Institute for Fundamental Energy Research. Special thank you to the supervisors Ronald Westra Menica Dibenedetto, Jonathan Citrin and Karel Lucas van de Plassche who provided support and guidance throughout the whole project.

References

[1] J. Shultis and R. Faw, *Fundamentals of Nuclear Science and Engineering*. Taylor & Francis, 2002.

[2] D. Leslie and C. Woods, *Front Matter*. John Wiley & Sons, Ltd, 2005.

[3] 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. S. and, "Tractable flux-driven temperature, density, and rotation profile evolution with the quasilinear gyrokinetic transport model qualikiz," *Plasma Physics and Controlled Fusion*, vol. 59, no. 12, p. 124005, nov 2017. [Online]. Available: https://doi.org/10.1088/1361-6587/aa8aeb

[4] K. L. van de Plassche, J. Citrin, C. Bourdelle, Y. Camenen, F. J. Casson, V. I. Dagnelie, F. Felici, A. Ho, S. V. Mulders, and J. Contributors, "Fast modeling of turbulent transport in fusion plasmas using neural networks," in *Invited Papers from the 2nd International Conference on Data-Driven Plasma Science*, 2020.

[5] A. Ho, "Development of neural networks towards predict-first plasma modelling," Ph.D. dissertation, Applied Physics, Mar. 2021, proefschrift.

[6] F. Wiki, "Pedestal," available online at: http://fusionwiki.ciemat.es/wiki/Pedestal, last accessed on 08.04.2021.

[7] G. M., "Turbulent transport modelling in the pedestal forming region with neural networks," Master's thesis, Department of Applied Physics Eindhoven University of Technology, 2019.

[8] K.-H. J., S. A., and T. W., "Predicting disruptive instabilities in controlled fusion plasmas through deep learning," *Nature*, vol. 568, 2019. [Online]. Available: https://doi.org/10.1038/s41586-019-1116-4

[9] D. Böckenhoff, M. Blatzheim, H. Hölbe, H. Niemann, F. Pisano, R. Labahn, and T. S. P. and, "Reconstruction of magnetic configurations in w7-x using artificial neural networks," *Nuclear Fusion*, vol. 58, no. 5, p. 056009, mar 2018. [Online]. Available: https://doi.org/10.1088%2F1741-4326%2Faab22d

[10] P. Gaudio, A. Murari, M. Gelfusa, I. Lupelli, and J. Vega, "An alternative approach to the determination of scaling law expressions for the l–h transition in tokamaks utilizing classification tools instead of regression," *Plasma Physics and Controlled Fusion*, vol. 56, no. 11, p. 114002, oct 2014.

[11] H. M. and N. E., "Development of a surrogate turbulent transport model and its usefulness in transport simulations," *Plasma and Fusion Research*, vol. 16, p. 2403002, feb 2021.

[12] O. Meneghini, S. Smith, P. Snyder, G. Staebler, J. Candy, E. Belli, L. Lao, M. Kostuk, T. Luce, T. Luda, J. Park, and F. Poli, "Self-consistent core-pedestal transport simulations with neural network accelerated models," *Nuclear Fusion*, vol. 57, no. 8, p. 086034, jul 2017.

[13] C. Rea1, R. Granetz, K. Montes, R. Tinguely, N. Eidietis, J. Hanson, and B. Sammuli, "Disruption prediction investigations using machine learning tools on diii-d and alcator c-mod," *Plasma Physics and Controlled Fusion*, vol. 60, no. 8, p. 084004, jun 2018.

[14] V. Škvára, V. Smidl, and J. Urban, "Robust sparse linear regression for tokamak plasma boundary estimation using variational bayes," *Journal of Physics: Conference Series*, vol. 1047, p. 012015, 06 2018.

[15] M. Irishkin, F. Imbeaux, T. Aniel, and J. Artaud, "Applications of bayesian temperature profile reconstruction to automated comparison with heat transport models and uncertainty quantification of current diffusion," *Fusion Engineering and Design*, vol. 100, pp. 204–219, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0920379615300247

[16] P. Rodriguez-Fernandez, A. E. White, A. J. Creely, M. J. Greenwald, N. T. Howard, F. Sciortino, and J. C. Wright, "Vitals: A surrogate-based optimization framework for the accelerated validation of plasma transport codes," *Fusion Science and Technology*, vol. 74, no. 1-2, pp. 65–76, 2018. [Online]. Available: https://doi.org/10.1080/15361055.2017.1396166

[17] A. Ho, J. Citrin, F. Auriemma, C. Bourdelle, F. Casson, H.-T. Kim, P. Manas, G. Szepesi, and H. W. and, "Application of gaussian process regression to plasma turbulent transport model validation via integrated modelling," *Nuclear Fusion*, vol. 59, no. 5, p. 056007, mar 2019. [Online]. Available: https://doi.org/10.1088/1741-4326/ab065a

[18] R. Myers and D. Montgomery, *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*, 2nd ed. Wiley Interscience New York, 2002.

[19] M. F. Hussain, R. R. Barton, and S. B. Joshi, "Metamodeling: Radial basis functions, versus polynomials," *European Journal of Operational Research*, vol. 138, no. 1, pp. 142–154, 2002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221701000765

[20] S. M. Clarke, J. H. Griebsch, and T. W. Simpson, "Analysis of Support Vector Regression for Approximation of Complex Engineering Analyses," *Journal of Mechanical Design*, vol. 127, no. 6, pp. 1077–1087, 08 2004. [Online]. Available: https://doi.org/10.1115/1.1897403

[21] A. I. Forrester and A. J. Keane, "Recent advances in surrogate-based optimization," *Progress in Aerospace Sciences*, vol. 45, no. 1, pp. 50–79, 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0376042108000766

[22] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed. Prentice Hall, July 1998.

[23] Á. B. Jiménez, J. L. Lázaro, and J. R. Dorronsoro, *Finding Optimal Model Parameters by Discrete Grid Search*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 120–127. [Online]. Available: https://doi.org/10.1007/978-3-540-74972-1_17

[24] C. Bourdelle, J. Citrin, B. Baiocchi, A. Casati, P. Cottier, X. Garbet, and F. I. and, "Core turbulent transport in tokamak plasmas: bridging theory and experiment with QuaLiKiz," *Plasma Physics and Controlled Fusion*, vol. 58, no. 1, p. 014036, dec 2015. [Online]. Available: https://doi.org/10.1088/0741-3335/58/1/014036

[25] C. D. Stephens, X. Garbet, J. Citrin, C. Bourdelle, K. L. van de Plassche, and F. Jenko, "Quasilinear gyrokinetic theory: A derivation of qualikiz," 2021.

[26] "Qualikiz 2.8.1." [Online]. Available: https://gitlab.com/qualikiz-group/QuaLiKiz/-/tags/2.8.1

[27] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," 2018.

Appendix A
QuaLiKiz Model

Predicting temperature, density and rotation of plasma in the confined core of tokamak requires accurate and rapid turbulent transport codes. Nonlinear gyrokinetic codes allow for a detailed understanding of tokamak core turbulent transport. However, these calculations are too computationally expensive for routine simulation of tokamak discharge evolution. To that effect, reduced turbulence models have been developed. QuaLiKiz is one of those models, calculating tokamak core plasma temperature and density by invoking the quasilinear approximation [24].

The QuaLiKiz calculation is comprised of three steps:

1) Eigenvalue solver calculating the complex frequencies for the ion temperature gradient (ITG), electron temperature gradient (ETG), and trapped electron mode (TEM) instabilities
2) Solving the quasilinear flux integrals to compute particle, angular momentum, and heat fluxes
3) A saturation rule that uses previously performed nonlinear kinetic simulations to build the final transport fluxes

The quasilinear approach thus comes into play with coupling the linear characteristics of the problem with nonlinear simulations informed by nonlinear physics [3].

This quasilinear approximation alone is not sufficient to reach the required level of tractability. Hence, further approximations and are assumptions are made in QuaLiKiz that greatly increase the calculation speed. A more exhaustive list of those assumptions QuaLiKiz makes use of can be found in [25]. Altogether, this allows to predict turbulent fluxes approximately six orders of magnitude faster than nonlinear codes.

More recently QuaLiKiz has been updated with more robust, better parallelizable and open-source internal integration routines, resulting in a 2x speedup in computational time. Additionally, the collisionality model and numerics have been improved, resulting in better predictions of the high-collisionality L-mode cases of the dataset. These changes are available in QuaLiKiz-2.8.1 [26].

APPENDIX B
MINI-BATCH GRADIENT DESCENT

Gradient descent is an optimization algorithm regularly used for finding the weights or coefficients of machine learning algorithms such as artificial neural networks. It works by making predictions on training data and using the error (i.e. the difference between the predicted output and target output) to update the model's parameters so that this error is reduced [27]. The updates are made by following the gradient of the cost function, moving in the opposite direction of its slope. They generally take the form

$$\theta_{n+1} = \theta_n - \alpha \nabla C(\theta_n) \tag{7}$$

where θ represents the parameters to update (weights and biases), α is the learning rate and C is the cost function. This updating of the hyperparameters at each iteration is commonly called *back-propagation*.

Mini-batch gradient descent is a variation of the gradient descent in which the training set is split into batches that are used to calculate model error and update model coefficients. Small batch size will generally be slower to converge, but the resulting model has better generalizing properties. The training starts with random initial values for θ_0 (in this work the initial values are taken from a random Gaussian distribution with mean 0 and standard deviation 1). The parameters are then updated after each batch using Equ. 7. Once the algorithm went through all the batches, the performance of the resulting neural network is evaluated using the validation set. One of those iterations is called an epoch. If convergence is reached, the training is stopped and the neural network is saved. If not, all samples are re-shuffled and new batches are taken, repeating this procedure until convergence is reached. In this work, early stopping is used in order to prevent overfitting. This technique stops training as soon as the validation loss has not decreased in some predefined length of epochs, a hyperparameter called patience.