A fast neural network surrogate model for the eigenvalues of QuaLiKiz

Physics of Plasmas

RESEARCH ARTICLE | DECEMBER 18 2023

A fast neural network surrogate model for the eigenvalues of QuaLiKiz

Special Collection: Papers from the 4th International Conference on Data-Driven Plasma Science

E. Fransson ; A. Gillgren ; A. Ho ; J. Borsander; O. Lindberg ; W. Rieck ; M. Åqvist ; P. Strand

Phys. Plasmas 30, 123904 (2023)
https://doi.org/10.1063/5.0174643

A fast neural network surrogate model for the eigenvalues of QuaLiKiz

Cite as: Phys. Plasmas **30**, 123904 (2023); doi: 10.1063/5.0174643
Submitted: 1 September 2023 · Accepted: 3 December 2023 ·
Published Online: 18 December 2023

E. Fransson,[1,a] A. Gillgren,[1,b] A. Ho,[2] J. Borsander,[1] O. Lindberg,[1] W. Rieck,[1] M. Åqvist,[1] and P. Strand[1]

AFFILIATIONS

[1]Chalmers University of Technology, Gothenburg, Sweden
[2]DIFFER–Dutch Institute for Fundamental Energy Research, Eindhoven, The Netherlands

Note: This paper is part of the Special Collection: Papers from the 4th International Conference on Data-Driven Plasma Science.
[a]**Electronic mail:** emil.fransson@chalmers.se
[b]**Author to whom correspondence should be addressed:** andreas.gillgren@chalmers.se

ABSTRACT

We introduce a neural network surrogate model that predicts the eigenvalues for the turbulent microinstabilities, based on the gyrokinetic eigenvalue solver in QuaLiKiz. The model quickly provides information about the dominant instability for specific plasma conditions, and in addition, the eigenvalues offer a pathway for extrapolating transport fluxes. The model is trained on a 5×10^6 data points large dataset based on experimental data from discharges at the joint European torus, where each data point represents a QuaLiKiz simulation. The most accurate model was obtained when the task was split into a classification task to decide if the imaginary part of eigenvalues were stable (≤ 0) or not, and a regression model to calculate the eigenvalues once the classifier predicted the unstable class.

I. INTRODUCTION

Integrated modeling is an important tool for the interpretation, prediction, and optimization of magnetically confined fusion experiments and will play an important role for designing future experiments and reactors, such as DEMO. The idea behind integrated modeling is that multiple different models that determine different phenomena in the plasma, such as transport, heating, and magnetic configuration, are coupled together into a single framework. An important part of the framework is the transport models that in part determine the confinement of the particles and energy. A good confinement is essential for the possibility of a future fusion power plant. The transport is dominated by plasma microinstabilities,[1,2] i.e., turbulent transport. The simulations with the greatest physics fidelity of turbulent transport are based on non-linear gyrokinetic theory. However, these models are much too computationally expensive to routinely be used in the integrated framework.

Hence, computationally cheaper reduced models based on quasi-linear theory have been developed over the past decades. These reduced models determine the eigenvalues of microinstabilites and use linear relations between the perturbations in the plasma to calculate the transport. A number of models with this approach have been developed such as QuaLiKiz,[3–5] TGLF,[6] and EDWM,[7] and they have been fitted and verified against non-linear gyrokinetic simulations.[8–10] These turbulent transport models have been used extensively in different integrated frameworks, such as JINTRAC,[11] ETS,[12,13] etc. Even though the reduced models are much computationally cheaper than their non-linear gyrokinetic counterpart, they still can be time and resource consuming when performing extensive analysis or long simulations, such as analysis of the ramp-up phase. As such, it is beneficial to speed up the reduced models, which also will allow for the possibility of real-time control applications.

Therefore, a surrogate model that reproduce the fluxes of QuaLiKiz has been developed and implemented in integrated frameworks.[14–16] This new model, QLKNN, has replicated the results of the full QuaLiKiz accurately at a fraction of the computational cost. However, QLKNN and the reduced models encounter challenges when applied to future devices due to the construction of the reduced models, i.e., in the quasi-linear saturation. The reduced models (and subsequently QLKNN) need a saturation rule to calculate the fluxes, and it needs to be benchmarked to achieve realistic values. This is done by comparing against non-linear gyro-kinetic simulations, but ultimately by validating with fluxes from experiments. This makes it

difficult to achieve reduced models with high accuracy for future devices, such as ITER and a DEMO.

In this work, we present a framework to address this issue by relying on the robustness of the underlying linear physics of the reduced models. The eigenvalues calculated by the reduced models translate between different devices since similar physics occur in different devices; thus, they make a more robust candidate for extrapolation. Therefore, we present the work of a surrogate model for the eigenvalues calculated by QuaLiKiz. This will make the eigenvalues quickly accessible, making it possible to perform high dimensional scans in parameter space to assess the instabilities. Additionally, the eigenvalues can be a combined with any saturation model, giving the possibility to determine the fluxes.

Here, we present the surrogate model for the eigenvalues calculated with QuaLiKiz trained on a large joint European torus (JET)-database. For a complete surrogate model for other devices and future device scenarios, datasets generated by QuaLiKiz for such parameter domains are needed. Additionally, the implementation of a saturation rule for the surrogate model is left as future work. In this study, we focus on a proof-of-principle for the underlying concept and investigate which considerations need to be accounted for to obtain an accurate model.

II. DATASET

A surrogate model generally consists of neural networks (NNs) that mimic a model/system by learning the mapping between input and output parameters from a dataset generated by that specific model/system, in this case, the QuaLiKiz gyrokinetic eigenvalue solver.[17] QuaLiKiz uses independent and normalized local plasma parameters to calculate the linear electrostatic drift wave and interchange instabilities. These input parameters are the ones that drive and affect the instabilities such as the gradients, collisionality, impurity content, etc. The outputs are the growth rates and associated real frequencies (i.e., the imaginary and real part of the eigenvalue) for the two fastest growing instabilities at 18 different wavenumbers. The benefits of a trained NN is that it can be much faster than the original QuaLiKiz but still give similar results, if the surrogate model is accurate. However, it can only mimic QuaLiKiz properly in the parameter space of the training dataset. Therefore, it is imperative to get a large dataset that include as many of the different parameter domains as possible. The generation of the dataset can be an arduous task, primarily due to fact that the NN need a large amount of data points and the number needed increase exponentially with the amount of input parameters. In this work, we use a subset of a recently created dataset[16,18] containing 3.38×10^7 data entries, with 15 input parameters. Each data entry represents a QuaLiKiz simulation. In this section, we give a brief presentation of the input and output parameters to QuaLiKiz and the dataset, for a more detailed description see Ref. 16. It is important to understand that a surrogate model only take into account the phenomenon that is described in the main model, i.e., if a physical phenomenon is not present in QuaLiKiz, it will not be present in the NN-model. Noteworthy physical limitations in the QuaLiKiz model are: no electromagnetic flux, it only accounts for electrostatic fluxes, bounce averaged trapped electrons, and no geometric effects as QuaLiKiz uses a shifted circular geometry. Therefore, geometric parameters such as triangularity, elongation, etc. are not included in the input parameters to the NNs.

A. Input and output parameters

QuaLiKiz uses certain plasma parameters as inputs, and most of them are normalized and dimensionless. For the simulations in the dataset, light impurities with a charge of less than 10 have been coalesced into one "light impurity species." Similarly, impurities with a higher charge than 10 have been a coalesced into one "heavy impurity species." Hence, the simulations have been performed with four species, one main ion species, two impurity species and electrons, which makes a total of 33 input parameters. However, by using constraints such as quasi neutrality and certain assumptions due to the availability of data for the JET discharges, the number of input parameters can be reduced to 15. The assumption used are:

- Z_{eff} is radially constant throughout the plasma, i.e., $\nabla Z_{eff} = 0$;
- $T_i = T_{imp}$, as the widely available diagnostics measure the temperature of impurity ion species, implying $R/L_{T_i} = R/L_{T_{imp}}$
- The main ion is deuterium, with $Z_i = 1$ and $A_i = 2$.

Here, L_X is the gradient length, in circular geometry $L_X := -(\frac{\partial}{\partial \rho_{tor}} \ln X)^{-1}$, ρ_{tor} is a flux label defined as

$$\rho_{tor} := \sqrt{\frac{\psi_{tor}(r)}{\psi_{tor}(a)}}, \tag{1}$$

where ψ_{tor} is the toroidal magnetic flux and a is the minor radius. The full set of the 15 input parameters in the dataset is presented in Table I. q is the safety factor, $\hat{s}$ the magnetic shear, ν^* the collisionality, α the normalized pressure gradient, M_{tor} rotation Mach number, $R/L_{u_{tor}}$ normalized rotation gradient, and γ_E the E × B-shearing rate. These parameters span a wide range of conditions in the JET-tokamak; hence, the model is valid for analysis at that particular device. However, if the normalized parameters for a discharge at another device are within the confines of the parameters in Table I, the surrogate model will also be applicable. If out-of-range input values were to be encountered post training of the models, they would not be able to

TABLE I. Input parameters to the NNs and their range. All parameters are normalized and dimensionless.

Dimensionless parameter	Associated physical parameter	Range min	Range max
ρ_{tor}	r	0.10	0.93
q	q	0.79	3.99
$\hat{s}$	∇q	−0.48	3.99
R/L_{T_e}	∇T_e	−4.97	24.99
Z_{eff}	Z_{eff}	1.00	3.97
$log_{10}(\nu^*)$	n_e	−1.499	0.49
R/L_{n_e}	∇n_e	−4.97	9.98
T_i/T_e	T_{imp}	0.500	1.75
R/L_{T_i}	∇T_{imp}	−4.99	19.98
$N_{imp,light}$	$n_{imp,light}$	0.0002	0.049
α	B_0	−0.047	1.499
M_{tor}	Ω_{tor}	−0.048	0.99
$R/L_{u_{tor}}$	$\nabla \Omega_{tor}$	−0.99	4.98
γ_E	$\nabla^2(n_i T_i)$	−1.49	0.49

output reliable values. This is partly due to the complexity and non-linearity of neural networks, which can lead to unexpected and disproportionate changes in the predictions for small changes in the input values.

The outputs, the eigenvalues, are calculated as dimensionless values. QuaLiKiz calculates the two fastest growth rates at 18 different spatial scales separately ranging from ion scales $k_y\rho_s < 1.5$ to electron scales $k_y\rho_s > 1.5$. Here, k_y is the wavenumber in the poloidal direction, and it is normalized with $\rho_s = c_s/\Omega_c$ where c_s is the ion sound speed and Ω_c is the cyclotron frequency. The 18 normalized wavenumbers are: 0.1, 0.175, 0.25, 0.325, 0.4, 0.5, 0.6, 0.8, 1, 2, 3.5, 6, 10.5, 15, 19.5, 24, 30, and 36. At each wavenumber, there are four outputs, the fastest growth rate and its associated real frequency, and the second fastest growth rate and its associated real frequency. A negative real frequency represents motion in the ion drift direction, usually associated with the ion temperature gradient (ITG)-mode, and a positive real frequency represents motion in the electron drift direction, usually associated with the trapped electron-mode (TEM) and electron temperature gradient (ETG)-mode. In total, there are 72 outputs for each QuaLiKiz simulation, and they are normalized with c_s/a. In Fig. 1, an example of a growth rate spectrum for the strongest instability is shown.

The outputs in the dataset showed that QuaLiKiz rarely found a second fastest growing instability, only 0.05% of the cases. This can be compared to 22.24% for the fastest growing instability. When QuaLiKiz does not find an instability, it sets the output for the growth rate and the real frequency to zero. The few non-zero data points for the second instability made us decide that in this study to focus only on the fastest growth rate. The small amount of non-zero data points would make it hard for the NNs to learn the patterns properly.

For the strongest instability, the rate of unstable solutions varied between the different wavenumbers. We display the rate of the unstable samples in the training set per k_y-index in Fig. 2. The k_y-indices at ion scales are 0–8 and electron scales 9–17, and the values for the two indices at the boundary region, k_y-indices 8 and 9 are $k_y\rho_s = 1$and2. We have the highest rate of unstable solutions for k_y-indices 2, 3, 4,

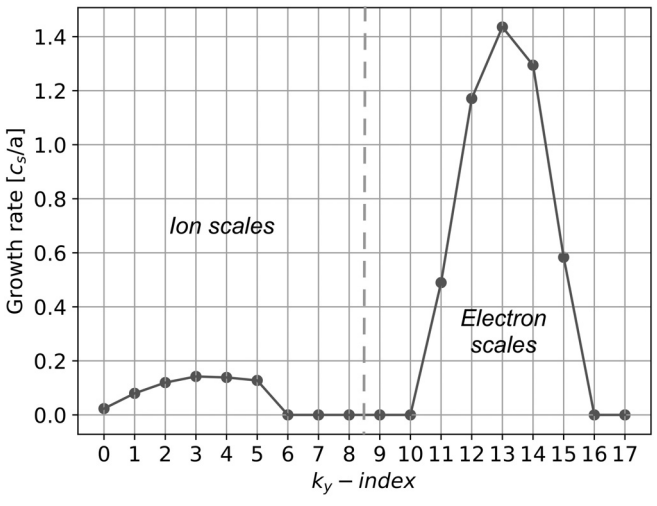


FIG. 1. An example of a growth rate spectrum (only growth rate for strongest instability), which is the output of the regression neural networks in this work.

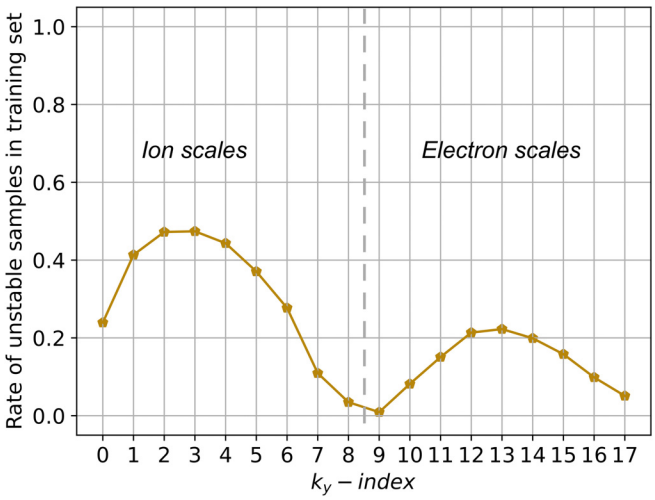


FIG. 2. Rate of unstable solutions per k_y-index for the strongest instability in the dataset.

and these unstable solutions represent the ITG-mode and TEM. At higher k_y-indices 12, 13, 14, we have an increased rate of unstable solutions representing the electron temperature gradient (ETG)-mode. However, at the meso-scales, the boundary region between ion- and electron-scales, and for the largest wavenumber, we have a low amount of unstable solutions. From a machine learning perspective, this is an example of a an unbalanced dataset, in particular for certain k_y-indices, which can cause problems for the NNs, which we will discuss in Sec. III. Later in this work, we employ a weighted loss function to solve this issue.

B. QuaLiKiz dataset

To be able to create a surrogate model for the eigenvalues a sufficiently large dataset of simulations with QuaLiKiz is needed. In this work, we have used a large dataset based on experimental values from the JET-experiments. The datasets were created from 2135 discharges, both from quasi steady state- and transient-scenarios and a total of 12 328 time windows were selected.

In certain cases, all necessary data were not available and certain assumptions were needed;

- The Z_{eff} contribution of the light impurity did not exceed 0.2 if insufficient impurity information is provided
- $M_{tor} = R/L_{u_{tor}} = \gamma_E = 0$ if no plasma rotation measurements are available
- $T_i = T_{imp} = T_e$ if no ion temperature measurements are available
- $Z_{eff} = 1.25$ if no line-integrated effective charge measurements are available.

The extracted experimental data were used to populate the dataset as input to QuaLiKiz, at nine equidistant radial positions between $\rho_{tor} = 0.1$ and 0.9. The drift waves instabilities, in general, and QuaLiKiz, in particular, are sensitive of certain plasma parameter presented in Table I. Therefore, we have expanded the dataset for $\{R/L_{n_e}, R/L_{T_e}, R/L_{T_i}, \hat{s}, \gamma_e\}$. The gradients are especially important

as the free energy in them drives the drift wave instabilities. Hence, a five-point expansion was performed for $\{R/L_{T_e}, R/L_{T_i}\}$ and a three-point expansion in $\{R/L_{n_e}, \hat{s}\}$. All experimental data points that had rotation data available were duplicate and assuming no rotation to improve the NN interpolation for the rotation. Additionally, a three point expansion of $\{\gamma_e\}$ was performed. The total number of data points is roughly 3.7×10^7.

C. Data cleaning

There are benefits and drawbacks to using experimental data to populate the input parameter space compared to using a predefined region. There might be uncertainties and discrepancies in the measurements which propagate to the dataset. For a dataset of this size, it is not possible to manually verify every entry. Therefore, there are bound to be some erroneous data points, but we can mitigate this by pruning the dataset. Data points with extreme values in the input and/or the output have been eliminated from the dataset. The range of the input parameters post cleaning are displayed in Table I. One benefit with using experimental data as method to populate the input space is that it eliminates nonphysical conditions.

III. METHOD AND RESULTS

In this section, we describe the architecture of the models and present their predictive capabilities. For the NNs, we have used TensorFlow[19] and fully connected feed forward NNs (FFNNs).[20] As all inputs and outputs are properly labeled, we have performed supervised learning to optimize the NNs.

A. Definitions

The following list contains brief descriptions of key machine learning related concepts and methods, and these are more thoroughly explained in Ref. 20.

- R^2—the coefficient of determination. This is commonly used as a metric to measure model performance. It is defined as

$$R^2 = 1 - \frac{\sum_i (y_i^{true} - y_i^{pred})^2}{\sum_i (y_i^{true} - \bar{y}^{true})^2},$$

  where y^{true} represents the dataset values of the output parameter, y^{pred} represents the predicted values of the output by the model, and $\bar{y}^{true}$ represents the mean value of the dataset values of the output. $R^2 = 1$ represents a perfect model with perfect predictions on all data points, and $R^2 = 0$ represents a model with equally poor predictive capabilities as a model that always outputs the mean value of the given output parameter.
- Recall—in a binary classification task, the recall evaluation metric is defined as

$$\text{Recall} = \frac{\text{True positives}}{\text{True positives} + \text{False negatives}}.$$

  Note that in a binary classification task, "positives" refer to the class that is being evaluated. In this work, we evaluate both classes in the binary classification tasks, which means that positives are not only referring to the positive growth rates in this work. When the recall of the stable class is evaluated, "True positives" refers to the number of correctly classified stable solutions, where the growth rate is 0. The same reasoning applies for the precision which is defined below.
- Precision—in a binary classification task, the precision evaluation metric is defined as

$$\text{Precision} = \frac{\text{True positives}}{\text{True positives} + \text{False positives}}.$$

- F1-score—in a binary classification task, the F1-score evaluation metric is defined as

$$\text{F1} - \text{score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}.$$

- Loss function—the trainable parameters in a machine learning model are iteratively adjusted during the training to lower a loss function. A typical loss function for regression tasks is the mean squared error (MSE)

$$\text{MSE} = \frac{1}{N}\sum_i^N (y_i^{true} - y_i^{pred})^2.$$

- Binary cross-entropy—a loss function that is commonly used in binary classification tasks. It is defined as

$$\text{Binary cross-entropy} = -\frac{1}{N}\sum_i^N y_i^{true} \cdot \log(y_i^{pred}) + (1 - y_i^{true})(\log(1 - y_i^{pred})).$$

- Activation function—the function that calculates the output of the individual nodes in a neural network. A common activation function is the Rectified Linear Unit (ReLU), which is defined as

$$\text{ReLU}(x) = max(0, x).$$

  Another common activation function is the Sigmoid function, which is common for the output layer in binary classification models

$$\sigma(x) = \frac{1}{1 + e^{-x}}.$$

- Epochs—one epoch refers to one complete pass through the entire training dataset during the training phase.
- Optimizer—the algorithm used to adjust the trainable parameters of a model in order to lower the value of a loss function. A common optimizer is Adam, which combines two key techniques for optimization: momentum and an adaptive learning rate.
- Learning rate—This controls the step size of the update of the trainable parameters during the training phase.
- Dropout—the key idea behind dropout is that by randomly disabling a set rate of neurons during training, the network cannot rely too heavily on any particular neuron or learn complex co-adaptations between neurons. This encourages the network to learn more robust and generalized features, reducing the risk of overfitting.
- Batch size—the number of training examples (data points) that are processed in parallel before the trainable parameters of a model are updated.

- MinMax scaling—linear scaling of the data to be within a certain range, such as [0,1].
- Standard scaling—scaling of the data by removing the mean and scaling to unit variance

$$z = \frac{x - u}{s},$$

where z is the scaled value, x is the original value, u is the mean value of the distribution of the considered parameter, and s is the standard deviation of the distribution of the considered parameter.

B. Models

As previously mentioned, this study will only focus on the strongest growth rate, which means that for each data point, we have 15 input parameters and 36 outputs (one growth rate and real frequency for each of the 18 wavenumbers). We have designed several different neural network based models to evaluate which is the most suitable for this problem. The outline of this study is as follows:

- Architecture of NNs
- One NN per k_y-index or one NN for all k_y-indices
- Splitting the task; classifier and regression with weighted loss function
- Classification of ITG and TEM

C. Evaluation metric

The distribution of the output data for each k_y is generally a combination of a Gaussian-like distribution for the unstable solutions and a large spike at 0 for the stable solutions. For instance, at k_y-index 3, the growth rate of the unstable entries across the dataset are spread out and resemble a Gaussian distribution if it were to be plotted in a histogram, and the stable entries with a growth rate of 0 adds a narrow peak at 0 in this distribution. This leads to different options regarding evaluation. For regression tasks with a linear output layer, R^2 is commonly used for evaluation, in particular when the output data are evenly (or Gaussian-like) distributed. However, in this case, the large spike at 0 (stable solutions) will bias/corrupt the R^2 value if the entire output domain is evaluated using this metric. Or rather, the performance on the unstable samples will be shrouded by the many stable data points. For this reason, we evaluate the following combination of metrics to get a more detailed interpretation of the performance of the different models in this work:

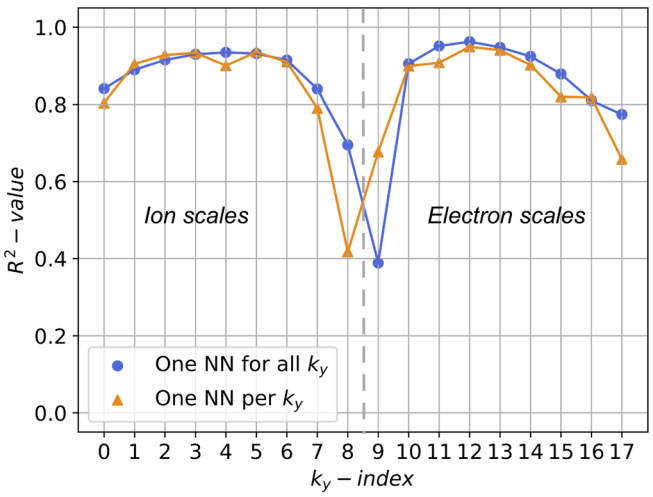
- The R^2 value of all unstable labels in the test set for each k_y.
- The recall and precision for classification of stable/unstable data, which is evaluated for both classes. Effectively, the recall and precision constitute the F1-score, however separately they provide a more detailed picture of the classifier since the F1-score represents a compressed version of the recall and precision. This metric is also later used for the ITG/TEM classification in this work.

D. Architecture of NNs

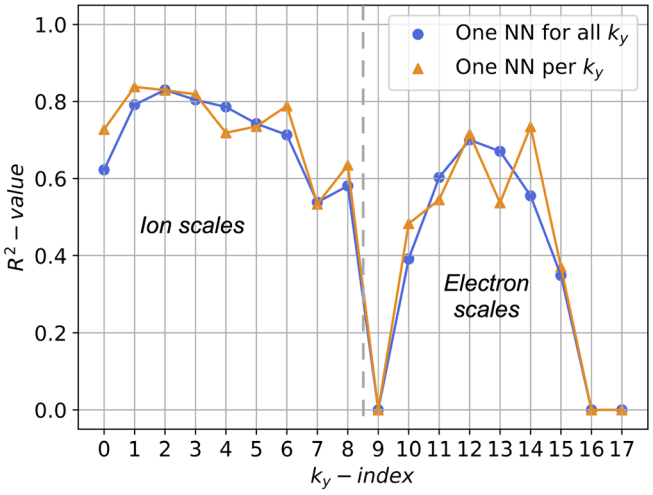
We performed a hyper-parameter search to find the most suited neural network for the problem. The search is based on the first model presented in this work, which is described in Sec. III E, and the found hyper-parameters are applied to all of the following tests in this work. It is possible that the hyper-parameters can be slightly tweaked further to improve the score on the following tests, although we defer the exploration of this topic to future investigations. The following hyper-parameter combination represents the optimal configuration: optimizer: Adam, loss function: MSE, learning rate: lr = 0.001, and batch size = 256. We did set a roof of a total of 400 000 trainable parameters (200 000 active trainable parameters for each training iteration using dropout). With this restriction, we found the best configuration to be two hidden layers with 600 nodes and 50% dropout in the second hidden layer, with the activation function: ReLu. The output of the NNs that perform regression in this work is a linear layer. For the classification tasks in this work, binary cross-entropy is used as the loss function, and the sigmoid function is used for the output layer. We also employ MinMax scaling to all the input and output parameters, where each distribution is scaled to be between 0 and 1. The reason for not using standard scaling is that the distribution of the output parameters is not Gaussian due to the large amount of stable solutions in the dataset which has a growth rate and real frequency of exactly 0. In the following tests, we did not see an increased performance by instead scaling the input parameters with standard scaling. For the training/validation, we used 5×10^6 entries randomly sampled from the cleaned dataset, as well as 3×10^6 randomly sampled entries from the cleaned set for the testing. The total number of training iterations for each model was set to approximately 20 000, which corresponds to 1 epoch with the full training set considering the batch size of 256.

E. One NN per k_y-index or one NN for all k_y-indices

We made two different models to determine if it is better to predict the eigenvalues with one NN for all k_y-indices or to have one NN for each k_y-index. The eigenvalue problems in QuaLiKiz are independent, but it is still of interest to explore if there are any benefits associated with predicting for all k_y-indices at once since similar physics govern for the different k_y-indices. Each NN in the multi-NN model has the same number of nodes as the NN for all k_y-indices except in the output layer. These models are trained on the entire training set including both stable and unstable solutions, and for this specific test, we are not implementing stable/unstable class balancing in either model to allow for fair comparison. This is because balancing strategies are not straightforward for the model that predicts the output at all k_y simultaneously, where each k_y-index has a different amount of unstable solutions. Additionally, out of the two evaluation metrics, in this particular test, we can only evaluate the R^2 value for the unstable labels. This is because the models have linear output layers and can thus not perform classification since no exact zeroes are obtained in the predictions. However, the test still serves as a useful comparison before we continue by splitting the task into a classification task and a regression task. The R^2 values are displayed in Fig. 3, the growth rates at the top, and the real frequency below. The model that has one NN for all k_y-indices is the blue curve with circle markers, and the model with one NN per k_y-index is the orange curve with triangle markers. From the figure, it is clear that the two different models have similar performance for all k_y-indices and both struggle at k_y-indices 8, 9, 16, and 17. At these indices, we have unbalanced data, which was shown in Fig. 2. The models are simply not encouraged to learn the patterns of the unstable solutions here since a low value of the loss function can be achieved by blindly predicting low values. The results for the model



(a) Growth rate



(b) Real frequency

FIG. 3. Comparison between using one NN per k_y-index (orange triangle) and one NN for all k_y-indices (blue dot). The models show a similar R^2 value on average. Both models struggle at k_y-index 8, 9, 16, and 17 where we have unbalanced data. The R^2 value for the model with one NN per k_y is more jagged across the spectrum. The R^2 value is generally higher for the growth rate (a) compared to the real frequency (b).

with one NN for each k_y-index are more jagged as its NNs are independent of each other, but overall, the results indicate that there is no major drawback from using the model with one NN for each k_y-index, which will simplify class balancing in the following tests.

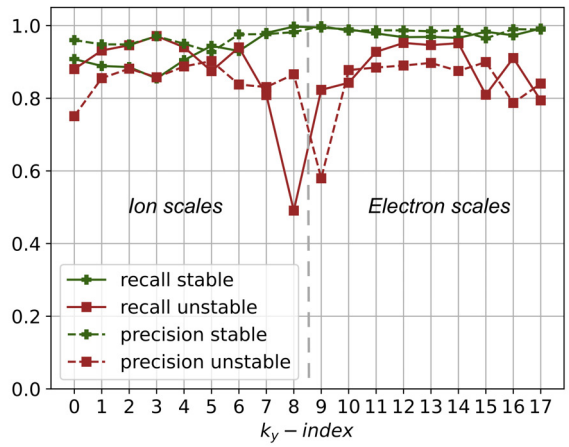
F. Splitting the task; classifier and regression

As mentioned, a limitation of these pure regression models is that neither of them ever predict exactly zero, which represents a stable solution from the eigenvalue solver, and as a large part of the dataset is exactly zero, it is imperative to rectify this to enable classification. This problem is an artifact from the output of the gyrokinetic eigenvalue solver in QuaLiKiz. The solution to the dispersion relation can yield both positive and negative values for the growth rates. However, QuaLiKiz truncates the output for the negative values to zero. The truncation is performed as negative growth rates do not contribute to the turbulent fluxes; nonetheless, the truncation means a loss of information which could have been useful during the training process of the NNs.

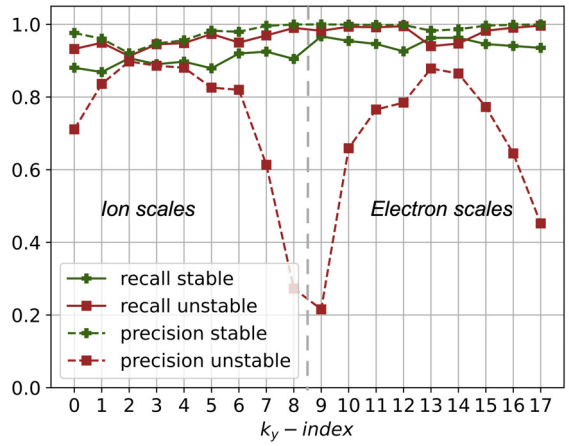
In this case, without the information of negative growth rate, we solve this problem by introducing a classifier, which determines if there is a stable or unstable solution for given input parameters. If we have a stable solution, we set the growth rate and real frequency to zero. In addition, we introduce a regression model that is only trained on unstable solutions and used post training when the classifier predicts an unstable solution. In other words, the classifier and regression NNs are trained separately. Both these models employ a dedicated NN for each k_y-index.

To balance the training, the regression model NNs have their number of epochs adjusted for each k_y-index to get same amount of training iterations as previous runs (∼20 000 iterations). This is because the amount of unstable data is different for each k_y-index; thus, the number of iterations per epoch varies between the k_y-indices since the batch-size is constant. For the classifier, we investigate the effect of balancing the loss function, such that the unstable class gets a weighted loss with the same ratio as the ratio between the number of stable/unstable entries for each k_y-index. In practice, this means that for k_y-indices with a low number of unstable entries, the loss function will be significantly higher per unstable entry. The neural network with the balanced loss function will thus be encouraged to not neglect the unstable entries, since these will contribute significantly to the total loss function of the entire dataset even if they are few in numbers. Since the balancing strategy is only active during the training, it is not expected to limit predictive capabilities of datasets from other configurations with different ratios of stable/unstable entries post training. Rather, the entire point of the balancing procedure is to not neglect uncommon unstable entries for certain k_y-indices that might be more common for those k_y-indices in other datasets. These two balancing strategies would be significantly more complicated for the model with one NN for all k_y-indices. Therefore, since the previous test showed similar results for the models with different amount of outputs, we perform this test using the model with one NN for each k_y-index.

In Fig. 4, we compare the recall and precision of two classification models, one with and the other without weighted loss, which we refer to as the balanced and unbalanced classifier, respectively. The top figure displays the metrics for the unbalanced classifier, and the bottom figure displays the metrics for the balanced classifier. As expected, the unbalanced classifier generally shows a high recall and precision for the stable class since the data are dominated by stable solutions, as well as a lower precision and recall for the unstable class, in particular for the k_y-indices with few unstable data. The balanced classifier manages to raise the recall of the unstable data but shows a lower precision of the unstable class. The lower precision is not surprising, and not necessarily as problematic as it may seem since for a highly imbalanced test set, even a small fraction of false unstable predictions will lead to a high number of false unstable predictions in relation to the total number of unstable data. In other words, the precision metric is highly biased by class imbalance which should be considered when evaluating this metric.

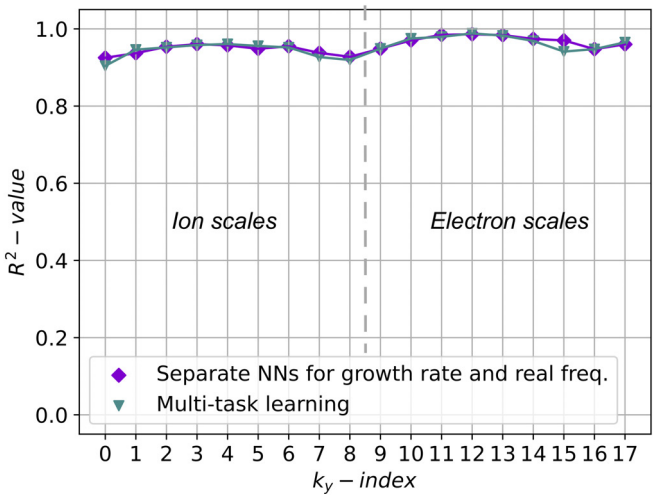


(a) Classifier trained using no weighted loss to balance classes.

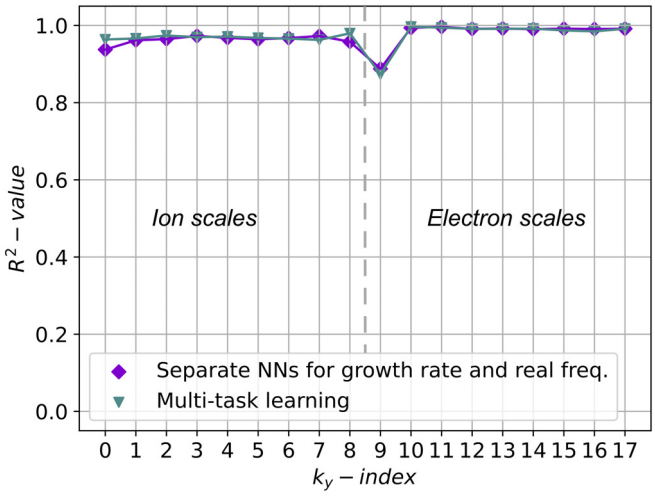


(b) Classifier trained using weighted loss to balance classes.

FIG. 4. Comparison between not using (a) and using (b) a weighted loss function in the stable/unstable solution classifier. Overall, the recall of the unstable class (red solid square) is significantly improved for the k_y-indices with few unstable data by using a weighted loss. This means that the model has improved in correctly classifying the entries that actually are unstable while maintaining a high recall on the stable class (green solid plus). It is not surprising that the precision is lower for the unstable class (red dashed square) when using a balanced loss function since this metric is highly biased by unbalanced test data. The y axes show both the recall and precision, both of which are unitless.



(a) Growth rates



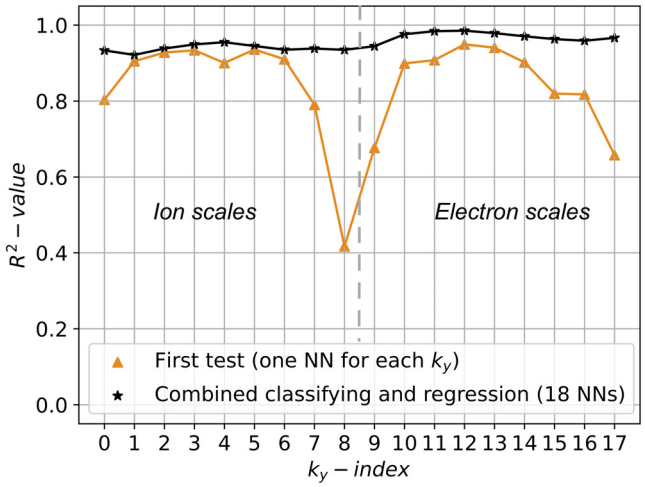
(b) Real frequencies

FIG. 5. R^2 values of regression models that are only trained and evaluated on unstable data. Two models are compared, where the growth rate (a) and real frequency (b) either are predicted together (turquoise, triangles) or separately (purple, rhombus). Overall, the R^2 value is close to 1 for almost all k_y-indices. There is no difference between predicting the growth rate and real frequency separately or together (the latter referred to as multi-task learning).

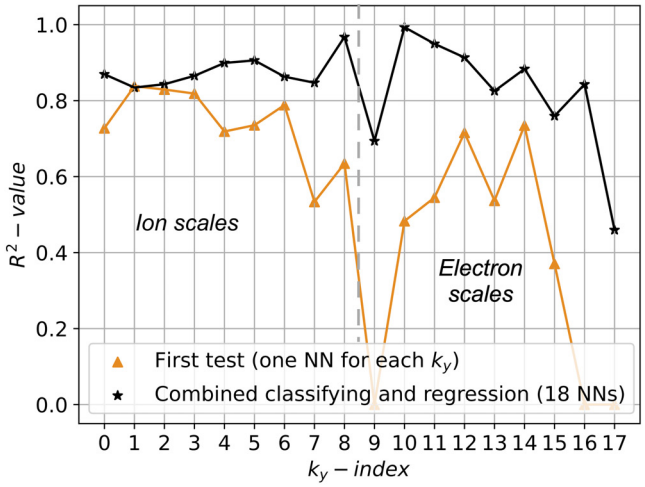
Before we evaluate the classifier and regression model together, we evaluate the regression model by itself in Fig. 5. Here, we also investigate if there is any benefit or drawback of predicting the growth rate and real frequency together (turquoise, upside-down triangles) or separately (purple, rhombus). We refer to the test when predicting the growth rate and real frequency together as "multi-task learning." The results show high R^2 for all k_y-indices for both the growth rate and real frequency, and there is no difference when predicting the growth rate and real frequency separately or together. There is a slightly lower R^2 value for k_y-index 9 at the meso-scale for the real frequency.

These R^2 values are informative since they tell that for cases where it is known that there is an unstable solution, the regression model performs exceptionally well. This is expected since it is not trained on stable data and thus not partly biased to making predictions in the stable domain, which the first models displayed in Fig. 3 are.

For a complete evaluation, we compare this combined balanced classifier and regression model with the model with one NN for each k_y-index from Sec. III E in Fig. 6, which we refer to as the "First test." The new combined model is the black curve with star markers, and the model with one NN per k_y-index is the orange curve with



(a) Growth rates



(b) Real frequencies

FIG. 6. Comparison between: (1) a combined classifier and regression model with a weighted loss function and one NN per k_y-index (black star) and (2) the regression model in the first test with one NN for each k_y-index with no class balancing strategies (orange triangle). The R^2 values of the growth rate (a) and real frequency (b) are higher for the combined classifier and the regression model. The combined model for the real frequency still shows a dip in R^2 at k_y-indices 9 and 17, although this is likely an artifact from how wrongly classified unstable entries affect the R^2-metric for these indices, since previous figures show high classification score as well as high regression performance for all k_y-indices.

triangle markers (same coloring and marker as when it appeared in Fig. 3). Here, the unstable data that the combined model wrongly classifies as stable data are included in the R^2-calculation in order to perform the comparison on the exact same test sets. The combined model has a higher R^2 value for all k_y-indices compared to the first model, both for the growth rates and real frequencies.

For the combined model predictions of the real frequency, we see that the R^2 value is lower for k_y-indices 9 and 17. To investigate this, we show the output for the combined classifier and regression model compared with the true values in the test set in scatter density plots in Fig. 7 for three k_y-indices. Here, we also show histograms of the distribution of false unstable predictions, as well as the dataset values for the false stable predictions. The k_y-indices are from the left to right: 3, 9, and 17. Growth rates are at the top row and real frequencies at the bottom. The three different k_y-indices are at different scales, k_y-index 3 is at ion scales, 9 at meso-scales, and 17 at the smallest scale resolved in the QuaLiKiz simulations in the dataset. The scatter plots only display the unstable (non-zero) entries in the dataset, as including stable solutions would lead to extreme intensity at (0,0). For k_y-index 3, the real frequencies have a large amount of data points both positive and negative, as there are two different instabilities at these scales. The negative real frequency represents the ITG-mode and positive TEM. By the intensity of the scatterplot, it is clear that there are more ITG-entries in the dataset. For the rest of k_y-indices, they have primarily entries with positive real frequency, at these scales it represents the ETG-mode. The figures visualizes how well the regression models perform for instances where there is a correct classification. The thin vertical lines at $x = 0$ in all of the figures are the cases there the classifier makes an wrong classification, predicting a stable solution when it is unstable in the dataset. This is also the reason why the R^2 value is lower for the real frequency compared to the growth rate in general. The regression models on their own have similar accuracy, but the vertical lines generally deviate more from $y = 0$ for the real frequency. This is particularly clear at k_y-index 17 for the combined model. Here, the vertical line representing incorrect classification leads to a larger decrease in R^2 since the line at $x = 0$ is not centralized around $y = 0$. In other words, some indices with distributions centralized further away from $y = 0$ are more penalized at wrong classification due to an artifact with the R^2-metric for this problem. Thus, it is important to consider all the different metrics to get a more detailed understanding of the model performance, including the classification score, the performance of the regression models when not considering false classification, as well as the performance of the combined model and the visualizations in Fig. 7. Overall, the histograms indicate that the false unstable predictions show a similar distribution as the general distribution of the unstable domain, which is displayed in the scatter density plots. The histograms also indicate that overall, when the classifier wrongly classifies an entry as stable, at least the database value of the growth rate is small.

G. Classification of ITG and TEM

There are three instabilities present in QuaLiKiz, two at the ion scales, the ITG-mode and TEM, and one at the electron scale, the ETG-mode. Here, we investigate if a classifier can predict if an unstable output is associated with the ITG-mode or TEM. Important to note is that QuaLiKiz only include the "pure" versions of the instabilities; however, mixed modes of ITG/TEM exists which has a real frequency close to zero. Since ITG and TEMs are only present at the ion scales, we build a class balanced classifier at (k_y-index ≤ 8). Fortunately, the two instabilities are easily distinguishable through the real frequencies; positive real frequency indicates a TEM and a negative real frequency indicates ITG-mode. This is generally true, but for every step density gradients, the ITG-mode can move in the electron direction, i.e., have a positive real frequency.[22] Nonetheless, these cases were deemed so rare no additional considerations were

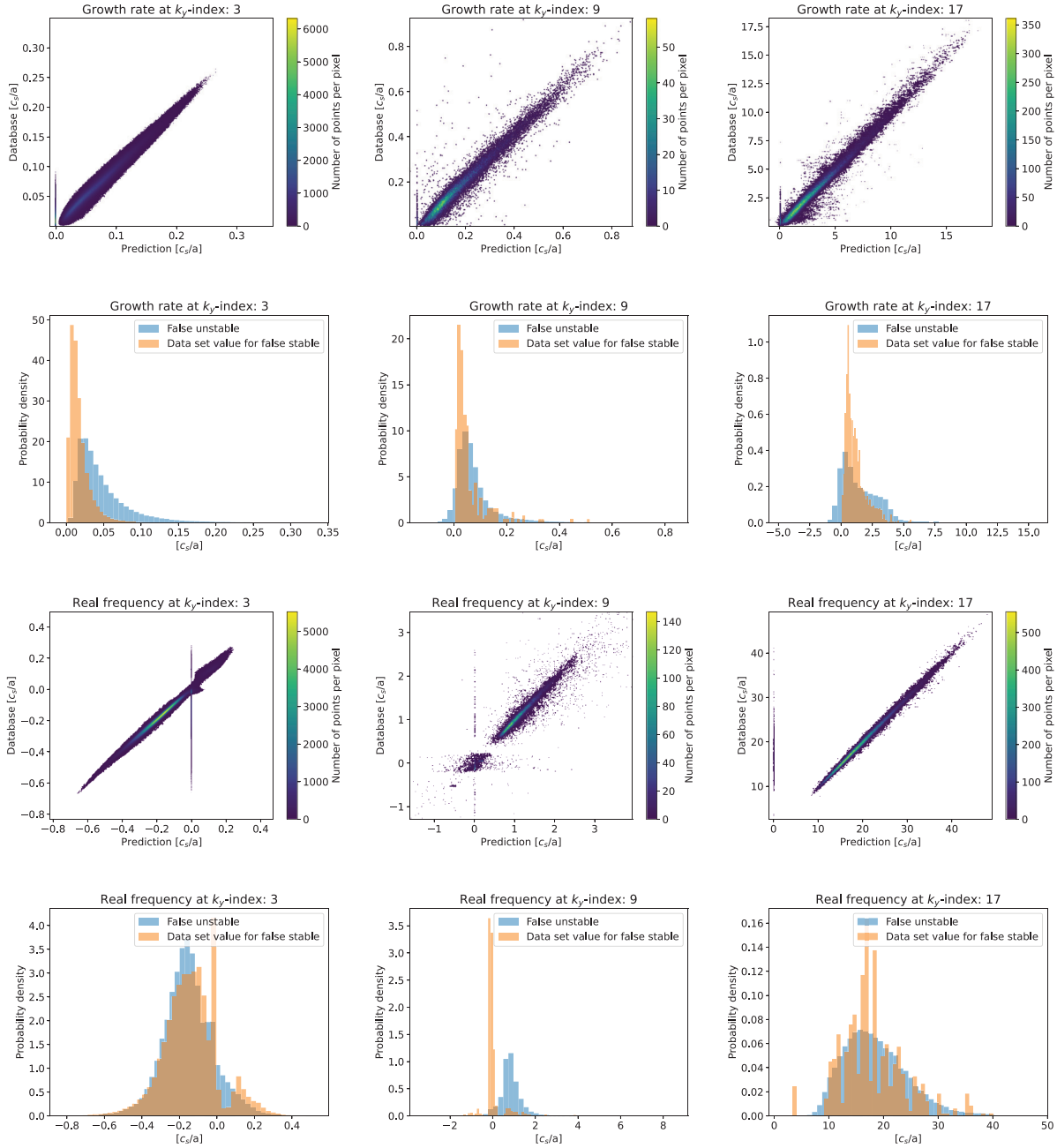


FIG. 7. Scatter density plots of the predicted eigenvalues vs the true values of the unstable solutions in the test set, and histograms of the distribution of false unstable predictions, as well as the data values for the false stable predictions. The k_y-indices are from the left to right: 3, 9, and 17. Growth rates are at the top row and real frequencies at the bottom. The plots show that the model accurately predicts the growth rate and real frequency, and how falsely predicted unstable entries create vertical lines at $x = 0$. These vertical lines have different impact on the R^2 value for the different k_y-indices since they are centered around different values. An important conclusion is that the model, when wrongly predicting a stable solution, at least predicts a low growth rate.

taken for them. Note that the real frequency is not an input to the classifier, but rather a tool for labeling the data. The recall and precision of the two instabilities are displayed in Fig. 8. The recall is high for both instabilities; however, the precision for the TEM is lower for the higher k_y-indices. Similarly to the case for the unstable/stable classifier, this is not surprising since the precision is greatly affected by class imbalance in the evaluation set for a classifier that has been balanced during training.

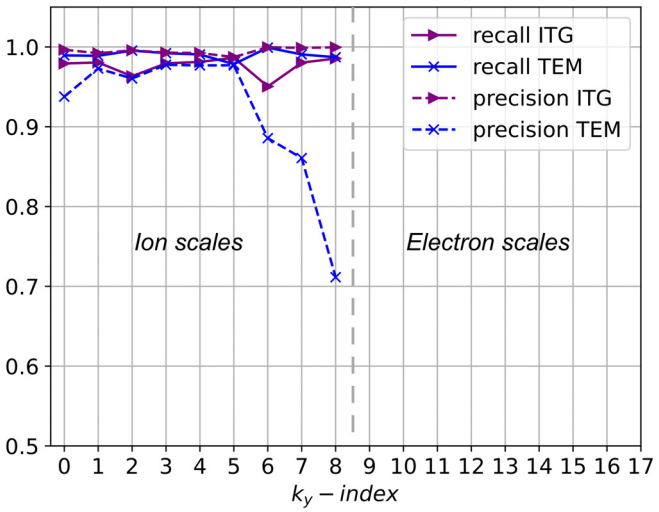


FIG. 8. The recall and precision of the ITG/TEM classification model. The classifier generally shows a high score. The precision for TEM (blue dashed cross) is lower for k_y-indices 6, 7, and 8, although this is because there are far fewer TEM entries for these indices (precision is skewed by unbalanced testing data). The y axes show both the recall and precision, both of which are unitless.

As the classifier showed high recall, we created a model for the prediction of eigenvalues with two classifiers and two regression models at ion scales. First, we used the same classifier as in Sec. III F to determine if the input parameters yield a stable or unstable solution. If the output is stable, again it sets the growth rate and real frequency to zero; otherwise, it is passed forward to the ITG/TEM-classifier. This classifier determines if the unstable solution corresponds to an ITG-mode or a TEM. For both of the instabilities a dedicated regression model, with a NN per k_y-index, was trained to predict the growth rate and real frequency. However, the R^2 values for this model with two classifiers and dedicated regression models for the ITG-mode and the TEM were lower than for the model with one stable/unstable classifier and one regression model for both instabilities presented in Fig. 6. Even though each of the separate parts (ITG/TEM classifier, ITG- and TEM-regression models) showed good performance, the added complexity reduced the overall performance. An explanation for this is that the occasions where the ITG/TEM-classifier makes wrong predictions and the wrong regression model is used, it degrades the overall R^2 values more compared with what is gained by having dedicated regression models for each instability.

IV. SUMMARY AND OUTLOOK

In this paper, we have presented a framework for a neural network surrogate model for the eigenvalues trained on a large dataset from the QuaLiKiz gyrokinetic eigenvalue solver. The dataset is populated with experimental data from JET discharges and the model predicts the growth rate and real frequency for the strongest instability for 18 different wavenumbers, ranging from the ion to the electron scales. When the eigenvalue solver in QuaLiKiz finds a stable solution to the dispersion relation, it sets the growth rate and real frequency to zero. As the dataset was unbalanced in terms of the stable/unstable ratio, proper data preparation and loss function weighting was shown to be vital for good performance over the full domain of wavenumbers.

The model that had the best performance was a combined stable/unstable classifier and a regression model. First, a classifier predicts if the input parameters are associated with a stable (zero growth rate) or unstable (non-zero growth rate) solution. If it is stable, it sets the growth rate and real frequency to zero; otherwise, it uses a regression model to predict the growth rate and real frequency. The metrics used to evaluate the models were recall and precision for the classification and R^2 for the regression, which is a metric related to the distance between a perfect prediction and the actual prediction. However, only unstable labels in the test set were included in the calculation of R^2, as the many stable solutions skewed the metric. Overall, the R^2 value for the unstable data were a useful metric to compare the performance of different models, even if the metric was affected by wrongly classified true unstable data. A more detailed understanding of the performance of the best model could be obtained through the accuracy, recall, and precision of the individual neural networks in the combined classifier and regression model.

We saw that the classifiers with and without weighted loss excel at different areas. With weighted loss, the precision was lower, but the recall was higher for the unstable solution at meso-scales, as a consequence by a higher relative number of wrongly classified unstable solutions due to class imbalance. As neither of these two models, with and without weighted loss, were superior for all of the four classification metrics displayed in Fig. 4, one needs to decide what to prioritize when creating the final surrogate model. One option is to use weighted loss and get more wrongly predicated unstable solutions, and the other option is to train without weighted loss and get more wrongly predicated stable solutions. When using the surrogate eigenvalue model with a saturation rule, this translates to if one would rather accept occasionally getting too much or too little flux. Thus, the alternative that provides the best model is indeed partly a matter of preference depending on the application that is considered. However, as the difference is at the meso-scales, we do not generally expect this to have a significant impact on the total fluxes.

Improvements of the model might be achieved by gaining access to the information of the negative growth rates, QuaLiKiz truncates negative values to zero. This additional information could be beneficial for the NNs to learn how "far" in input parameter space they are from an unstable solution. Another area of improvements might be to correct for crashes in the eigenvalue solver in the dataset. On few occasion, the gyrokinetic eigenvalue solver in QuaLiKiz crashes due to numerical issues, and it sets the output to zero for growth rate and real frequency. It has been suggested in Ref. 21 that ensemble learning might alleviate this problem. However, the crashes were deemed too few to be significant to affect the overall performance for this first framework. Additionally, in terms of increased performance of the best model in this work, further improvements might be obtained through more extensive hyper-parameter search for each individual sub-task in the combined model, since the hyper-parameters in this work, which were kept constant when comparing different models, were based on the hyper-parameter search of the first model that performed regression on both stable and unstable data. The performance may also be slightly improved by removing dropout, in particular for the regression tasks, although in this work, we choose to use dropout

to overall lower the risk of overfitting. Finally, a potential challenge with using independent neural networks for each k_y-index is that this may lead to slightly less smooth spectra if the networks learn slightly different patterns between the input and output parameters. This is indicated by the slightly more jagged R^2-score when using one neural network for each k_y-index compared to the model with one neural network for all k_y-indices. Ensuring smoothness will help avoid occasional misidentification of the k_y-index with the maximum growth rate, which is important for the saturation rule. Future work thus involves strategies to ensure smoothness, either through solving the class imbalance problem for the model with one neural network for all k_y-indices, or through other smoothing techniques applied to the outputs of the individual neural networks. Future work also includes an investigation if there are physics based constraints that could be added to the loss function to enhance the robustness of the model.

In addition to swiftly providing information about the most dominant instability for specific plasma conditions, eigenvalues also offer a pathway for extrapolating transport fluxes. As the linear physics of the eigenvalues translate between different devices. With eigenvalues, multiple saturation rules could be used to evaluate the turbulent fluxes for present and future devices.

ACKNOWLEDGMENTS

This work is a continuation of a B.Sc thesis by Borsander *et al.* at Chalmers University of Technology,[21] which was supervised by A. Gillgren and E. Fransson.

This work, as well as the supervision of the aforementioned students, was partly funded by the EUROfusion Enabling Research Project ENR-MOD.01.FZJ "Development of machine learning methods and integration of surrogate model predictor schemes for plasma-exhaust and PWI in fusion." This work has also received funding from the Swedish Research Council with the diary No. 2020-05465.

AUTHOR DECLARATIONS

Conflict of Interest

The authors have no conflicts to disclose.

Author Contributions

Emil Lars Arvid Fransson: Formal analysis (equal); Investigation (equal); Methodology (supporting); Project administration (equal); Software (supporting); Supervision (equal); Validation (equal); Writing – original draft (lead). **Andreas Wilhelm Magnus Gillgren:** Formal analysis (equal); Investigation (equal); Methodology (lead); Project administration (equal); Software (equal); Supervision (equal); Validation (equal); Visualization (lead); Writing – original draft (supporting). **Aaron Ho:** Conceptualization (lead); Data curation (lead); Validation (equal). **Jonathan Borsander:** Investigation (supporting); Methodology (supporting); Software (equal). **Oscar Lindberg:** Investigation (supporting); Methodology (supporting); Software (equal). **Walter Rieck:** Investigation (supporting); Methodology (supporting); Software (equal). **Malte Åqvist:** Investigation (supporting); Methodology (supporting); Software (equal). **Pär Strand:** Funding acquisition (lead); Project administration (supporting); Supervision (supporting).

DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding author upon reasonable request.

REFERENCES

[1]P. C. Liewer, "Measurements of microturbulence in tokamaks and comparisons with theories of turbulence and anomalous transport," Nucl. Fusion **25**, 543 (1985).
[2]W. Horton, "Drift waves and transport," Rev. Mod. Phys. **71**, 735 (1999).
[3]C. Bourdelle, J. Citrin, B. Baiocchi, A. Casati, P. Cottier, X. Garbet, F. Imbeaux, and JET Contributors, "Core turbulent transport in tokamak plasmas: Bridging theory and experiment with QuaLiKiz," Plasma Phys. Controlled Fusion **58**, 014036 (2016).
[4]J. Citrin, C. Bourdelle, F. J. Casson, C. Angioni, N. Bonanomi, Y. Camenen, X. Garbet, L. Garzotti, T. Görler, O. Gürcan *et al.*, "Tractable flux-driven temperature, density, and rotation profile evolution with the quasilinear gyrokinetic transport model QuaLiKiz," Plasma Phys. Controlled Fusion **59**, 124005 (2017).
[5]See http://qualikiz.com for "QuaLiKiz homepage."
[6]G. M. Staebler, J. E. Kinsey, and R. E. Waltz, "Gyro-Landau fluid equations for trapped and passing particles," Phys. Plasmas **12**, 102508 (2005).
[7]P. Strand, G. Bateman, A. Eriksson, W. A. Houlberg, A. H. Kritz, H. Nordman, and J. Weiland, "Comparisons of anomalous and neoclassical contributions to core particle transport in tokamak discharges," in Proceedings of the 31st EPS Conference on Plasma Physics (European Physical Society, 2004), Paper No. 28G, P-5.187.
[8]A. Casati, C. Bourdelle, X. Garbet, F. Imbeaux, J. Candy, F. Clairet, G. Dif-Pradalier, G. Falchetto, T. Gerbaud, V. Grandgirard *et al.*, "Validating a quasi-linear transport model versus nonlinear simulations," Nucl. Fusion **49**, 085012 (2009).
[9]G. M. Staebler, E. A. Belli, J. Candy, J. E. Kinsey, H. Dudding, and B. Patel, "Verification of a quasi-linear model for gyrokinetic turbulent transport," Nucl. Fusion **61**, 116007 (2021).
[10]E. Fransson, H. Nordman, P. Strand, and JET Contributors, "Upgrade and benchmark of quasi-linear transport model EDWM," Phys. Plasmas **29**, 112305 (2022).
[11]G. Cennachi and A. Taroni, JETTO: A free-boundary plasma transport code, JET-IR(88)03 (IAEA, 1988).
[12]D. Coster, V. Basiuk, G. Pereverzev, D. Kalupin, R. Zagórski, R. Stankiewicz, P. Huynh, F. Imbeaux, and Members of the Task Force on Integrated Tokamak Modelling, "The European transport solver," IEEE Trans. Plasma Sci. **38**(9), 2085–2092 (2010).
[13]D. Kalupin, I. Ivanova-Stanik, I. Voitsekhovitch, J. Ferreira, D. Coster, L. L. Alves, T. Aniel, J. F. Artaud, V. Basiuk, J. P. Bizarro *et al.*, "Numerical analysis of JET discharges with the European transport simulator," Nucl. Fusion **53**, 123007 (2013).
[14]J. Citrin, S. Breton, F. Felici, F. Imbeaux, T. Aniel, J. F. Artaud, B. Baiocchi, C. Bourdelle, Y. Camenen, and J. Garcia, "Real-time capable first principle based modelling of tokamak turbulent transport," Nucl. Fusion **55**, 092001 (2015).
[15]K. L. van de Plassche, J. Citrin, C. Bourdelle, Y. Camenen, F. J. Casson, V. I. Dagnelie, F. Felici, A. Ho, S. Van Mulders, and JET Contributors, "Fast modeling of turbulent transport in fusion plasmas using neural networks," Phys. Plasmas **27**, 022310 (2020).
[16]A. Ho, J. Citrin, C. Bourdelle, Y. Camenen, F. J. Casson, K. L. van de Plassche, H. Weisen, and JET Contributors, "Neural network surrogate of QuaLiKiz using JET experimental data to populate training space," Phys. Plasmas **28**, 032305 (2021).
[17]C. Bourdelle, X. Garbet, G. T. Hoang, J. Ongena, and R. V. Budny, "Stability analysis of improved confinement discharges: Internal transport barriers in Tore Supra and radiative improved mode in TEXTOR," Nucl. Fusion **42**, 892 (2002).
[18]A. Ho (2021). "QuaLiKiz-v2.6.2 linear instability spectra based on JET experimental plasma profiles," Zenodo, V 1.0. Dataset https://doi.org/10.5281/zenodo.7418108
[19]M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, see https://www.tensorflow.org/

for "TensorFlow: A system for large-scale machine learning software" (2016).
[20]I. Goodfellow, Y. Bengio, and A. Courville, see http://deeplearningbook.org for "Baggin and other ensemble methods deep learning" (MIT Press, Cambridge, MA, 2016).
[21]J. Borsander, O. Lindberg, W. Rieck, and M. Åqvist, "Surrogatmodell av QuaLiKiz för turbulenta instabiliteter i en tokamak," B.Sc thesis (Chalmers University of Technology, 2023).
[22]H. Nordman and J. Weiland, "Transport due to toroidal η_i mode turbulence in tokamaks," Nucl. Fusion **29**, 251 (1989).