Neural network training data optimization for turbulence modeling

Kremers, B.J.J.

Award date:
2020

EINDHOVEN UNIVERSITY OF TECHNOLOGY

Department of Applied Physics:
Science and Technology of Nuclear Fusion

Neural Network Training Data Optimization for Turbulence Modeling

Master Thesis

Bart Kremers

Supervisors:
dr. J. Citrin
dr. J. van Dijk
A. Ho MSc.

version 1.0

Eindhoven, March 2020

Abstract

In order to achieve a sustainable nuclear fusion reactor, among other problems turbulence needs to be modeled and optimized. Enhanced crossfield transport is induced by turbulence which leads to degradation of confinement of a hot fusion plasma. Turbulence can be modelled with direct numerical simulation. However, due to the enormous span of time and spatial scales, involved in these calculations, are not fast enough for specific applications like modelling discharge timescales even with current supercomputers. Reduced turbulence models have been created, such as QuaLiKiz, a quasilinear gyrokinetics code which is applicable for tokamak simulations. QuaLiKiz is still too slow for extensive optimization and control applications. Further speed up is possible by training of neural networks on reduced model input-output mapping. A critical part of neural network training is the population of the training data set, i.e. the input-output mapping. Under-population of the data set leads to enormous errors in the trained surrogate models. Over-population leads to wasted computational resources in creating the input-output mapping from the turbulence models. General practice is to over-populate databases, which is still tractable when using models such as QuaLiKiz, but will be impossible with higher fidelity codes, such as GENE, due to limitations in computing power. Therefore this work focuses on the following research question: Can a robust methodology be developed for reducing training set size while maintaining sufficient data density for accurate surrogate model development? In this work accuracy of the surrogate neural network is measured by the root mean squared and mean absolute error.

Multiple input-output data sets are computed using QuaLiKiz. A tool is constructed which employs two-step clustering to firstly detect regions of high data density, using the DBSCAN algorithm, and secondly reduces regions of high data density by splitting them in to k clusters using the kmeans clustering algorithm. Adjustments have been made to the original DBSCAN algorithm such that each parameter can be weighted differently, in this way taking into account data distribution per parameter. With this tool, reduced data sets are constructed. On these reduced input-output mappings committee neural networks are trained and these are compared via RMSE and MAE to the original data set and a data set with its noise removed called filtered data set. The results show that up to a certain reduction threshold, accuracy of the neural network stay constant, and after this threshold accuracy starts to decrease. A 15D input-output database started out consisting out of roughly 12 million data points. From this database multiple reduced data sets are created and are used for neural network regression training. The reduction factor threshold found was 25. As higher fidelity input to output generation with quasilinear GENE cost approximately 5 CPU hours per simulation, population of such a database could be reduced by roughly 570×10^6 CPU hours to 30×10^6 CPU hours,which is extremely large but possible. Such a database is to be built over years of simulations. This work is focused on surrogate models for transport modeling, however the developed tool could be used in other fields with similar multi-scale

modeling. These type of simulations occur in field such as engineering, bio-medicine, material science and weather forecasting. There might also be applications in data visualization, where data is too densely populated to be visualized.

Preface

I could not have done this work without my friends and family, in special I would like to thank Rik Limpens for the endless discussions we had. And of course my dear parents for their unending support.
I would like to thank all my colleagues but in special Aaron Ho who was my daily supervisor, who was always available for answering questions. I would like to thank Jan van Dijk who during our meetings always was extremely interested and who pushed me to think outside the box. And last but not least Jonathan Citrin who was always able to pick up what the next step should be and what the implications of certain results were.
I would like to end this preface with a quote that in any sort of fitting work should be remembered. As there is a big difference between understanding and reproducing.
"With four parameters I can fit an elephant, with five I can make him wiggle his trunk." - John Von Neumann

Contents

Contents vii

List of Figures ix

List of Tables xiii

1 Introduction 1

2 Theory 5
2.1 Physics . . . 5
2.1.1 Nuclear Fusion . . . 5
2.1.2 Confinement . . . 6
2.1.3 Transport . . . 7
2.2 Artificial intelligence . . . 11
2.2.1 Neural networks . . . 11
2.2.2 Clustering . . . 16

3 Method 18
3.1 Overall approach to the research . . . 18
3.2 Explanation of setup and data collection . . . 19
3.3 The clustering procedure . . . 20
3.4 Methods of analysis . . . 23

4 Analysis 25
4.1 3D case . . . 25
4.1.1 Reduction procedure . . . 25
4.1.2 Training . . . 26
4.2 4D case . . . 31
4.3 JET case . . . 33

5 Conclusions and outlook 36

Bibliography 40

Appendix 43

A Settings QuaLiKiz 3D data set 44

List of Figures

1.1 World energy consumption million tonnes oil equivalent. An increase in global energy consumption can be seen. In the year 2018 roughly 85% of the world energy consumption is created using non renewable and non nuclear sources. Taken from [1]. 2

1.2 The magnets create an helical field which confine the plasma particles. The toroidal field coils create a toroidal magnetic field, by running a plasma electric current using an inner poloidal field coil, a poloidal magnetic field is produced. The other magnets are used to shape and position the plasma. Taken from [2]. 3

2.1 Reaction rate versus Temperature showing that the D-T reaction occurs at the lowest temperature with the highest reaction rate, making it the easiest to achieve . Taken from [2]. 6

2.2 Stability plot of both ion and electron modes for a single wave number as function of the logarithmic density and temperature gradients R/L_n R/L_T. Adapted from [3] 8

2.3 A artificial neuron takes multiple inputs and computes an output by multiplying the inputs with weights sums the results adds a bias and puts it through an activation function. Taken from [4]. 12

2.4 A schematic representation of a two-hidden-layer feedforward neural network. $w^3_{1,3}$ is the weight between the second and third layer between the first node of the third layer and the third node of the second layer. b^2_3 the bias of the second hidden layer and the third node. Taken from [4]. 12

2.5 Sigmoid activation function $f(z) = \frac{1}{1+e^{-}z}$ often used as it ensures that small changes in the weights and biases, lead to small changes in the output. Taken from [5]. 13

2.6 Settings are $n_{\min} = 4$ and $\epsilon = 1$. Point A and all other red points are core points, since the area surrounding these points in an $\epsilon = 1$ radius contain at least $n_{\min} = 4$ points (including the point itself) they form a single cluster. Points B and C are density reachable from A (via other core points) and thus belong to the cluster. Point N is neither a core point nor a density reachable point and is thus labeled as noise. Taken from [6]. 16

2.7 2D representation of the kmeans algorithm with $k = 3$. (a) The random initialization step, (b) assigning points to cluster, the partitions represent the Voronoi diagrams generated by the means, (c) calculation of new mean of the cluster, (d) repetition until final clusters are formed. Taken from [7]. 17

3.1 In (a) the DBSCAN algorithm is shown where all dimensions have the same radius of neighborhood. The yellow point is considered part of the cluster however the green point is not. In (b) the DBSCAN is adjusted in order to set different radii for various dimensions. By setting the radius of the x component to 1.8 and keeping the y component the same the green point now also becomes part of the cluster. 21
3.2 2D representation of the reduction procedure, in (a) the original data is depicted, in (b) the data is clustered by the DBSCAN algorithm and noise is detected, in (c) the noise has been removed and kmeans clusters are determined, in (d) the reduced data set. 22
3.3 For the 3D artificial data set the number of clusters per number of points in the cluster, extremely small clusters < 6 occur relatively often. 23

4.1 Original data set containing 32761 data points created using QuaLiKiz where R/L_{Te} and R/L_{Ti} are input parameters and q_e in Gyrobohm units is the output calculated. 26
4.2 Original data set split into DBSCAN clusters, each color corresponds to one DBSCAN cluster with q_e in Gyrobohm units. 27
4.3 All points that do not fall into a DBSCAN cluster are labeled noise points are depicted in black. These noise points are removed from the original data set. q_e in Gyrobohm units. 28
4.4 One DBSCAN cluster is clustered into 194 kmeans clusters, each kmeans cluster is depicted in a specific color. Every kmeans cluster is averaged over and replaced by one point. q_e in Gyrobohm units. 28
4.5 Reduced data set, the final data that is left after the clustering routines, in this specific case the reduction factor was set to 10, and the reduced data set consists of 3209 points. q_e in Gyrobohm units. 29
4.6 In (a) and (b) depicted the RMSE for the neural networks trained on the original and filtered data sets respectively, in (c) and (d) the MAE is shown for the neural networks trained on the original and filtered data sets respectively. All show that accuracy of the neural network stays rather constant up to a certain threshold which in this case is a reduction factor of roughly 10, after which the errors start to increase due to the fact that the training data becomes too sparse. 30
4.7 In (a) and (b) depicted the RMSE for the neural networks trained on the original and filtered data sets respectively, in (c) and (d) the MAE is shown for the neural networks trained on the original and filtered data sets respectively. All show that accuracy of the neural network stays rather constant, up to a certain threshold, which in this case is a reduction factor of roughly 10. After the threshold the errors start to increase due to the fact that the training data becomes too sparse. 32
4.8 Points per dimension for different reduction factors. Due to the distinction between extremely small clusters and other clusters there is a limit to which the data set can be reduced. For the JET rotational data base this limit is encountered around a reduction factor of 100, after which almost all points are part of an extremely small cluster and are thus not reduced. 34

4.9 In (a) and (b) depicted the RMSE for the neural networks trained on the original and filtered data sets respectively, in (c) and (d) the MAE is shown for the neural networks trained on the original and filtered data sets respectively. All show that accuracy of the neural network stays rather constant up to a certain threshold which in this case is a reduction factor of roughly 25 after which the errors start to increase due to the fact that the training data becomes too sparse. 35

5.1 Overpopulated data set, data in black, three regions of high data density are 'hidden' under huge amounts of data. 38
5.2 After applying the tool to the overpopulated data set, visualization of the three regions of high data density. 39

List of Tables

4.1 Parameters in the JET rotation database ϵ values used for the DBSCAN algorithm. 33

Chapter 1

Introduction

This work is written as a master's thesis for the dual masters degree of Applied Physics (Plasma Physics and Radiation Technology) and Science and Technology of Nuclear Fusion at the Technical University of Eindhoven. Therefore the concepts explained here incorporate generalizations and simplifications, in order to keep the text readable for readers of both faculties.

Ever since the industrial revolution in the beginning of the 19^{th} century the global energy demand has increased, as more technology became available to individuals and industry. This increase over time applies not only to the amount of available technology, but also the amount of people who could access these technologies. The U.S. Energy Information Administration (EIA) projects nearly 50% increase in world energy usage by 2050, led by growth in Asia [1, 8, 9]. Roughly 85% of the worldwide energy supply is created using non renewable or non nuclear energy sources as can be seen in Figure 1.1 [1]. Non-renewable sources lead to greenhouse emissions [10]. As emission of CO_2 and other greenhouse gasses are linked to global warming there is a need for clean and sustainable energy [11]. Therefore changing to a new energy mix is a necessity, with this the question arises, what should this mix look like? Solar power using photovoltaics and wind energy using wind turbines seem promising, however these are intermittent sources and work only when there is sun and wind respectively. There is thus a need for an energy source which can provide a steady base load while also having less greenhouse gas emissions. Most energy scenarios assume fission power plants to fill this gap [1, 12], but this technology has a few drawbacks, namely:

1. The current generation of nuclear fission reactors produce long-lived nuclear waste.
2. The current negative public opinion on nuclear fission energy.
3. The possibility of a nuclear meltdown.
4. The possibility of escape of radiating materials.

During the 1950s people started working on designing other types of nuclear power plants, namely nuclear fusion reactors. A machine which produces no long-lived nuclear waste, on which the public opinion is currently positive, and in which the reaction is inherently safe. Nuclear fusion is the mechanism that powers the stars. Nuclear fusion happens when two light atoms are brought extremely close together, when these atoms are in close enough proximity the nuclear force which is an attractive force becomes larger than the repulsive electrostatic

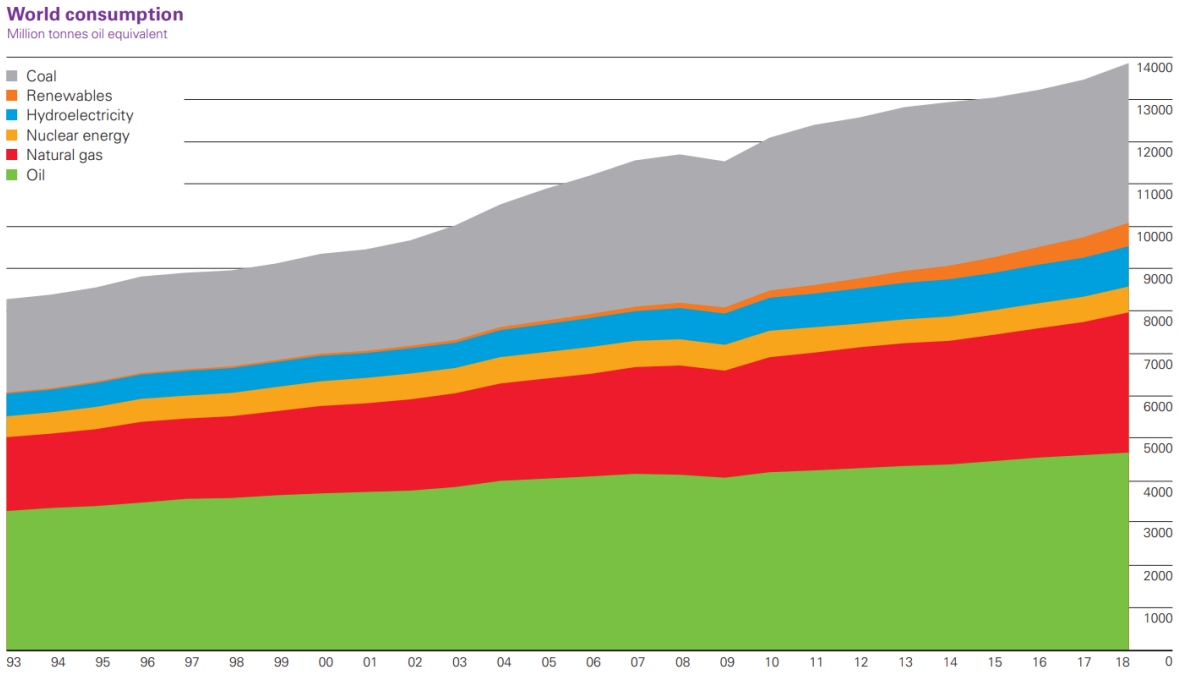


Figure 1.1: World energy consumption million tonnes oil equivalent. An increase in global energy consumption can be seen. In the year 2018 roughly 85% of the world energy consumption is created using non renewable and non nuclear sources. Taken from [1].

force. At this point, the atoms fuse together into other atoms and the difference in mass is converted into energy. The 'easiest', i.e. the highest chance of occurring at the lowest temperature, of this fusion reactions is the reaction between deuterium (D) and tritium (T), both isotopes of hydrogen (H). The reaction creates a helium (He) particle and a neutron (n) [2].

$${}^{2}_{1}\mathrm{D} + {}^{3}_{1}\mathrm{T} \longrightarrow {}^{4}_{2}\mathrm{He}\,(3.5\,\mathrm{MeV}) + \mathrm{n}^{0}\,(14.1\,\mathrm{MeV})$$

In order to make this reaction happen at a high enough rate, the particles must be able to meet each other with enough energy, this energy corresponds to a temperature of roughly 150 million degrees Celsius at which a hot gas is formed. At this temperature the deuterium and tritium are completely stripped from their electrons and the gas is called a plasma. Which is thus a gas consisting out of ionized particles that generally show collective behaviour. In order to get more energy out of this reaction than needs to go in, enough reactions have to happen, this is determined by the Lawson criterion $nT\tau_E >$ critical value, where n is the particle density, T the temperature and τ_E the energy confinement time. This Lawson criterion simply states that there must be enough particles for them to meet each other, that these particles must have enough energy in order to overcome the repulsive Coulomb force and that their energy must be contained for long enough to produce net energy at sufficiently low input power [13].

The fact that the plasma consists of charged particles plays into our favor here as this property can be used to confine the plasma. For this magnetic fields can be used, and the main machines which are now being designed are of the so called Tokamak type. A Tokamak is a machine in the shape of a doughnut, a torus as can be seen in Figure 1.2 [2]. The magnets create an helical field which confine the plasma particles. The toroidal field coils create a

toroidal magnetic field. Using an inner poloidal field coil, called central solenoid, a plasma current is created from which, a poloidal magnetic field is produced. The other magnets are used to shape and position the plasma.

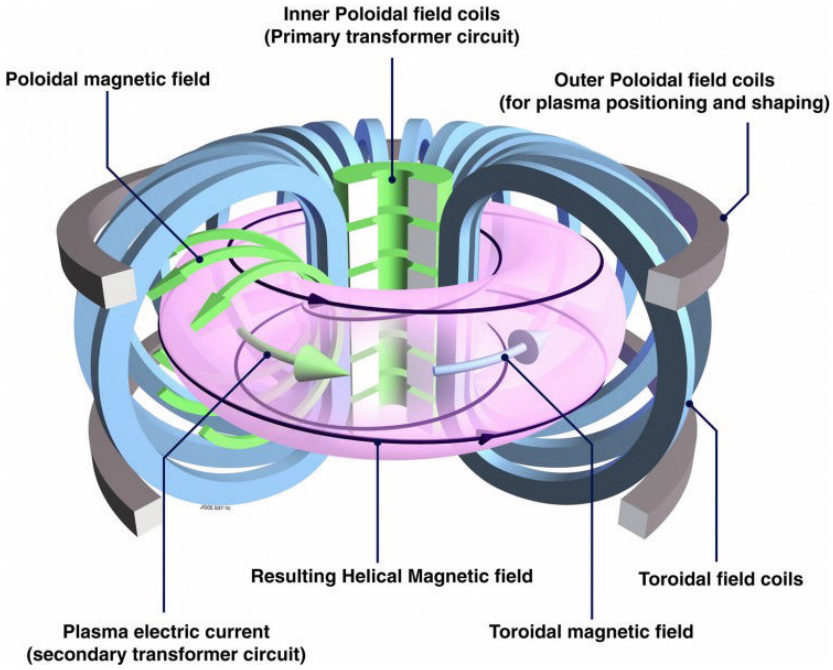


Figure 1.2: The magnets create an helical field which confine the plasma particles. The toroidal field coils create a toroidal magnetic field, by running a plasma electric current using an inner poloidal field coil, a poloidal magnetic field is produced. The other magnets are used to shape and position the plasma. Taken from [2].

In order to achieve the highest possible efficiency, confinement of particles, heat and momentum should be optimized. This confinement is determined by transport processes, thus far classical and neo-classical transport are theoretically understood well [2]. These transport mechanisms are of the collisional type and are insufficient to explain observed transport levels. Anomalous transport is believed to be due to turbulent transport.

Thus the next challenge is in understanding transport due to turbulent behaviour of the plasma. Turbulence in the plasma happens at relatively large length scales of the ion Lamor radius (millimeters to centimeters) and on time scales much slower than the gyro-motion. Experiment, theory and simulation have focused on the understanding of tokamak turbulence for decades, culminating in nowadays having the availability of experimentally validated direct-numerical-simulations based on the nonlinear gyrokinetic modelling. Computations are feasible with current computing power, but cost 10^4-10^5 CPU hours for a single turbulence flux calculation at a single radius. It is not feasible to use these simulations for discharge timescale temperature and density profile evolution [14].

Ideally for scenario optimization and control, these simulations should be on the order of tenths of seconds or faster. One way of increasing the speed is by training neural networks on a complex model such as QuaLiKiz or GENE. This trained neural network can then function as a surrogate model, which is simply a mapping from input to output avoiding complex calculations. Work by van der Plassche et al. has shown that this is feasible [4].

Generally speaking neural network training gets better when presented with more data to train on. However building this training data from QuaLiKiz or higher fidelity models is computationally expensive. Therefore it is of interest to know how much data is enough data, to capture the core features and keep an high enough accuracy for the surrogate model. Often the input for these models comes from experiments and is highly dimensional. The output of these codes are generally particle, heat or momentum fluxes.

This leads to the research question of this work which is, **Can a robust methodology be developed for reducing training set size while maintaining sufficient data density for accurate surrogate model development?**

For this the following sub questions are researched. How to detect high density regions in an automated way in order to deduce core feature regions. When these regions of high density are detected how can they be reduced. What metric is a good measure for accuracy in the trained neural networks. How do the trained neural networks on the reduced data sets compare to the original.

The second Chapter of this thesis introduces background theory to understand the work. Starting with the required physics for understanding nuclear fusion moving into turbulent transport and the derivation of a turbulence transport code. This explains the validity of the QuaLiKiz code which in principle is being used in order to train the surrogate neural networks. The next subsection in Chapter 2 is on artificial intelligence focusing on neural network regression, in this part the fundamentals of neural networks are discussed. Followed by a piece on data clustering techniques used for determining high data density regions and reducing data. Chapter 3 addresses the overall approach to the research, the setup of the work and the way data is collected. Then the tool that is constructed is introduced and discussed and the method of analysis is introduced. Chapter 4 is on the practical work, showing 3 different cases where a data set is generated and reduced using the developed tools and then the root mean squared error and mean absolute error calculated for trained neural networks. The results are discussed as well as the trends that appear from the data. Chapter 5 summarizes the whole work and makes concluding statements on the gathered information.

The developed tool gives us the possibility to reduce data sets, and to validate, whether it is possible to reduce a training data set, keeping the same level of accuracy. The results show that reduction is possible as long as reduction is not above a certain data dependent threshold. This enables us to construct a database more precisely with a higher fidelity code reducing computational times up to a factor 20. This work has been presented on a poster during the ACOS 2019 symposium, and will be published in a paper on neural network surrogate modeling by A. Ho et al. later this year, and the developed tool is to be published in a data science journal to be determined.

Chapter 2

Theory

In this Chapter the theoretical basis for this work is laid down. The first section introduces the physical concepts of nuclear fusion, focusing on magnetic confinement fusion. Transport of particles, energy and momentum in such a confined plasma is discussed. Starting with the theoretically well understood classical and neo-classical transport, next turbulent transport is introduced, reviewing distinct turbulent regimes. Following this, a piece on turbulence modeling where multiple transport modeling codes are discussed. Finally the QuaLiKiz transport model is introduced and its limitations are discussed.

The second section will introduce the fundamental concepts of machine learning which is a scalable approach to function fitting [15]. A tool which became extremely useful due to recent advances in computational power. After this general explanation the theoretical basis for a surrogate model will be discussed combining the from physics derived QuaLiKiz model with a trained neural network.

The final section is a more general section on essential data analysis techniques used in this work. Mainly focusing on the clustering algorithms kmeans and DBSCAN.

2.1 Physics

2.1.1 Nuclear Fusion

Nuclear fusion is the process where two light atoms merge into other elements. The sum of the newly formed elements masses is lower than the sum of the initial two atoms masses, the difference in the sum of masses is converted into kinetic energy via Einstein's famous non relativistic formula:

$$E_{kin} = \Delta m c^2$$

Where E_{kin} is the kinetic energy, Δm is the difference in the sum of the masses and c is the speed of light. A quantitative form to describe the probability of the fusion reaction to occur, is by looking at its cross section σ. The cross section depends on the energy of the initial particles. By multiplying this cross section with the velocity and averaged over a Maxwellian velocity distribution with temperature T the reaction rate is determined. In Figure 2.1 the reaction rate for the three most common fusion reactions are depicted. From Figure 2.1 it can be seen that the reaction rate for a D-T reaction is highest at the lowest temperatures of the three reactions and is thus considered the easiest obtainable fusion reaction. The D-T

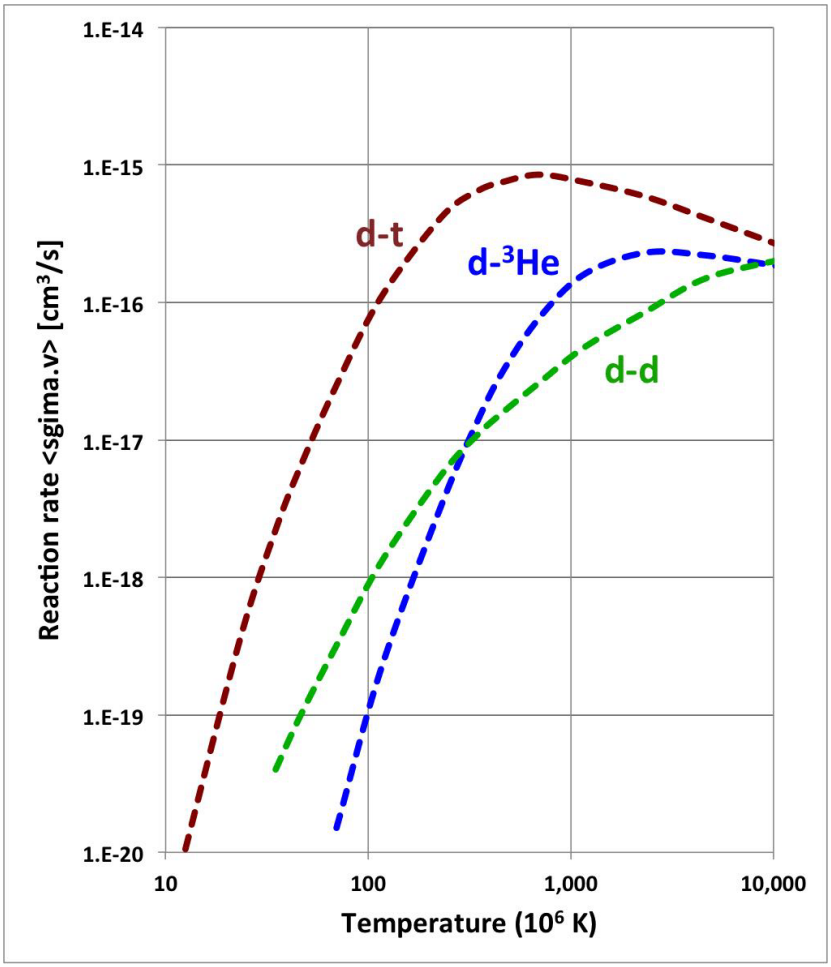


Figure 2.1: Reaction rate versus Temperature showing that the D-T reaction occurs at the lowest temperature with the highest reaction rate, making it the easiest to achieve . Taken from [2].

reaction is given by:

$$ {}^{2}_{1}D + {}^{3}_{1}T \longrightarrow {}^{4}_{2}He(3.5MeV) + n^{0}(14.1MeV) \tag{2.1} $$

As the power loss in a reactor depends on the temperature, the optimum temperature is lower than the temperature where the maximum reaction rate is located. The temperature at which a reactor would operate is roughly around 150 million degrees Celsius or roughly 15 keV. At these temperatures a D-T gas is completely ionized and is thus in the plasma state [2].

2.1.2 Confinement

As stated in the introduction the fact that the D-T gas is in the plasma state can be used to confine the gas, by using a magnetic field. The reason confinement is needed is, that in order to achieve net energy gain, the high temperatures needed for fusion reactions must be sustained with low enough external energy input. Also known as $Q > 1$, where $Q = \frac{P_f}{P_{ext}}$ with

P_f being the fusion power and P_{ext} being the external power. This magnetic field interacts with the charged particles via the Lorentz force $\mathbf{F}_{Lorentz} = q(\mathbf{E} + \mathbf{v} \times \mathbf{B})$. Where q is the charge of the particle, $\mathbf{E}$ is the electric field, $\mathbf{v}$ is the velocity and $\mathbf{B}$ is the magnetic field. Due to this force particles start to gyrate around the magnetic field lines, and cross field diffusion is significantly reduced. Parallel to the magnetic field lines the diffusion is unchanged. By using a toroidal geometry a plasma can be confined. A machine where such a field is created is called a tokamak.

2.1.3 Transport

Classical transport

In a homogeneous straight magnetic field charged particles follow field lines, without collision, cross field transport would be zero. In practice collisions do occur making particles hop from field line to field line. During a collision the average displacement is the gyro-radius ρ and the collision time τ describes the time between collisions. The classical heat diffusivity per species s is then $\chi_s \approx \rho_s^2/\tau$. Due to quasi-neutrality the particle transport coefficient D is bound to the slow species namely the ions $D \approx \chi_i$ [16, 2].

Neo Classical transport

In a tokamak the magnetic field is not homogeneous and straight but helical due to the toroidal geometry. This leads to increased cross-field diffusivity of one order of magnitude and a decreased parallel conductivity of roughly a factor two. This is due to the fact that at smaller radius the magnetic field increases compared to the magnetic field on the outer radius and due to this particles might be reflected by the magnetic mirror effect. Another reason for this is that the particles are drifting in the vertical direction due to the curvature and gradient of the field [2].

Turbulent transport and regimes

From experiments the neo-classical predicted parallel transport seems to be well described, cross-field transport however is significantly higher. This anomalous transport is induced by turbulence. As this enhanced cross-field transport leads to degradation of confinement, it is important to identify and understand the mechanisms that determine the level of anomalous transport. Once identified, this can be used for predicting and even reduction of these non-linear effects [17].
Turbulent transport is driven by micro-instabilities, instabilities that occur on spatial scales comparable to the ion gyro-radius due to pressure and temperature gradients. Above certain gradient critical thresholds the growth rates of these instabilities increase sharply and small fluctuations in the electromagnetic fields lead to strong particle, moment and heat fluxes. The ratio between plasma pressure and magnetic pressure is described by $\beta = nk_bT/(B^2/2\mu_0)$ where k_b is the Boltzmann constant. At low β, the most common instabilities are of the electrostatic type and drive $E \times B$ drift velocity fluctuations. At high β, modes with strong magnetic components arise which can dominate transport. Two examples are the micro tearing modes (MTM) and the kinetic ballooning modes (KBM) [18].
Various types of drift waves are the main type of electrostatic linear instabilities, mostly driven by temperature gradients and only little by density gradients. There are three groups

of drift wave instabilities namely: slab modes, toroidal modes and trapped modes [18]. Slab modes arise due to the parallel motion. Toroidal modes arise due to magnetic curvature drifts of passing particles coupling to electrostatic waves. Ion and electron temperature gradient modes (ITG, ETG) are thus passing particle instabilities and can be both of the slab or toroidal type [19]. Slab, toroidal and trapped modes are idealized and contribute to mode destabilization. Generally due to strong anisotropy of ITG and ETG the resulting transport is two dimensional. The main driver of ion heat transport due to turbulence are believed to be ITG modes. ETG on the other hand, drives only electron heat transport. Due to interaction between the toroidal procession of trapped particles and electronic fluctuations, trapped modes such as collisionless trapped ion modes (TIM) and trapped electron modes (TEM) develop. ETG, ITG and TEM modes are resolved by the QuaLiKiz code. TIMs do not contribute to transport since they get suppressed by nonlinear effects linked to the electron collision frequency [19]. TEMs are a cause for both electron heat and particle transport and are driven by low energy trapped electrons and electron temperature gradients. Figure 2.2 shows a stability plot for both ion and electron modes at a specific wave number, for low gradients the region is stable, at higher gradients multiple modes can be unstable. The saturated nonlinear effect of drift wave modes is determined by nonlinear interactions through zonal flows, which are linearly stable band modes of poloidal and toroidal rotation. Zonal flows contribute to shearing the turbulence radial structures, and also saturate turbulence through coupling unstable drift waves to damped modes.

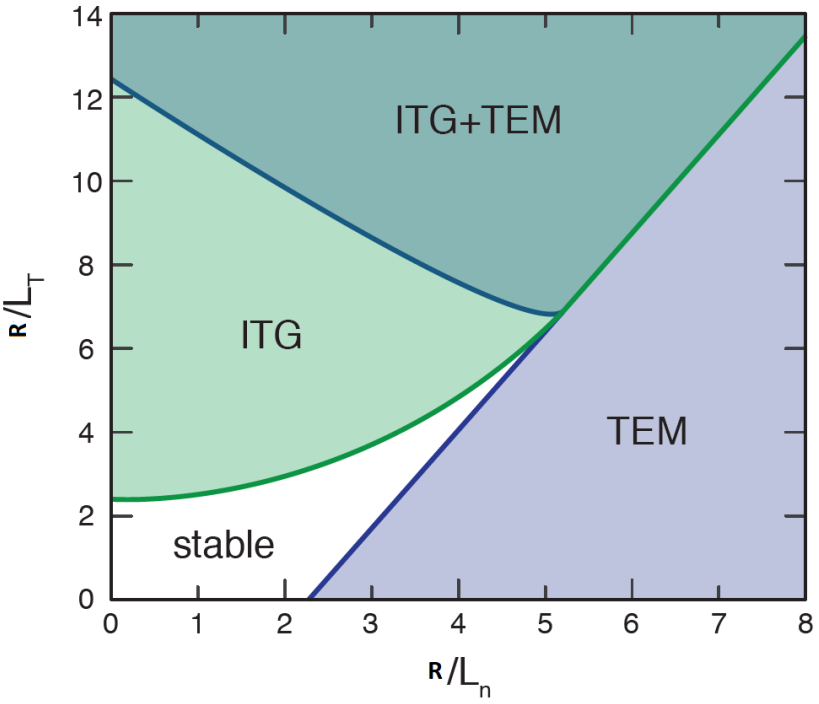


Figure 2.2: Stability plot of both ion and electron modes for a single wave number as function of the logarithmic density and temperature gradients R/L_n R/L_T. Adapted from [3]

Turbulent transport modeling

Confinement is determined by transport of heat, particles and momentum. The first principle description to describe the collective kinetic behaviour of the ensemble of charged particles in such a device boils down to the Vlasov-Maxwell-Fokker-Planck (VMFP) system of equations [19]:

$$\frac{\partial f_s}{\partial t} + \mathbf{v} \cdot \frac{\partial f_s}{\partial \mathbf{r}} + \frac{Z_s e}{m_s}(\mathbf{E} + \mathbf{v} \times \mathbf{B}) \cdot \frac{\partial f_s}{\partial \mathbf{v}} = C_s(f_s) - S_s \tag{2.2}$$

$$\nabla \cdot \mathbf{E} = \frac{1}{\epsilon} \sum_s Z_s e \int f_s d\mathbf{v} \tag{2.3}$$

$$\nabla \cdot \mathbf{B} = 0 \tag{2.4}$$

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} \tag{2.5}$$

$$\nabla \times \mathbf{B} = \mu \sum_s Z_s e \int \mathbf{v} f_s d\mathbf{v} + \frac{1}{c^2} \frac{\partial \mathbf{E}}{\partial t} \tag{2.6}$$

Where the subscript s is for the species, $f_s(\mathbf{r}, \mathbf{v}, t)$ is the six dimensional time dependent velocity distribution function for particle ensemble, $\mathbf{r}$ is the spatial vector, $\mathbf{v}$ is the velocity vector, Z_s is the charge, m_s is the mass, C_s is the collision operator, S_s is the source/sink term, ϵ is the plasma permittivity and μ is the plasma permeability. In fusion plasmas often the Fokker-Planck collision operator is used as collisional processes are dominated by small-angle scattering [19]. By taking moments of the distribution function, macroscopic quantities such as density and temperature can be determined. Theoretically it is possible to solve the VMFP system of equations numerically however due to the enormous span of time and spatial scales involved in these calculations this is not tractable, even with current supercomputers. Therefore there is a need for simplifications and approximations such that the system of equations can be solved and used for transport modeling.

Due to the strong magnetic field in a tokamak, fluctuations of the distribution in phase space are assumed to occur on much larger spatial scales than the gyro-radius ρ_s and on a much longer timescale than the gyro-frequency ω_s [20]. By averaging the kinetic equation over the gyro-motion generated via the Lorentz force, transport on spatial and time scales can be studied on the order of the plasma instabilities. This averaging has the benefit of reducing the six dimensional phase space to five dimensions. By expansion of the VMFP equation and the assumptions discussed and performing the gyro-averaging this gives us the drift-kinetic equation (DKE) [21]. A more general form of the DKE can be derived recursively starting from the kinetic equation [22]. These DKE models are frequently used in neoclassical transport models, neoclassical transport is discussed later in more detail.

By including small fluctuations in the amplitude of the electromagnetic fields on the gyro-radius scale during the averaging procedure we obtain the gyro-kinetic (GKE) equations [19]. This non-linear set of equations is the most complete theory that is solve-able with modern supercomputers. From this set of equations turbulent models are derived where electromagnetic fluctuations generated by kinetic instabilities alter the distribution function.

Using perturbation theory DKE and GKE can be solved, this is done by expansion of the velocity distribution function as $f_s = f_{0s} + \varepsilon_c f_{1s} + \ldots$, where ε_c is a small number representing the order parameter, and truncating terms beyond a specified power of the order parameter. Evaluating the lowest order equation, it can be seen that the lowest order is a Maxwellian

distribution function $f_{0s} = f_M$. Higher orders are corrections to the Maxwellian distribution function and thus $f_s = f_M + \sum_n f_{ns}$ where we can write $\sum_n f_{ns} = \delta f$. This allows us to solve δf for the desired accuracy. For more information on this [23, 24, 25] are recommended however are considered to be outside the scope of this work. Due to the assumption that global transport processes happen at large spatial scales compared to ρ_s in the δf approximation, radial variation of macroscopic quantities such as temperature and density and their gradient scale-lengths may be neglected locally. Thus grants the possibility for local simulations covering a narrow sub-domain of a few tens of gyro radii wide, perpendicular to the B field that is extended along the field lines, which is called a flux tube [19]. Even with the δf approximation global simulations are extremely computationally expensive, and thus mostly local formulations are used. Global simulation is sometimes used for benchmarking for reduced physics models in regions where strong radial variation in the macroscopic quantities exists as the local assumption does not hold in these regions [26].
The next approximation to be made, is the quasi-linear approximation. For this the linear dispersion relation is solved directly from the linearized GKE. By closing the linear response with the Maxwell equations this dispersion relation is obtained. This can be done with the electrostatic approximation, as in QuaLiKiz, this reduces down to the quasi-neutrality constraint $n_e = \sum_s n_{i,s}$. A spectrum of linear transport fluxes is calculated from the instabilities. This is then combined with a nonlinear saturation rule in order to estimate the saturated nonlinear amplitude of the fluctuations. This nonlinear saturation rule generally comes from a combination of analytical models with direct tuning to nonlinear simulations [27]. Transport due to microturbulence can be approximated from the quasi-linear approximation over a wide range of core plasma parameters [27].
The final goal for these models is calculation of the radial evolution of macroscopic quantities, thus the moments of the distribution function. As stated earlier global gyrokinetics simulations are not a practical tool to study the radial transport due to computational costs, often a fluid model consisting of the kinetic and velocity moments is used. The general equations for density, parallel momentum and energy are obtained by taking the zeroth, first and second order velocity moments and flux surface averages of the kinetic equation are [28, 29]:

$$\frac{\partial \langle n_s \rangle}{\partial t} + \frac{1}{V'}\frac{\partial}{\partial r}(V'\Gamma_s) = \langle \int S_{n,s} d^3v \rangle \tag{2.7}$$

$$\frac{\partial}{\partial t}\langle \omega_o R^2 \sum_s m_s n_s \rangle + \frac{1}{V'}\frac{\partial}{\partial r}(V' \sum_s \Pi_s) = \sum_s \langle \int m_s v (C_s + S_{\omega,s}) d^3v \rangle \tag{2.8}$$

$$\frac{\partial \langle W_s \rangle}{\partial t} + \frac{1}{V'}\frac{\partial}{\partial r}(V' Q_s) + \Pi_s \frac{\partial \omega}{\partial \psi} = \langle \int \frac{m_s v^2}{2} (C_s + S_{W,s}) d^3v \rangle \tag{2.9}$$

Where again the subscript s is for species s, n_s is the particle density , V' is the derivative with respect to r of the flux surface volume where r is the radial coordinate of the flux surface at the mid plane, Γ_s is the flux surface averaged radial particle flux, $S_{n,s}$ is the particle source rate, ω_0 the angular moment R the major radius, Π_s the flux surface averaged radial momentum flux, $S_{\omega,s}$ the momentum source rate, $W_s$the energy density, Q_s the flux surface averaged radial heat flux, ψ the poloidal flux function and $S_{W,s}$ the heat source rate. The flux surface averaged radial particle momentum and heat fluxes are given by [29]:

$$\Gamma_s \equiv \langle \int f_s \mathbf{v} \cdot \nabla \psi d^3v \rangle \tag{2.10}$$

$$\Pi_s \equiv \langle \int f_s m_s v_\zeta \mathbf{v} \cdot \nabla \psi d^3 v \rangle \tag{2.11}$$

$$Q_s \equiv T_s \langle \int f_s (\frac{\epsilon}{T_s} - \frac{5}{2}) \mathbf{v} \cdot \nabla \psi d^3 v \rangle \tag{2.12}$$

This is not yet a closed set, because each moment as they are called is a function of the next moment, the general closing strategies are asymptotic closure and truncation [30].

2.2 Artificial intelligence

Artificial intelligence (AI) is the field in computer science in which a computer develops the capability of carrying out complex tasks that usually require human intelligence e.g. image recognition, speech recognition and decision-making. One important segment of AI is machine learning (ML), this is the part that uses statistics to create statistical models to predict and identify patterns in data [31]. ML itself can be divided in subsection such inductive logic programming, reinforcement learning, Bayesian networks, deep learning and clustering. In this work we will focus on deep learning and then specifically neural networks and clustering techniques.

2.2.1 Neural networks

Neural networks are computing systems that are inspired by biological neural circuits. These systems are not told how to solve problems, instead they learn from observed data. After learning, the result is a trained neural network which is an input to output mapping for the given problem, not based on physics but on example.
The first mention of such a model for computational neural networks was in 1943 by McCulloch and Pitts [32]. Inspired by this work, Rosenblatt created the perceptron [33]. Which was later used to derive the sigmoid neuron which will be explained shortly. Until 2006 neural networks were not able to surpass more traditional approaches, except for a few specific problems. A breakthrough by Geoffrey Hinton et al. came by the discovery of back propagation techniques for learning in so-called deep neural networks [34]. Back propagation is explained shortly. These techniques have now been further developed, also due to the current available computing power, and are now able to attain exceptional performance on many problems such as computer vision, speech recognition, and natural language processing. Companies such as Google, Facebook and Microsoft are now applying these techniques at large scale.
The key element to the artificial neural network is the artificial neuron as depicted in Figure 2.3, such a node takes multiple inputs and has an activation function $f()$ leading to an output y, the activation function is discussed later in this chapter in more detail.

This can be mathematically formulated as $y = f(\sum_j^n w_j x_j + b)$ where w is the weight of the node, n is the total number if inputs and b is the bias, bias is a measure of how easy it is to get the neuron to fire [4]. Weights and biases are what are optimized to match the input-output relation of a given training set, when they are optimized the network is considered 'trained'. Combining multiple artificial neurons as shown in Figure 2.4 we obtain a deep neural network, a neural network is considered deep when it has two hidden layers. This (relatively shallow) deep neural network takes three inputs in the input layer, than has two hidden layers and one output layer [4], the notation is as followed, the superscript means which layer it is and the subscript denotes the node starting at 1 at the top. The bias of the second hidden layer

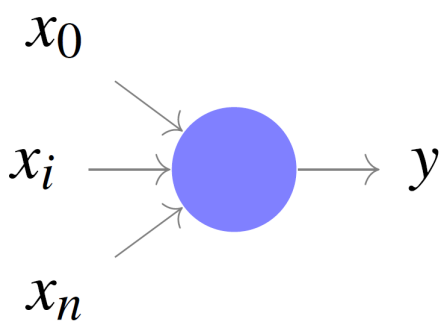


Figure 2.3: A artificial neuron takes multiple inputs and computes an output by multiplying the inputs with weights sums the results adds a bias and puts it through an activation function. Taken from [4].

and the third node is the b_3^2. Weights are labeled in superscript the layer it is in front of and in subscript the nodes it connects first mentioning the node on the right and than the node on the left, as depicted in Figure 2.4 $w_{1,3}^3$ is the weight between the second and third layer between the first node of the third layer and the third node of the second layer. To put it more general w_{jk}^l is the weight of the connection from the k$^{\text{th}}$ neuron in the (l-1)$^{\text{th}}$ layer to the j$^{\text{th}}$ neuron in the l$^{\text{th}}$ layer. Using similar notation for the biases b_j^l and activation a_j^l for the j$^{\text{th}}$ neuron of the l$^{\text{th}}$ layer. A vast array of neural network configurations are possible: from very deep meaning lots of layers to very wide meaning many nodes per layer, the general trend is to go wider instead of deeper [15].

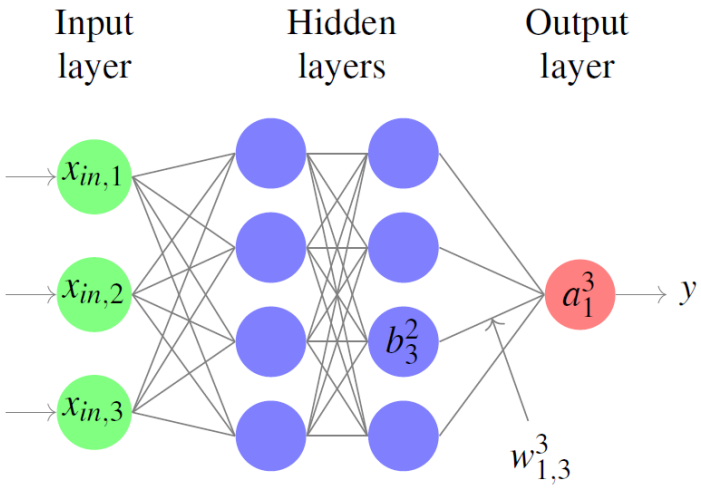


Figure 2.4: A schematic representation of a two-hidden-layer feedforward neural network. $w_{1,3}^3$ is the weight between the second and third layer between the first node of the third layer and the third node of the second layer. b_3^2 the bias of the second hidden layer and the third node. Taken from [4].

Let's take as an example a simple classification problem of determining whether a pho-

tograph includes a dog or not. A large number of photographs has to be labeled by hand with yes or no. These photographs are then given to the neuron as input. Depending on the neuron it might be associated with determining whether there is a tail or not, whether there is fur or not etc. Based on this information and the weights and biases the neuron spits out a number between 0 and 1. 0 meaning there is no tail and 1 there is a tail for example. Ideally a small change in weights and biases should be associated with a small change in the output. Something which almost certainly a dog should still be classified as a dog when only small changes are made. This is not the case for a step activation function for instance. There are many activation functions possible the most commonly used is the sigmoid activation function $f(z) = \frac{1}{1+e^{-z}}$, depicted in Figure 2.5 [5]. These weights and biases are not a given, and are what the neural network has to learn by example, using the labels which have been provided to the training set by hand. The earlier constructed training data set information is handed to the neural network and the result is compared with the original label, then an update is done to the weights and biases in order to get better predictions. The most common way of training a neural network of this sort is by the back propagation method, which will be explained later in this section.

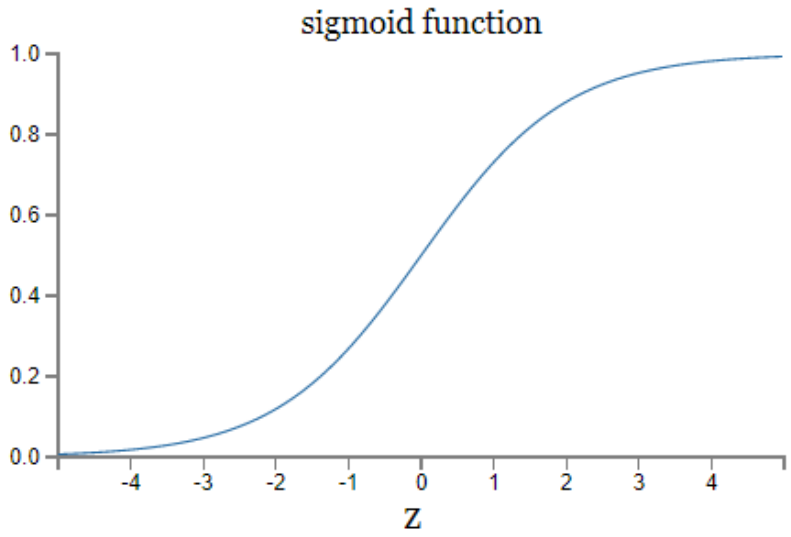


Figure 2.5: Sigmoid activation function $f(z) = \frac{1}{1+e^{-z}}$ often used as it ensures that small changes in the weights and biases, lead to small changes in the output. Taken from [5].

Due to the smoothness of the sigmoid activation function a small change in the weights Δw_{jk} and Δb_j will result in a small change in the output $\Delta output$. We can approximate this as [5]:

$$\Delta \text{output} \approx \sum_j \frac{\partial \text{output}}{\partial w_{jk}} \Delta w_{jk} + \frac{\partial \text{output}}{\partial b_j} \Delta b_j \tag{2.13}$$

Thus Δouput is a linear function of the changes in Δw_{jk} and Δb_j in the weights and bias where for ease we drop the superscript labels. In order to track how well a neural network is performing a cost function (also known as loss or objective function) is defined. A very

general one is the following:

$$C(w,b) \equiv \frac{1}{2n} \sum_x \parallel y(x) - a \parallel^2 \tag{2.14}$$

Where a is the vector of outputs from the network when x is the input and the sum is over all training inputs. This general cost function is called the quadratic cost function and is simply the mean squared error. Minimizing the cost function leads to a better neural network as this means that $y(x) \approx a$ which is the thing that is trying to be achieved. This minimizing can be done by an algorithm known as gradient descent. Assume we try to minimize $C(v)$ where $v = v_1, v_2, ...$ as this might be millions of weights and biases simply using calculus to take the derivative will be extremely tedious. Imagine a ball rolling down the slope of a valley. By randomly choosing the starting point and letting the ball roll down to the bottom we might be able to find the minimum. If the ball moves a small amount Δv_1 in the direction of v_1 and small Δv_2 in the v_2 direction, C will change as:

$$C(w,b) \approx \frac{\partial C}{\partial v_1} \Delta v_1 + \frac{\partial C}{\partial v_2} \Delta v_2 \tag{2.15}$$

We want the ball to roll down which means that Δv_1 and Δv_2 should be chosen such that ΔC is negative. ΔC can be written as $\Delta C \approx \nabla C \cdot \Delta v$. Suppose we chose $\Delta v = -\eta \nabla C$ where η is a small positive parameter which we call the learning rate. We update $v \rightarrow v' = v - \eta \nabla C$ over and over again to 'hopefully' reach a global minimum. Thus computing gradient C and move in the opposite direction [5].
Applying this idea comes with a few challenges. In order to compute C all gradients for each training input, x, should be calculated and averaged over. Here stochastic gradient descent helps us, the idea is to estimate ∇C for a small sample of randomly chosen training inputs. Randomly picking a small number m of training inputs, X_1, X_2 ,$\cdots$, X_m and calling them mini batches $\nabla C \approx \frac{1}{m} \sum_{j=1}^{m} \nabla_{X_j}$.
A mini-batch is picked randomly the training algorithm is used, then the next mini-batch is picked and trained on until all training inputs have been used, this is called an epoch. This is done multiple times. A fast algorithm for computing gradient needed for learning the weights and biases using gradient descent is the back propagation algorithm [35]. The activation a_j^l for the j^{th} neuron of the l^{th} layer is given by:

$$a_j^l = \sigma(\sum_k w_{jk}^l a_k^{l-1} + b_j^l) \tag{2.16}$$

The term $z^l \equiv \sigma(\sum_k w_{jk}^l a_k^{l-1} + b_j^l)$ is defined as the weighted input to neurons in layer l, and will be used shortly. The goals of the backpropagation algorithm is to compute the partial derivatives $\partial C/\partial w$ and $\partial C/\partial b$, therefore two assumptions about the cost function have to be made. The first is that the cost function can be written as an average over cost functions C_x for individual training examples x thus $C = \frac{1}{n} \sum_x C_x$. The second assumption is that it can be written as a function of the outputs of the neural network. Introducing an intermediate quantity $\delta_j^l \equiv \frac{\partial C}{\partial z_j^l}$ which we call the error in the j^{th} neuron in the l^{th} layer. There are four fundamental equations which are used in back propagation. They allow for the computation of both the error δ^l and the gradient of the cost function. The error in the output layer:

$$\delta^L = \nabla_a C \odot \sigma'^{(z^L)} \tag{2.17}$$

Where $\odot$ is the Hadamard product which is nothing else than the element-wise product of two vectors. The error l in terms of the error in the next layer, l+1:

$$\delta^l = ((w^{l+1})^T \delta^{l+1} \odot \sigma'^{(z^l)} \tag{2.18}$$

The rate of change of the cost with respect to any bias in the network:

$$\frac{\partial C}{\partial b^l_j} = \delta^l_j \tag{2.19}$$

The rate of change of the cost with respect to any weight in the network:

$$\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j \tag{2.20}$$

Using these four equations we can apply the backpropagation algorithm which goes as follows,

1. the input is x, set the corresponding activation a^1 for the input layer.
2. feedforward for each $l = 2, 3, .., L$ compute the weighted input $z^l = w^l a^{l-1} + b^l$ and $a^l = \sigma(z^l)$
3. output error compute the vector $\delta^L = \nabla_a C \odot \sigma(z^L)$
4. back propagate the error for $l = L-1, L-2, .., 2$ compute $\delta^l = ((w^{l+1})^T \delta^{l+1} \odot \sigma'^{(z^l)}$ and the last step is the output where the gradient and cost function are given by $\frac{\partial C}{\partial b^l_j} = \delta^l_j$ $\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j$

After all this the neural network is trained and can be presented with photographs that it has never seen, and will be able, to up to a certain accuracy, tell whether this new photograph contains a dog or not. In practice, applications like image recognition use alternative and more efficient topologies like convolutional neural nets (CNN) [36].
A trained neural network is trained on a specific data set is considered good if it generalizes well to new data. If the training of a neural network is too explicit it almost perfectly fits the training data but it performs poorly on new data, this is known as overfitting. A trained neural network over-fits when it learned the very specific pattern and noise from the training data, therefore it is not able to extract the big picture nor the general pattern the data. In order to overcome this early stopping is employed, meaning that if the change in cost function is not increasing enough for multiple epochs the training is stopped. Another way overfitting is prevented is by adding a regularization term to the cost function, the regularization term punishes large values of the weights and biases in the optimization. Meaning that this technique discourages learning a more flexible or complex model [37]. Beyond classification, neural networks can be used for regression, i.e. fitting the input-output mapping of smooth functions. This is what was done for the QuaLiKiz quasilinear gyrokinetic transport model [4] and these types of models are the main focus of this thesis.

2.2.2 Clustering

Clustering is a machine learning technique that involves the grouping of data points. These groups are sorted to have similar features/properties. The research question of this work focuses on the reduction of data, over represented data points are to be removed/replaced. Clustering enables the grouping of similar data points which than can be removed or replaced.

DBSCAN

The Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm was proposed in 1996 by Ester et al. [38]. DBSCAN aims to cluster data points in regions of high density, meaning points with many nearby neighbors, and labels all points not in high density regions as noise. The DBSCAN algorithm takes two input parameters, namely the radius of neighborhood ϵ and n_{min} the minimum amount of points that have to be in the radius of neighborhood for it to be considered a so called core point. The algorithm sorts the data into three different groups. The first group are the so called core points displayed in red in Figure 2.6, points which are at least n_{min} within a distance ϵ. the second are density reachable points, points within ϵ from a core point which are not themselves core points, displayed in yellow in Figure 2.6 All other points fall into the last category the noise points and are marked blue in Figure 2.6. The first step of the algorithm is randomly chose a point, check whether it is a core point, if so a cluster is started, if not it is assigned to noise. Then repeat the process. This means that in later steps this noise point might be changed into a density reachable point and thus becomes part of a cluster.

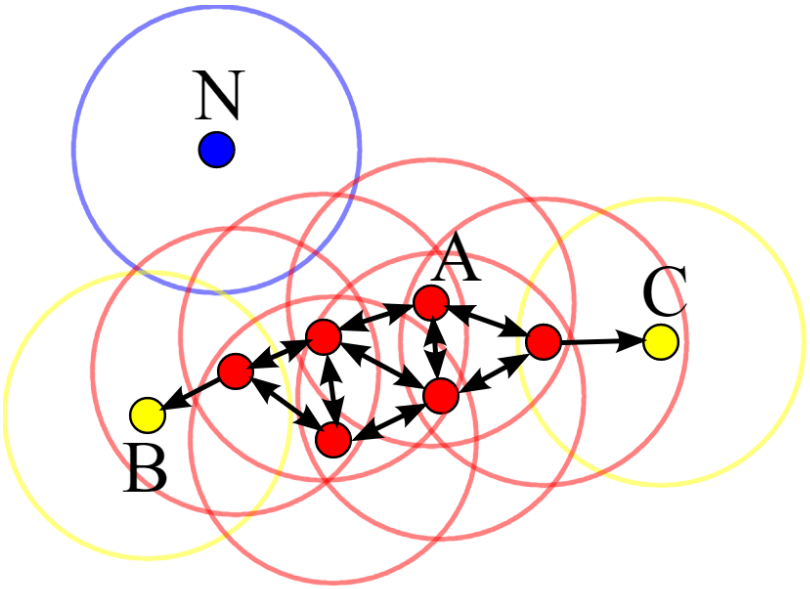
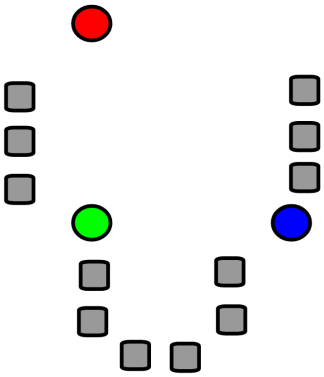


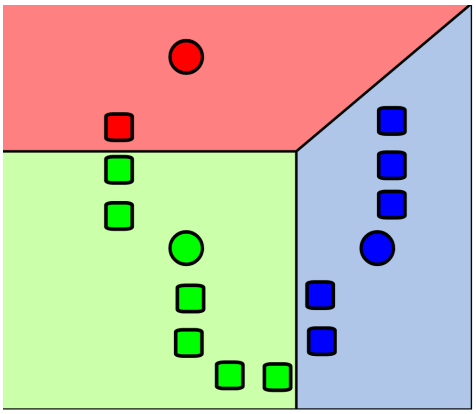
Figure 2.6: Settings are $n_{\text{min}} = 4$ and $\epsilon = 1$. Point A and all other red points are core points, since the area surrounding these points in an $\epsilon = 1$ radius contain at least $n_{\text{min}} = 4$ points (including the point itself) they form a single cluster. Points B and C are density reachable from A (via other core points) and thus belong to the cluster. Point N is neither a core point nor a density reachable point and is thus labeled as noise. Taken from [6].

k-means clustering

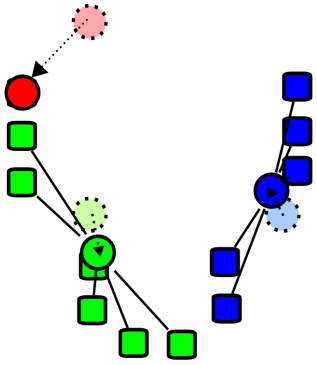
The k-means clustering algorithm is an algorithm which splits the data up into k clusters with the nearest mean. The user of the algorithm has to set the number of clusters k. With this information the algorithm randomly selects k distinct point in the data domain depicted in Figure 2.7a, than the distance is calculated to each point and each point is clustered with the nearest mean shown in Figure 2.7b. Then the center of the cluster is determined and this is chosen as the new mean illustrated in figure 2.7c. By repeating the clustering step and the calculation of the new mean the data is split into k clusters with the nearest mean as shown in Figure 2.7d.



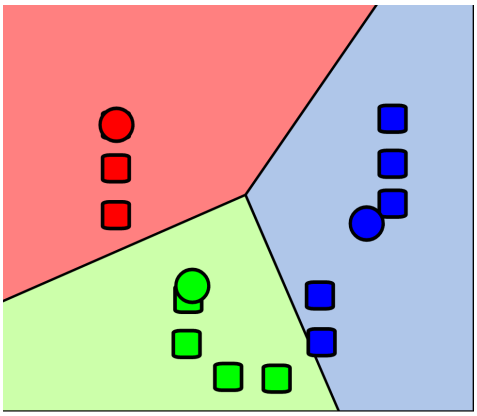
(a) $k = 3$ initial means are randomly selected in the data domain depicted color.



(b) $k = 3$ clusters are formed, by associating every observation with the nearest mean based on euclidean distance.



(c) Center of the clusters becomes the new mean.



(d) b and c are repeated until convergence.

Figure 2.7: 2D representation of the kmeans algorithm with $k = 3$. (a) The random initialization step, (b) assigning points to cluster, the partitions represent the Voronoi diagrams generated by the means, (c) calculation of new mean of the cluster, (d) repetition until final clusters are formed. Taken from [7].

Chapter 3

Method

In this chapter the overall approach to answering the research question is addressed. The setup and the data collection procedure are discussed, then a tool for data reduction is explained and created ending the chapter with a piece on analysis methods.

3.1 Overall approach to the research

The aim of this work is to answer the following research question, **Can a robust methodology be developed for reducing training set size while maintaining sufficient data density for accurate surrogate model development?** If this is achieved less computation is needed for creating a surrogate turbulence model. There are two main challenges: 1. It is not clear how much data is necessary to get a good regression. 2. The training data base can be slowly built over time, adding points in regions that are underpopulated. Adding points in overpopulated regions must be prevented. Therefore a way must be devised to control this. Training data sets are modular, and can be expanded over time.

In order to answer this question neural networks will be trained on data from an original data set, in which the input is obtained from experiment and the output is calculated using QuaLiKiz, which is then reduced by different amounts separately, in such a way that core features are kept. This data is then used to train neural networks. The current state of the art QuaLiKiz neural network architecture will be used in order to compare results. As described in Chapter 2, QuaLiKiz is a quasi linear gyrokinetics transport code, van der Plasche et al. have successfully trained a surrogate neural network for this transport code in 10D called QLKNN-10D [4]. In this thesis QuaLiKiz is the main workhorse for computing an input to output mapping from which the neural networks to be trained on. Experimental data from the JET tokamak is obtained and is used as input for QuaLiKiz, to calculate fluxes. As the input space is not continuous, there are regions of data space which are overpopulated and regions which are clearly underpopulated. On this input to output mapping, regression analysis is than done in the form of neural network training. By detecting high data regions using clustering techniques and reducing these, the research question can be answered. The general architecture of the neural network as described in [4] is kept, however input dimensionality is varied as will be explained shortly. A brief summary of the neural network is that it is a feed forward neural network consisting of 3 layers of 30 fully connected nodes, the activation function is the hyperbolic tangent, the cost function is defined as $C = C_{\text{good}} + \lambda_{\text{regu}} C_{\text{regu}} +$

$\lambda_{\text{stab}} C_{\text{stab}}$ where,

$$C_{good} = \begin{cases} \frac{1}{n} \sum_{i=1}^{n} (QLK_i - NN_i)^2, & \text{if } QLK_i \neq 0 \\ 0, & \text{if } QLK_i = 0 \end{cases} \tag{3.1}$$

$$C_{regu} = \sum_{i=1}^{k} w_i^2 \tag{3.2}$$

$$C_{stab} = \begin{cases} 0, & \text{if } QLK_i \neq 0 \\ \frac{1}{n} \sum_{i=1}^{n} NN_i - c_{stab}, & \text{if } QLK_i = 0 \end{cases} \tag{3.3}$$

This cost function was used using domain knowledge, based on physics [4]. For the training the Adaptive Moment Estimation (Adam) gradient descent algorithm was used with early stopping. For more information on these specific neural networks read [4].

The reason for choosing neural networks based on QuaLiKiz and not higher fidelity transport codes is:

1. QuaLiKiz itself has shown to be an accurate model for core turbulence in wide parameter space [14].

2. QuaLiKiz is much faster than higher fidelity transport models and thus training set generation with sufficient density is feasible.

The structure of higher fidelity input-output mapping is similar to QuaLiKiz, so results about data reduction are likely transferable. Meaning that if in this work a neural network regression training data set, can be reduced while keeping core features and thus keeping comparable accuracy for surrogate turbulence modeling this theoretically should also be possible with a higher fidelity code. The main goal is to give a proof of principle.

It is important to keep the core features of the original data set in the reduced data sets. Therefore it is important to distinguish what are core features. The main driver for core features is data density, a region containing higher data density than the rest of the data space, has in general more importance. Having high data density means that most experiments have been done in this region of parameter space. It is thus important to be able to detect regions in the data set where high data density is present. These high data regions are then to be reduced and the procedure to do so will be explained later in this Chapter.

Generally training data sets are not reduced but are preferably increased in amount of data. However since in this specific case the underlying model for which the surrogate model is made is computationally expensive the aim is to use the minimal amount of data while still providing sufficient data density for a good regression. to the best of our knowledge this type of research has not yet been done with the specific goal of optimizing training sets for physics-model neural network surrogates.

3.2 Explanation of setup and data collection

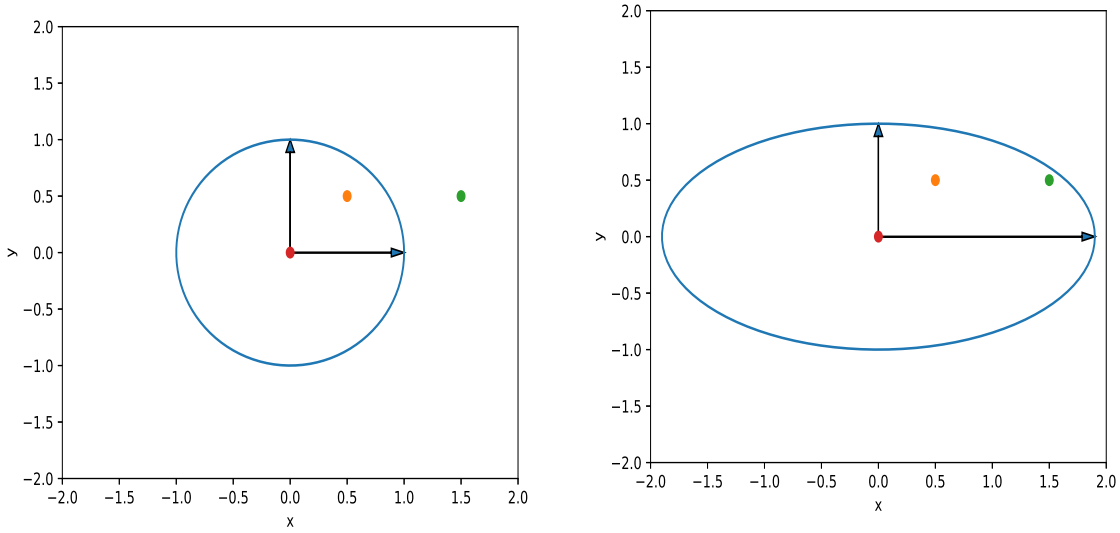
The data needed for neural network regression is two part, firstly a relatively small data set is constructed, with two input dimensions and one output dimension calculated with QuaLiKiz [39]. This allows for the tools and workflow to be applied, three dimension facilitate

visualization and the size reduces computational latency, allowing to develop the methodology faster. A grid scanning over R/L_{Te} and R/L_{Ti} the logarithmic electron and ion temperature gradient respectively ranging from 0 to 18 in steps of 0.1 is fed to QuaLiKiz with the other parameter settings as found in Appendix A. The chosen output dimension in this work is the electron heat flux q_e. Both inputs and outputs are bundled together into a vector which is then used for determining data density, meaning that the 3 values together create 1 data point. This data set is named the 3D data set, and is used for data processing by the tool introduced in the next subsection. In order to order to test how the tool generalizes to higher dimensional data, one dimension is added into the 3D data set, namely $R/L_{Ti,e}$ the logarithmic ion electron particle density gradient ranging from 2 to 4 in steps of 1, as this influences the electron heat flux the most. This is then applied to QuaLiKiz from which the electron heat flux is computed as output. The 4 dimensions combined lead to 1 data point. The 3D data set is a subset of the 4D data set. After generalization and development of the methodology, a new data set is constructed. This data set consist of experimental data gathered from experiments on the Joint European Torus (JET) which is located in Cullham in the United Kingdom combined with output parameters calculated using QuaLiKiz. The experimental data is processed using Gaussian process regression (GPR) techniques and for more information read [40]. The final number of input dimensions is 14, and is used to compute the electron heat flux with QuaLiKiz. These 15 dimensions combine to 1 data point. This data set is modular, the data set starts 1149975 points. New experiments are to be added to this data base later. The idea is to populate the data base over time. Using this database computed by data of JET experiments the boundaries of the neural networks are determined by experimental relevant data space. This also means that generalization to other tokamaks is not obvious. As neural networks tend to extrapolate badly.

3.3 The clustering procedure

In this section the tool for data reduction is explained. The goal of the tool is to identify high data density clusters, remove outliers, reduce data in high density regions by a reduction factor set by the user and return a data set which has had its high data density regions reduced. This tool is developed in Python 3.7.3, as it is a general purpose and high level programming language which is free to use, open source and works with the MARCONI tier-0 supercomputer at Cineca [41, 42]. In Chapter 2 the DBSCAN algorithm has been introduced as an algorithm which clusters data, based on spatial density. And is thus chosen as the algorithm to detect regions of high data density. The Python module scikit-learn is used at it contains an implementation of the DBSCAN algorithm.

Before supplying the DBSCAN algorithm with the input output mapping computed using QuaLiKiz there is a need for data preparation. As the DBSCAN algorithm is determining the neighbour points by euclidean distance, all data has to be scaled to unit variance and the mean is removed using the Standardize algorithm also from the scikit-learn module. In order to account for the different distributions in different dimensions the DBSCAN algorithm is adjusted slightly instead of having one radius of neighborhood ϵ each dimension gets its own length ϵ_p with $p = 1, 2, ..., d$ with d the number of dimensions. In Figure 3.1 a two dimensional example is shown, in Figure 3.1a the non modified DBSCAN algorithm is shown where the red point and the yellow point fall into the neighborhood radius. Lets now say that the x dimension has a much larger spread in data space and is considered to be a neighbor for



(a) Unaltered DBSCAN radii

(b) Adjusted DBSCAN radii to include various radii of neighborhood per dimension

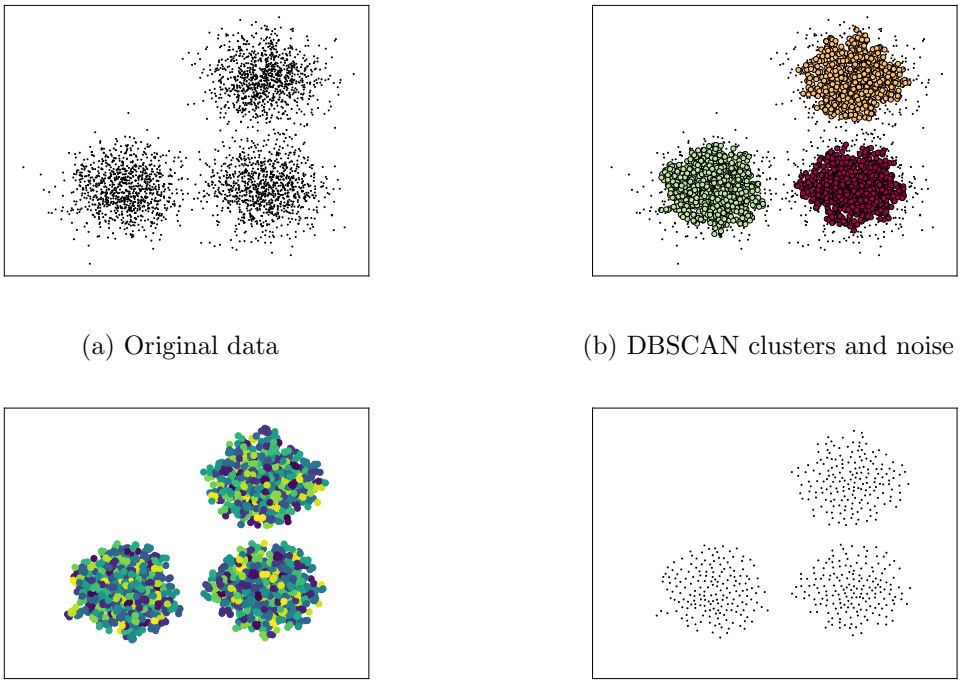
Figure 3.1: In (a) the DBSCAN algorithm is shown where all dimensions have the same radius of neighborhood. The yellow point is considered part of the cluster however the green point is not. In (b) the DBSCAN is adjusted in order to set different radii for various dimensions. By setting the radius of the x component to 1.8 and keeping the y component the same the green point now also becomes part of the cluster.

larger radii, in this case it is set to 1.8 reshaping the area of which neighborhood points fall into as shown in Figure 3.1b. This procedure makes the DBSCAN algorithm suitable for the application where different dimensions span over different orders of magnitude as is the case in this work, this is then translated to the unit variance size such that the radii of neighborhood ϵ can be given in normal units.

After the data preparation, the DBSCAN algorithm is applied to the data set, as can be seen in Figure 3.2. Data is either labeled part of a cluster or labeled as noise. The noise points are removed and the data in the clusters is categorised into two groups, group one the so called large clusters and group two the small clusters. Large clusters are clusters with more or equal points in the cluster than the set reduction factor. Small clusters in turn have less points in the cluster than the reduction factor. This distinction is made because when a cluster consists of fewer points than the reduction factor it can not be reduced by this amount. For example a cluster consisting of 5 points can not be reduced 10 times.

The next step in the process is the actual reduction, for this the kmeans algorithm is used to split the clusters found by DBSCAN into smaller clusters as can be seen in Figure 3.2c. In general the difficulty in using kmeans is that the user has to set k the amount of clusters that are expected. However in this case we know the number of points in a DBSCAN cluster and know the reduction factor which is set by the user giving us the value for k per DBSCAN cluster, in current application kmeans is applied to DBSCAN clusters which are scaled to unit variance, in this case there is no specific re-scaling per dimension as in the DBSCAN routine. The procedure is as follows, looping over the DBSCAN clusters, one by one each

DBSCAN cluster is reduced. If the DBSCAN cluster is a so called large cluster, kmeans is used to determine clusters in the DBSCAN cluster, every kmeans point in this kmeans cluster is than averaged over. One important note is that here not only is averaged over the input data but also the output, which is not good practice due to possible non linear behaviour of the output. In future work this should be addressed, for instance by looking for the nearest point to the center of the kmeans cluster and to pick this as the replacement point. For now this averaged value replaces all points found for that kmeans cluster.



(a) Original data

(b) DBSCAN clusters and noise

(c) Removed noise and kmeans clustering

(d) Reduced data

Figure 3.2: 2D representation of the reduction procedure, in (a) the original data is depicted, in (b) the data is clustered by the DBSCAN algorithm and noise is detected, in (c) the noise has been removed and kmeans clusters are determined, in (d) the reduced data set.

Now for the small DBSCAN clusters, there is another distinction to be made, extremely small clusters up to a value of roughly 6 points per clusters occur relatively often, as it is more likely to have for example 3 points together than to have 10 points together. The distribution can be seen in Figure 3.3. These extremely small clusters smaller than 6 points are left as is. As these extremely small clusters are not believed to be noise points which should be removed but have significance. The rest of the small DBSCAN clusters is simply averaged over as if it is a kmeans cluster in the large DBSCAN clusters. And thus this data is reduced less than the other parts of the data. After this procedure the original data set is reduced in areas where a high data density is present but kept in regions where information is sparse. This new reduced data set is then used to train neural networks with.

An important step for setting up the tool is determining ϵ_p, the maximum distance between two samples for one to be considered as in the neighborhood of the other, per dimension for

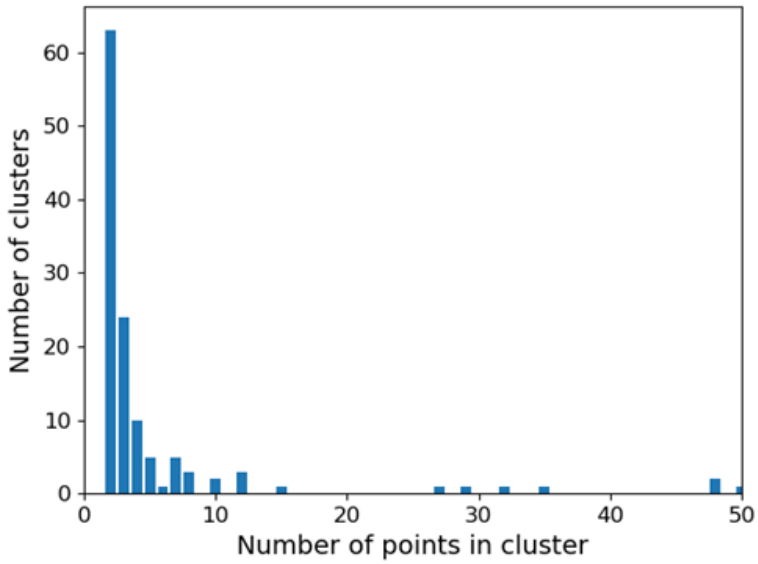


Figure 3.3: For the 3D artificial data set the number of clusters per number of points in the cluster, extremely small clusters < 6 occur relatively often.

the DBSCAN algorithm. The current approach is to looking at the distribution of the data set, excluding the top and bottom 0.01 percent of the data and than taking 10 percent of the span of the data meaning the $0.1 * (X_{max} - X_{min})$. Although arbitrary it is a reasonable starting point. As stated before points which are not put in a cluster are labeled noise, the amount of data labeled noise is also a gauge for tuning the DBSCAN settings. The idea is that if the noise percentage is low enough, the key features of the data are kept. What low enough is, is of course highly debatable and for the time being the aim is to keep this around 10 percent.

3.4 Methods of analysis

Now that new data sets are computed using the previously described reduction tool neural networks can be trained and be compared. However this comparison is not completely trivial, as there are many options to look at. The goal of the neural network is to be used as a surrogate model for turbulence modeling, thus it is important to capture how well the neural network can reproduce values from the model it is trying to surrogate, in this case QuaLiKiz. Standard procedures for this type of comparison are looking at the root mean square error (RSME) and the mean absolute error (MAE). The RMSE gives a relatively high weight to large errors and is thus more useful when large errors are particularly undesirable as a reminder RMSE is defined as,

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2} \tag{3.4}$$

And MAE is defined as,

$$\text{MAE} = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y}_j| \tag{3.5}$$

Computing these for the neural networks gives a very fast/easy number to compare the neural networks. There is random initialisation of the weights in the neural network training and hence different local minima are reached in the optimization. Therefore multiple neural networks are trained with the same initial settings which are called committee networks. committee removes noise by averaging over a number of such optimisations. In this work for every reduction factor 10 equal initialized neural networks are trained and then the mean and standard deviation of the RMSE and MAE is computed. Therefore we obtain some idea of the error in the neural networks.

Chapter 4

Analysis

In this chapter the main results of the practical work are presented, explained and discussed. As explained in the method chapter the first case that will be examined is the created 3D data set. Due to the case being 3D it is possible to visualize what happens in each step of the reduction process. Neural networks are trained on different reduced data sets and the RMSE and MAE are determined. The next case is the 4D case where similar trends in the RMSE and MAE are observed as in the 3D case. The final section of this chapter is on an experimental JET case, where the same procedure is applied, as is to the 3D and 4D case. This shows that reducing such a high dimensional experimental data set, leads to similar RMSE and MAE as the original data set as long as the reduction factor is not increased too much.

4.1 3D case

4.1.1 Reduction procedure

In the previous chapter the construction of the 3D data set was explained, using QuaLiKiz and the parameters as can be found in Appendix. Figure 4.1, shows the calculated data set, which from now on will be referred to as the original data set. This original data set contains 32761 data points, the ion heat flux q_e ranges from 0 to 53.5 Gyrobohm units.

The next step is setting up the radii ϵ_p for the DBSCAN algorithm. As in this case all points in both R/L_{Te} and R/L_{Ti} are separated 0.1 from each other $\epsilon_{R/L_{Te}}$ and $\epsilon_{R/L_{Ti}}$ are set to 0.11. For the ϵ_{q_i} the value of 5 was chosen as it is approximately 10 percent of the span of the parameter. With these settings the DBSCAN algorithm can be applied to the original data set. The algorithm detects 170 clusters and detects 537 noise points which is 1.6% of the original data set. The noise in this case is due to occasional jitter in the QuaLiKiz eigenvalue solver leading to spurious output. In Figure 4.2 each cluster is displayed with a specific color. The first thing that can be seen from this Figure is that the dark red is an extremely large cluster consisting of 28723 points which is 87,6% of the original data set, and is smooth.

In Figure 4.5 the 537 noise points are depicted in black. These points are considered outliers and are removed from the data set.

After removing the noise points from the original data set, the algorithm goes through the DBSCAN clusters depicted in Figure 4.2 one by one and reduces these clusters. As an example the large orange cluster from Figure 4.2 is taken. This cluster consists of 1946 points. The kmeans algorithm is applied to this cluster, and the reduction factor is set to 10. k the

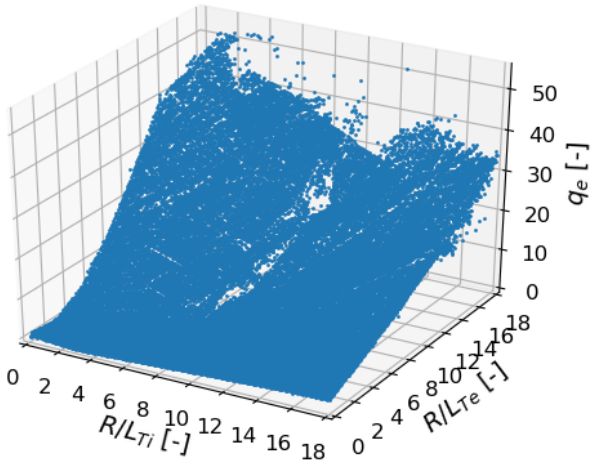


Figure 4.1: Original data set containing 32761 data points created using QuaLiKiz where R/L_{Te} and R/L_{Ti} are input parameters and q_e in Gyrobohm units is the output calculated.

number of kmeans clusters in this cluster is thus $k = \text{round}(1946/10) = 194$ where round() is a function that rounds the value between brackets down. In Figure 4.4 the 194 kmeans clusters are shown each with an individual color. Each cluster is then averaged over and this average replaces all points in this cluster, meaning that on average 10 points are reduced to 1 point.

After all DBSCAN clusters have gone through the kmeans procedure, the data set shown in Figure 4.1 remains, and this is called the reduced data set. This reduced data set consists of 3208 points and is 9.8 % of the original data set. This is slightly less than 10% due to the removal of the noise points.

4.1.2 Training

Using the previous described reduction procedure, multiple data sets are created each with a specific reduction factor. Using the original data set, and using DBSCAN also a data set is created which is not reduced by a reduction factor but where the noise is filtered out, from now on this will be referred to as the filtered data set. Per data set 10 neural networks are trained. Then the trained neural networks are compared via RMSE and MAE to the original data set and the filtered data set individually. The mean and standard deviation are then calculated and are plotted in the Figures 4.6. Figure 4.6a shows the RMSE of the neural networks compared to the original data set per reduction factor. Up to a reduction factor of 10 the RMSE stays roughly constant, this is an indication that these data sets are similar enough to the original data set. After the reduction factor 10 the RMSE starts to increase. In Figure 4.6b the RMSE of the neural networks compared to the filtered data set is depicted, except for the point at reduction factor 2 a similar trend as in the 4.6a is observed. Figure 4.6c shows the MAE compared to the original data set and again shows a similar trend to the RMSE although the increase in MAE is not as sharp as in the RMSE case. The last

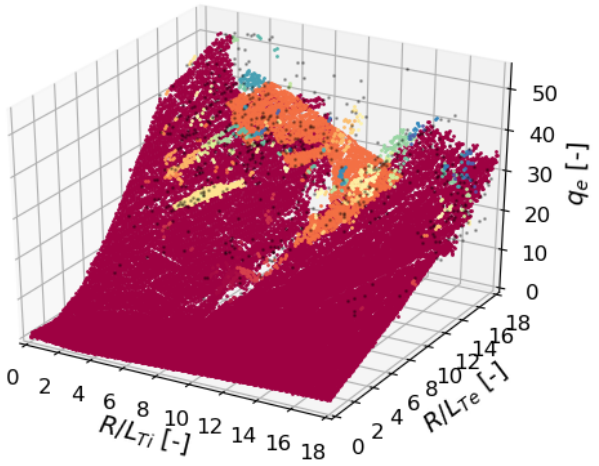


Figure 4.2: Original data set split into DBSCAN clusters, each color corresponds to one DBSCAN cluster with q_e in Gyrobohm units.

comparison is the MAE of the neural networks to the filtered data set in Figure 4.6d. As in Figure 4.6b the first point is a bit high, but the rest of the figure show a similar trend.

Figures 4.6 show a similar trend that for low reduction factors (low meaning less than 10) both RMSE and MAE stay approximately the same. When the data sets gets sparse, by too much reduction, the RMSE and MAE start to increase. Although the absolute values in the figures have no meaning, they can be compared to each other. The absolute values for the filtered data set are lower than those of the original data set, this is expected as the neural networks are trained on the filtered data set and have never seen the noise which is included in the original. Also a bigger difference in the absolute sense is seen in the RMSE than in the MAE which is also expected as the errors in RMSE are squared before they are averaged, the RMSE gives a relatively high weight to large errors compared to MAE. In order to check whether these trends are not an artifact of the regular grid spacing in R/L_{Te} and R/L_{Ti}, the same work was done on a 3D case where values for R/L_{Te} and R/L_{Ti} where chosen randomly between 0 and 18. These simulations with an irregular grid gave similar results within the error bars shown in Figure 4.6.

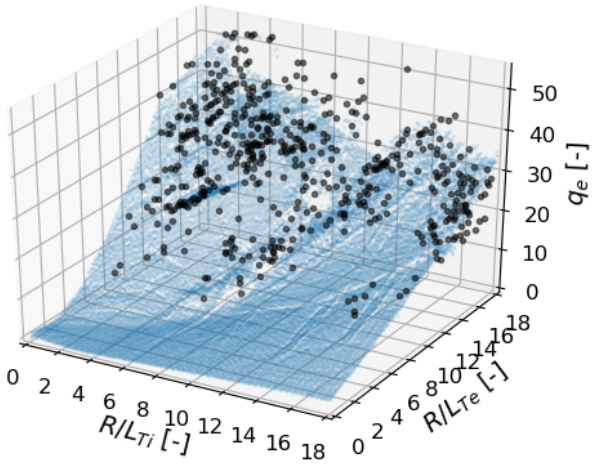


Figure 4.3: All points that do not fall into a DBSCAN cluster are labeled noise points are depicted in black. These noise points are removed from the original data set. q_e in Gyrobohm units.

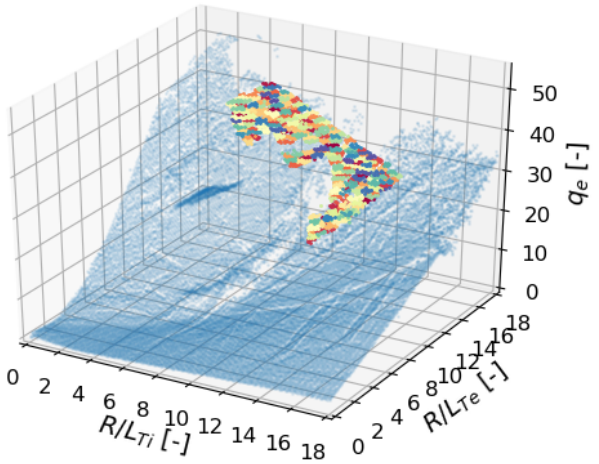


Figure 4.4: One DBSCAN cluster is clustered into 194 kmeans clusters, each kmeans cluster is depicted in a specific color. Every kmeans cluster is averaged over and replaced by one point. q_e in Gyrobohm units.

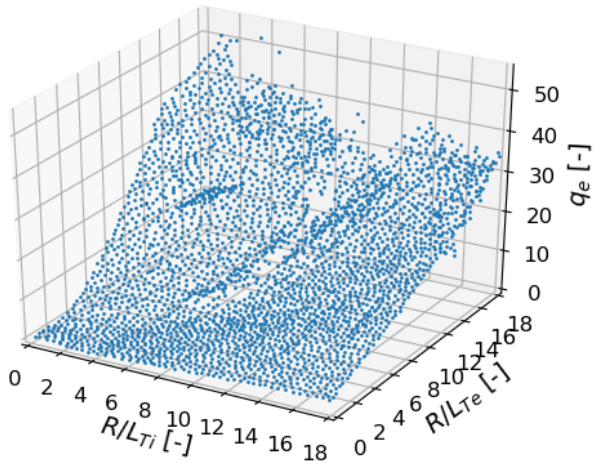
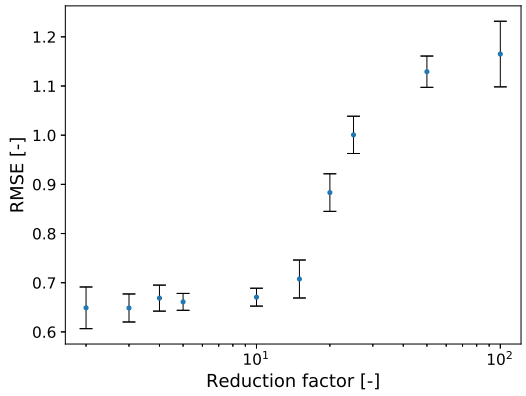
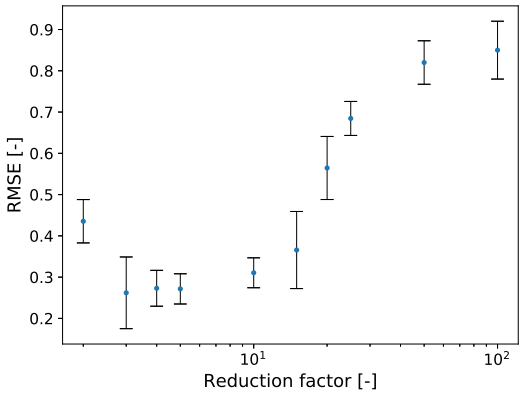


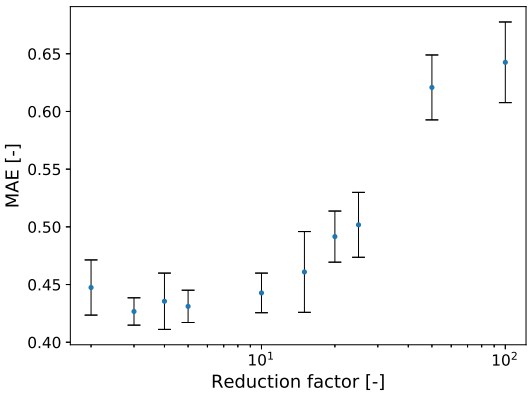
Figure 4.5: Reduced data set, the final data that is left after the clustering routines, in this specific case the reduction factor was set to 10, and the reduced data set consists of 3209 points. q_e in Gyrobohm units.



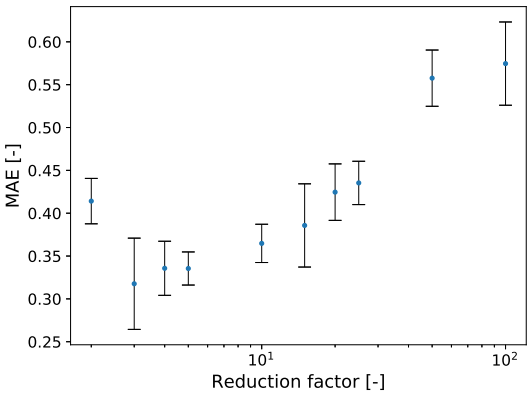
(a) RMSE original data set



(b) RMSE filtered data set



(c) MAE original data set



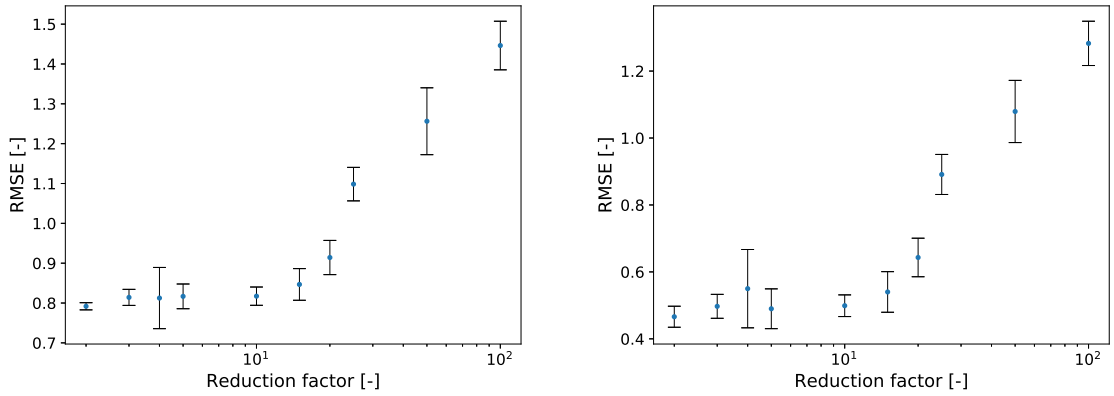
(d) MAE filtered data set

Figure 4.6: In (a) and (b) depicted the RMSE for the neural networks trained on the original and filtered data sets respectively, in (c) and (d) the MAE is shown for the neural networks trained on the original and filtered data sets respectively. All show that accuracy of the neural network stays rather constant up to a certain threshold which in this case is a reduction factor of roughly 10, after which the errors start to increase due to the fact that the training data becomes too sparse.

4.2 4D case

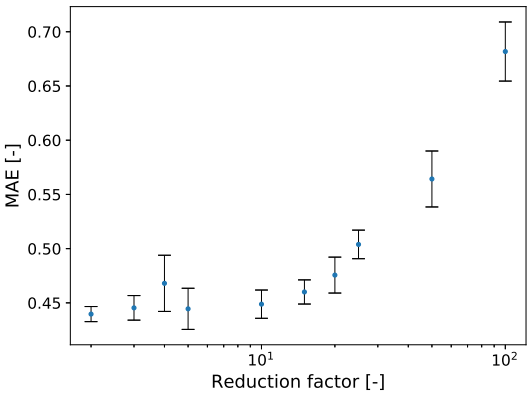
In this section the 3D case is expanded by including R/L_{ne}, keeping both R/L_{Ti} and R/L_{Ti} from 0 to 18 every 0.1 and R/L_{ne} 2, 3 and 4. With these settings q_e is again calculated using QuaLiKiz ranging from 0 to 128.4 Gyrobohm units. By the addition of the 3 values for R/L_{ne} the data set has tripled in size and is thus 98283 points. Similar to the 3D case this data set is referred to as the original data set. The next step is setting up the DBSCAN parameters $\epsilon_{R/L_{Te}}$ and $\epsilon_{R/L_{Ti}}$ are set to 0.11, $\epsilon_{R/L_{ne}}$ is set to 1.1 and ϵ_{q_e} is set to 8, as this was 10 % of the range of the parameter after removing 0.1 quantile, thus the value of 128.4 reported earlier as a maximum is labeled an outlier, it is however not yet removed, this will be labeled as a noise point and removed by DBSCAN after the DBSCAN procedure. The DBSCAN detects 2598 clusters and detects 2273 noise points which 2.3% of the original data set, from this the filtered data set is created by removing the noise points from the original data set. Using kmeans on the filtered data set with different reduction factors, the reduced data sets are created. Per reduced data set 10 neural networks are trained, starting with the same initial settings. These neural networks are than compared to the original and the filtered data sets by calculating RMSE and MAE. The mean and standard deviations are plotted in Figures 4.7. Figure 4.7a shows the RMSE of the neural networks with the original data set per reduction factor. Similar trends as in the 3D case are observed, the absolute values on the axis differ, but up to a reduction factor of 10, RMSE is almost constant and starts to increase for higher reduction factors. In Figure 4.7b the RMSE with the filtered data set is displayed, with again a very similar trend, although the absolute values do not matter, the values are lower than in Figure 4.7a which is to be expected as the neural networks are trained on the filtered data set, and will not predict the noise points in the original data set. Figure 4.7c show MAE of the original data set shows a similar trend, and Figure 4.7d also depicts a similar trend.

Figures 4.7 confirm that reducing data sets does not reduce accuracy as long as the reduction factor is kept under a certain threshold, in this case the threshold is at a reduction factor of 10. Since the 3D and 4D case are not extremely different, a subset of the 4D case is actually the complete 3D case it is not peculiar that the reduction factor threshold is at the same value as in the 3D case.

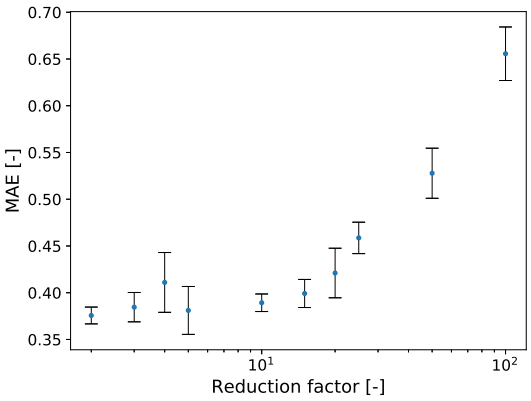


(a) RMSE original data set

(b) RMSE filtered data set



(c) MAE original data set



(d) MAE filtered data set

Figure 4.7: In (a) and (b) depicted the RMSE for the neural networks trained on the original and filtered data sets respectively, in (c) and (d) the MAE is shown for the neural networks trained on the original and filtered data sets respectively. All show that accuracy of the neural network stays rather constant, up to a certain threshold, which in this case is a reduction factor of roughly 10. After the threshold the errors start to increase due to the fact that the training data becomes too sparse.

4.3 JET case

The first important change in this data set is that QuaLiKiz was used not to compute q_e but only the ETG part of the q_e thus $q_{e,\mathrm{ETG}}$. Because individual modes capture the threshold behaviour discussed in the cost function of the neural networks and is the dominant driver of electron heat flux. The original data set created from JET data, consists of 11499975 data points and has in total 15 dimensions, thus 14 input and 1 output dimension. Table shows the ϵ settings for the DBSCAN algorithm. Where R/L_{ne} is the logarithmic electron density gradient, R/L_{Te} logarithmic electron temperature gradient, v_{tor}/c_{ref} is the toroidal velocity normalized to the sound speed, x the midplane-averaged minor radius of flux surface, normalized by midplane-averaged minor radius of last-closed-flux-surface, Z_{eff} is the effective ion charge, γ_E is the perpendicular $E\times B$ flow shear, q the safety factor, $r\frac{q'}{q}$ the magnetic shear, α the 'MHD alpha', related to Shafranov shift, and proportional to the pressure gradient, $R/L_{ni,0}$ the logarithmic ion density gradient, $R/L_{Ti,0}$ the logarithmic ion temperature density, $T_e/T_{i,0}$ the ratio electron temperature ion temperature, $\log\nu^*$ logarithmic collisionality, the reason $\log\nu^*$ is used instead of the collisionality ν is that ν^* scales many orders of magnitude in the data set. And finally $q_{e,ETG,GB}$ the electron ETG heat flux in Gyrobohm units.

Table 4.1: Parameters in the JET rotation database ϵ values used for the DBSCAN algorithm.

Parameter	ϵ_p
R/L_{ne}	1.2
R/L_{Te}	1.5
v_{tor}/c_{ref}	0.08
x	0.07
Z_{eff}	0.40
γ_E	1.60
q	0.26
$r\frac{q'}{q}$	0.24
α	0.13
$R/L_{ni,0}$	3.2
$R/L_{Ti,0}$	1.7
$T_e/T_{i,0}$	0.005
$\log\nu^*$	0.11
$q_{e,ETG,GB}$	0.1

With these settings, the DBSCAN algorithm detects 1423744 noise points which is 12.4% of the original data set. The noise is again removed from the original data set and a filtered data set is created. Neural networks are trained on the original and filtered data set, for different reduction factors and the mean and standard deviation are plotted in Figures 4.9. Due to the distinction made between extremely small clusters and other clusters there is a limit to the amount of reduction possible. As seen in Figure 4.8 a reduction factor larger than 100 does not reduce the data points per dimension anymore, because all of the data is in these extremely small clusters. This is due to the fact that a certain percentage of the data set falls in the extremely small data group which is left as is. Because of this no neural networks are trained on data sets with a larger reduction factor than 100. Figure 4.9a shows

the RMSE of the original data set, although for large reduction factors the RMSE tends to go up it is not as clear, as it is in the 3D and 4D RMSE original case. In Figure 4.9b the trend is again visible the increase is at a reduction factor of about 30. In these RMSE plots the original data plot's values are a lot higher than those of the filtered one, this is due to the high amount 12.4% of the data, that the neural networks have never seen. Both 4.9c and 4.9d show MAE for the original and the filtered data set respectively. The same trend is observed, for low reduction factors the MAE is rather constant up to a threshold, in this case somewhere around a reduction factor of 30, after which the MAE increases. Although 4.9c ranges over a larger range than 4.9d the difference between original and filtered are not as apparent as in the RMSE case, this is believed to be because MAE is less sensitive to outliers.

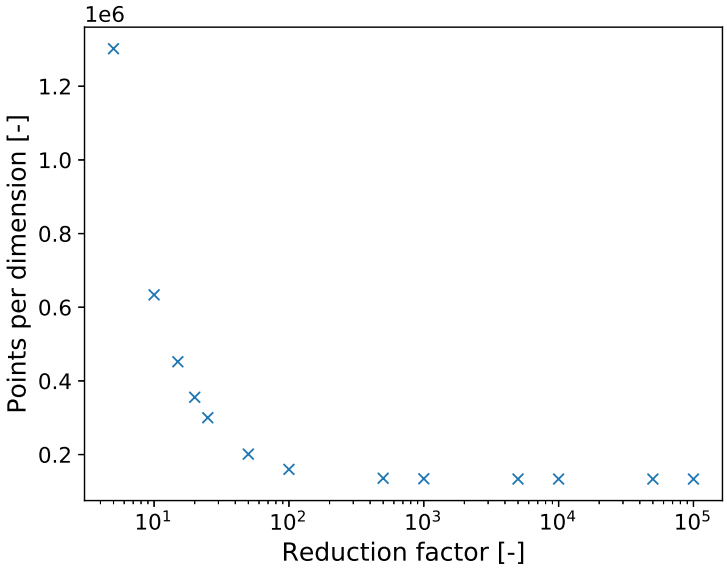
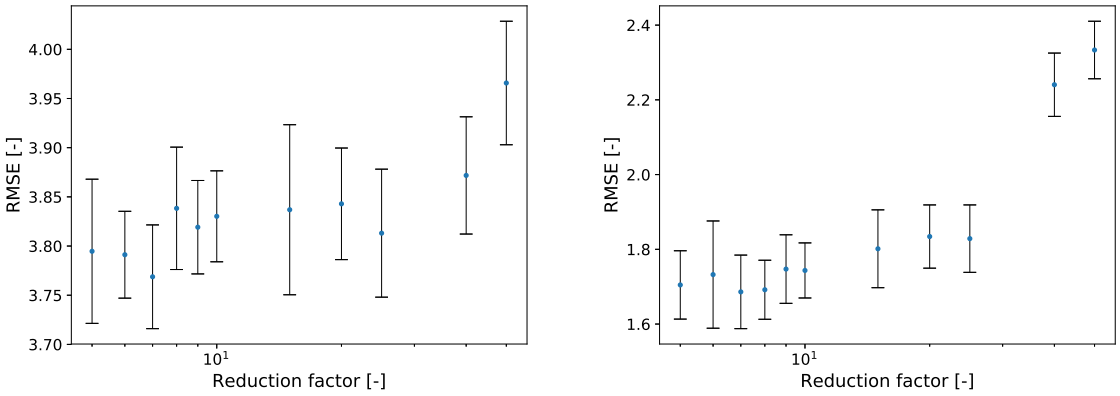
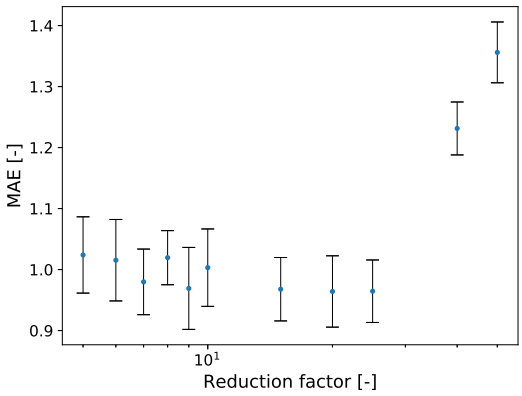


Figure 4.8: Points per dimension for different reduction factors. Due to the distinction between extremely small clusters and other clusters there is a limit to which the data set can be reduced. For the JET rotational data base this limit is encountered around a reduction factor of 100, after which almost all points are part of an extremely small cluster and are thus not reduced.

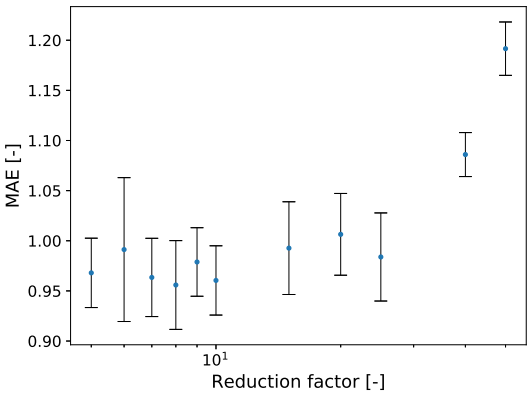


(a) RMSE original data set

(b) RMSE filtered data set



(c) MAE original data set



(d) MAE filtered data set

Figure 4.9: In (a) and (b) depicted the RMSE for the neural networks trained on the original and filtered data sets respectively, in (c) and (d) the MAE is shown for the neural networks trained on the original and filtered data sets respectively. All show that accuracy of the neural network stays rather constant up to a certain threshold which in this case is a reduction factor of roughly 25 after which the errors start to increase due to the fact that the training data becomes too sparse.

Chapter 5

Conclusions and outlook

The creation of a surrogate neural network for turbulence modeling is needed in order to achieve computational speed ups for real time application, such as scenario design and control. In this study, the neural network is trained on data calculated by a physics based model, QuaLiKiz, a quasi linear gyrokinetics code predicting plasma transport due to microturbulence. With the QuaLiKiz code it is still reasonable to over populate the data space, however with higher fidelity codes such as GENE over populating certain regions of data space is too costly. Therefore in this thesis reduction of high density data space is investigated. The research question for this work is: Can a robust methodology be developed for reducing training set size while maintaining sufficient data density for accurate surrogate model development?

In order to investigate this, a data reduction tool is built combining two clustering algorithms with which high data density regions are detected and split up into smaller portions to be reduced. For detection of data density, the DBSCAN algorithm was modified in such a way that in each dimension the maximum distance between two samples for one to be considered as in the neighborhood of the other can be set, in this way accounting for distinctive data distributions. The kmeans algorithm is then used on each DBSCAN detected cluster to split up the cluster into k clusters, where k is determined by the reduction factor and the size of the DBSCAN detected cluster. These smaller kmeans clusters are then averaged over and replaced by the average.

This tool was then applied to a 3D and 4D data set in order to facilitate visualization and reduce computational latency while developing the methodology. These data sets were carefully constructed in order to have certain characteristics in the output space. With the tool multiple reduced data sets have been created on which committee neural networks have been trained. The RMSE and MAE were calculated between the neural networks and the original and a filtered data set. The trend observed was as expected, that up to a certain reduction factor threshold, in these cases a reduction factor of 10, no significant change in the accuracy of the neural networks occurred. After this threshold, the overall accuracy reduced and the RMSE and MAE grew due to the data space being to sparse.

The procedure devolved on the 3D and 4D case was applied to a QuaLiKiz run database with 15D, based on inputs derived from a dedicated JET profile database and variants thereof. Such a database is modular and should be continued to be populated over time as more experiments are done. The observed trend is that, as in the 3D and 4D case RMSE and MAE are constant up to a certain reduction factor threshold. After this threshold, a similar loss in accuracy was detected. Therefore the research question: Can a robust methodology be

developed for reducing training set size while maintaining sufficient data density for accurate surrogate model development? can be answered with: yes, if the reduction factor is kept below a database dependent threshold value.

Now that the research question is answered, the question arises what can now be done which could not have been done before this work. The developed tool enables the control of data size of training sets used for QuaLiKiz surrogate model generation. The aim is to expand these training data sets over time when new profiles become available, either from experiments on the same or different machines. or from dedicated integrated modelling simulations. As new data points tend to have similar parameters to previous experiments, the developed tool is needed to reduce the data piling up in one region of data space, as this is computationally expensive. From the data set created with QuaLiKiz and the developed tool, the minimum data density can be determined. This is important for population of a database for higher fidelity models like quasilinear GENE from which than neural network surrogate models can be trained, reducing computational cost significantly by not overpopulating data space. For example, the JET case is found to be reducible by a factor 25, in order to stay on the safe side we pick a reduction factor of 20 and do the calculations. The starting data set consisted out of roughly 12 million data points, reducing this by a factor 20 leaves roughly 600 thousand data points. A quasilinear GENE simulation on average requires 5 CPU hours, this means a reduction on $(12 \times 10^6 - 600 \times 10^3) \cdot 5 = 570 \times 10^6$ CPU hours to 30×10^6 CPU hours. $30 \cdot 10^6$ CPU hours is still extremely large, but tractable if the data base is build up over time.

The reduction procedure uses the DBSCAN algorithm to detect regions of high density, which is determined by the settings the maximum distance between two samples for them to be considered neighbours. This means that a region either is dense enough or not, to be reduced. There is no distinction in how dense it actually is other than it being over the threshold density. In future work it is of interest to make this distinction as extremely dense regions might be reducible by a higher amount than other regions. Instead of having DBSCAN followed by kmeans, an intermediate step could be taken where within DBSCAN clusters another round of DBSCAN is employed with data density set to a higher level. This way potentially reducing data in regions of higher data density than the density threshold set by the first DBSCAN clustering even more.

In this work, data points belonging to one kmeans cluster are removed and are replaced by the average of those points, this is not good practice as this might introduce errors in the output dimensions due to the nonlinear output space. Therefore, the tool has been modified such that the center of the kmeans cluster is determined by the kmeans sklearn implementation and is used to find the closest point within the cluster to this center. This closest point is picked as the replacement of the whole cluster. By doing this, the reduced data set consists of actual data points calculated from the QuaLiKiz code. This implementation has been done after writing the analysis part of this thesis and is thus not included in the results. Expectations are that this modifications have little effect but make the reduction procedure more trustworthy.

In this thesis the focus was on the reduction of data sets used for transport predictions for neural network training. However, the developed tool can be used in a much broader array of problems. Neural network surrogate models could be important in any field where computing full models is too expensive. In general, these simulations are of the multiscale modeling type, meaning that they embody multiple space and time scales [43]. Multiscale modelling is widely used in most areas of science and engineering [44], such material science [45], bio-medicine [46, 47, 48] and engineering [49].

The tool could be used in a similar fashion, using a simplified model to over populate data space. Reduce the data sets, train surrogate neural networks on these data sets and determine the optimum data population which can then be filled by using higher fidelity codes.

Another possible application could be used for visualization of extreme dense data, by representing densely populated data in lower density using the tool, one could identify regions of interest. In order to illustrate the idea in Figure 5.1 an over populated artificial data set is depicted in which three high density regions are hidden under the enormous amount of data in the figure. By applying the tool to this data set 5.2 is created, which shows the region of space where most information is. This idea could of course also be used in the inverse problem where the tool is used to detect regions of low data density. All though no real world examples are found within the scope of this work, there might be applications in high data density visualization, in fields such as astronomy, biology and geography.

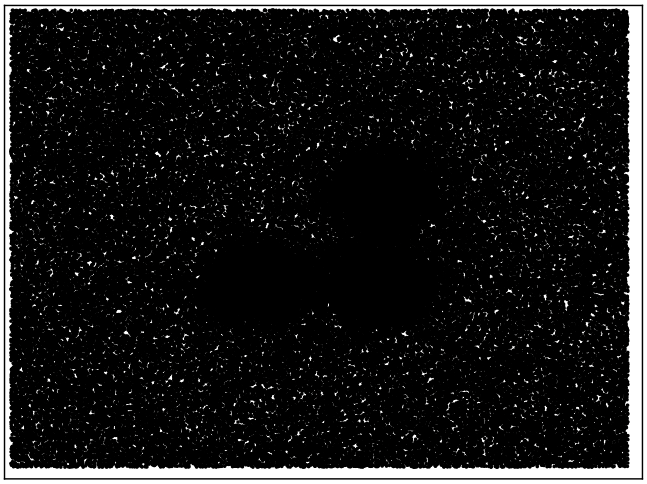


Figure 5.1: Overpopulated data set, data in black, three regions of high data density are 'hidden' under huge amounts of data.

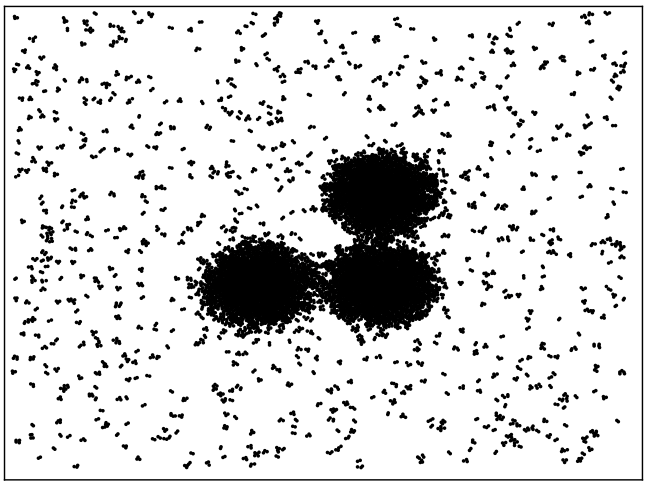


Figure 5.2: After applying the tool to the overpopulated data set, visualization of the three regions of high data density.

Bibliography

[1] BP. BP Statistical Review of World Energy Statistical Review of World. Technical report, 2019. ix, 1, 2

[2] Niek Lopes Cardozo. Fusion on the back of an envelope About this course ; its place in the MSc Fusion curriculum . 2018. ix, ix, 2, 3, 6, 7

[3] Garud Snoep. Steady-state integrated impurity transport modelling of DIII-D discharges. (March):1–74, 2019. ix, 8

[4] K. L. Van De Plassche, J. Citrin, C. Bourdelle, Y. Camenen, F. J. Casson, V. I. Dagnelie, F. Felici, A. Ho, and S. Van Mulders. Fast modeling of turbulent transport in fusion plasmas using neural networks. *Physics of Plasmas*, 27(2):1–17, 2020. ix, ix, 3, 11, 12, 15, 18, 19

[5] Michael Nielsen. Neural Networks and Deep Learning, 2019. `http://neuralnetworksanddeeplearning.com/` accessed on: 28-02-2020. ix, 13, 14

[6] DBSCAN Wikipedia, 2020. `https://en.wikipedia.org/wiki/DBSCAN` accessed on: 28-02-2020. ix, 16

[7] Kmeans wikipedia, 2020. `https://en.wikipedia.org/wiki/K-means{_}clustering` accessed on: 26-02-2020. ix, 17

[8] Iea. Global Energy & CO2 Status Report. Technical report, 2018. 1

[9] Ari Kahan. EIA projects nearly 50% increase in world energy usage by 2050, led by growth in Asia, 2019. 1

[10] M. R. Morgan. *Climate change 2001*, volume 59. 2004. 1

[11] Iea. Global Energy & CO2 Status report 2017. Technical report, 2017. 1

[12] Shell energy scenarios to 2050. *Energy*, page 52, 2008. 1

[13] Jeffrey P Freidberg. *Plasma Physics and Fusion Energy*, volume 1. 2007. 2

[14] J Citrin, C Bourdelle, F J Casson, C Angioni, and N Bonanomi. Tractable flux-driven temperature, density, and rotation profile evolution with the quasilinear gyrokinetic transport model QuaLiKiz. *Plasma Physics and Controlled Fusion*, 59, 2017. 3, 19

[15] V Koelman. *From Logistic Regression to Deep Learning*. 2019. 5, 12

[16] Michael A. Lieberman and Allan J. Lichtenberg. *Discharges and Materials Processing Principles of Plasma Discharges and Materials*. 1994. 7

[17] Florian Merz. *Gyrokinetic simulations of Multimode Plasma turbulence*. PhD thesis, Westfalischen Wilhems-Universitat Munster, 2008. 7

[18] X. Garbet, Y. Idomura, L. Villard, and T. H. Watanabe. Gyrokinetic simulations of turbulent transport. *Nuclear Fusion*, 50(4), 2010. 7, 8

[19] John Wesson. *Tokamaks*. Oxford University Press, 4th editio edition, 2004. 8, 9, 10

[20] L Hinton and R D Hazeltine. Theory of plasmasia transport in toroidal confineraient. *Reviews of Modern Physics*, 48(2):239, 1976. 9

[21] Lung Cheung and Wendell Horton. Equilibrium and electrostatic stability theory of tokamaks from the drift-kinetic equation. *Annals of Physics*, 81(1):201–230, 1973. 9

[22] R. D. Hazeltine. Recursive derivation of drift-kinetic equation. *Plasma Physics*, 15(1):77–80, 1973. 9

[23] E. A. Belli and J. Candy. Kinetic calculation of neoclassical transport including self-consistent electron and impurity dynamics. *Plasma Physics and Controlled Fusion*, 50(9), 2008. 10

[24] Felix I. Parra, Michael Barnes, Iván Calvo, and Peter J. Catto. Intrinsic rotation with gyrokinetic models. *Physics of Plasmas*, 19(5), 2012. 10

[25] I. Parra Felix and J. Catto Peter. Turbulent transport of toroidal angular momentum in low flow gyrokinetics (Plasma Physics and Controlled Fusion (2010) 52 (045004)). *Plasma Physics and Controlled Fusion*, 52(5), 2010. 10

[26] T. Görler, X. Lapillonne, S. Brunner, T. Dannert, F. Jenko, F. Merz, and D. Told. The global version of the gyrokinetic turbulence code GENE. *Journal of Computational Physics*, 230(18):7053–7071, 2011. 10

[27] C Bourdelle, J Citrin, B Baiocchi, A Casati, P Cottier, and X Garbet. Core turbulent transport in tokamak plasmas : bridging theory and experiment with QuaLiKiz. *Plasma Physics and Controlled Fusion*, page 14036. 10

[28] J. Candy, C. Holland, R. E. Waltz, M. R. Fahey, and E. Belli. Tokamak profile prediction using direct gyrokinetic and neoclassical simulation. *Physics of Plasmas*, 16(6), 2009. 10

[29] H. Sugama and W. Horton. Nonlinear electromagnetic gyrokinetic equation for plasmas with large mean flows. *Physics of Plasmas*, 5(7):2560–2573, 1998. 10

[30] Stefanie Braun. *Effect of Impurities on Kinetic Transport Processes in Fusion Plasmas*. PhD thesis, Ernst-Moritz-Arndt_universitat Greifswald, 2010. 11

[31] Simon Haykin. *Neural Networks - A Comprehensive Foundation*. second edition, 2005. 11

[32] Warren McCulloch and Walter Pitts. A Logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943. 11

[33] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958. 11

[34] Hinton, Geoffrey Teh, Yee-Whye. *IEEE Transactions on Neural Networks*, 17(6):1623–1629, 2006. 11

[35] David Rumelhart, Geoffrey Hinton, and Ronald Williams. Learning Representations by back-propagating errors. *Nature*, 323(9), 1986. 14

[36] Anne Bonner. The Complete Beginner's Guide to Deep Learning: Convolutional Neural Networks and Image Classification, 2020. 15

[37] Gupta Prashant. Regularization in Machine Learning. 15

[38] Martin Ester and Hans Kriegel. Density-Based Clustering Methods. *Comprehensive Chemometrics*, 2:635–654, 2009. 16

[39] C. Bourdelle, X. Garbet, F. Imbeaux, A. Casati, N. Dubuit, R. Guirlet, and T. Parisot. A new gyrokinetic quasilinear transport model applied to particle transport in tokamak plasmas. *Physics of Plasmas*, 14(11), 2007. 19

[40] A. Ho, J. Citrin, F. Auriemma, C. Bourdelle, F. J. Casson, Hyun Tae Kim, P. Manas, G. Szepesi, and H. Weisen. Application of Gaussian process regression to plasma turbulent transport model validation via integrated modelling. *Nuclear Fusion*, 59(5), 2019. 20

[41] Python, 2020. `https://www.python.org/` accessed on: 28-02-2020. 20

[42] SuperComputing Application and Innovation, 2020. `http://www.hpc.cineca.it/` accessed on: 01-03-2020. 20

[43] S. Alowayyed, T. Piontek, J. L. Suter, O. Hoenen, D. Groen, O. Luk, B. Bosak, P. Kopta, K. Kurowski, O. Perks, K. Brabazon, V. Jancauskas, D. Coster, P. V. Coveney, and A. G. Hoekstra. Patterns for High Performance Multiscale Computing. *Future Generation Computer Systems*, 91:335–346, 2019. 37

[44] Derek Groen, Stefan J. Zasada, and Peter V. Coveney. Survey of multiscale and multiphysics applications and communities. *Computing in Science and Engineering*, 16(2):34–43, 2014. 37

[45] James L. Suter, Derek Groen, and Peter V. Coveney. Chemically specifi C multiscale modeling of clay-polymer nanocomposites reveals intercalation dynamics, tactoid self-assembly and emergent materials properties. *Advanced Materials*, 27(6):966–984, 2015. 37

[46] Hannan Tahir, Carles Bona-Casas, Andrew James Narracott, Javaid Iqbal, Julian Gunn, Patricia Lawford, and Alfons G. Hoekstra. Endothelial repair process and its relevance to longitudinal neointimal tissue patterns: Comparing histology with in silico modelling. *Journal of the Royal Society Interface*, 11(94), 2014. 37

[47] D. Groen, J. Borgdorff, C. Bona-Casas, J. Hetherington, R. W. Nash, S. J. Zasada, I. Saverchenko, M. Mamonski, K. Kurowski, M. O. Bernabeu, A. G. Hoekstra, and P. V. Coveney. Flexible composition and execution of high performance, high fidelity multiscale biomedical simulations. *Interface Focus*, 3(2), 2013. 37

[48] Alfons G. Hoekstra, Saad Alowayyed, Eric Lorenz, Natalia Melnikova, Lampros Mountrakis, Britt Van Rooij, Andrew Svitenkov, Gabor Zavodszky, and Pavel Zun. Towards the virtual artery: A multiscale model for vascular physiology at the physics-chemistry-biology interface. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2080), 2016. 37

[49] R. Delgado-Buscalioni and P. V. Coveney. Continuum-particle hybrid coupling for mass, momentum, and energy transfers in unsteady fluid flow. *Physical Review E - Statistical Physics, Plasmas, Fluids, and Related Interdisciplinary Topics*, 67(4):13, 2003. 37

Appendix A

Settings QuaLiKiz 3D data set

"scan_type": "parallel",
"xpoint_base": {
"elec": {
"T": 8.0,
"n": 5.0,
"At": 9.0,
"An": 3.0,
"type": 1,
"anis": 1.0,
"danisdr": 0.0
},
"ions": [
{
"A": 2.0,
"Z": 1.0,
"T": 2.0,
"n": 1.0,
"At": 9.0,
"An": 3.0,
"type": 1,
"anis": 1.0,
"danisdr": 0.0
}
],
"meta": {
"phys_meth": 1,
"coll_flag": false,
"rot_flag": 0,
"verbose": true,
"separateflux": true,
"write_primi": true,
"numsols": 2,
"relacc1": 0.001,

"relacc2": 0.02,
"maxruns": 1,
"maxpts": 500000.0,
"timeout": 60,
"ETGmult": 1,
"collmult": 1,
"R0": 3.0,
"simple_mpi_only": 0
},
"special": {
"kthetarhos": [
0.01,
0.03,
0.05,
0.07,
0.09,
0.1,
0.13,
0.15,
0.17,
0.19,
0.21,
0.23,
0.25,
0.27,
0.29,
0.31,
0.33,
0.35,
0.37,
0.39,
0.41,
0.43,
0.45,
0.47,
0.49,
0.51,
0.53,
0.55,
0.57,
0.59,
1.0,
2.0,
3.5,
6.0,
10.5,
15,

19.5,
24,
30,
36
]
},
"geometry": {
"x": 0.5,
"rho": 0.5,
"Ro": 3.0,
"Rmin": 1.0,
"Bo": 3.0,
"q": 2.0,
"smag": 1.0,
"alpha": 0.0,
"Machtor": 0.0,
"Autor": 0.0,
"Machpar": 0.0,
"Aupar": 0.0,
"gammaE": 0.0
},
"options": {
"set_qn_normni": true,
"set_qn_normni_ion": 0,
"set_qn_An": true,
"set_qn_An_ion": 0,
"check_qn": true,
"x_eq_rho": true,
"recalc_Nustar": false,
"recalc_Ti_Te_rel": false,
"assume_tor_rot": true,
"puretor_abs_var": "Machtor",
"puretor_grad_var": "gammaE"
}
}