MASTER

Minimizing neoclassical transport in the Wendelstein 7-X stellarator using Variational autoencoders

van Rijn, Lennart P.

Award date:
2023

Minimizing neoclassical transport in the Wendelstein 7-X stellarator using Variational autoencoders

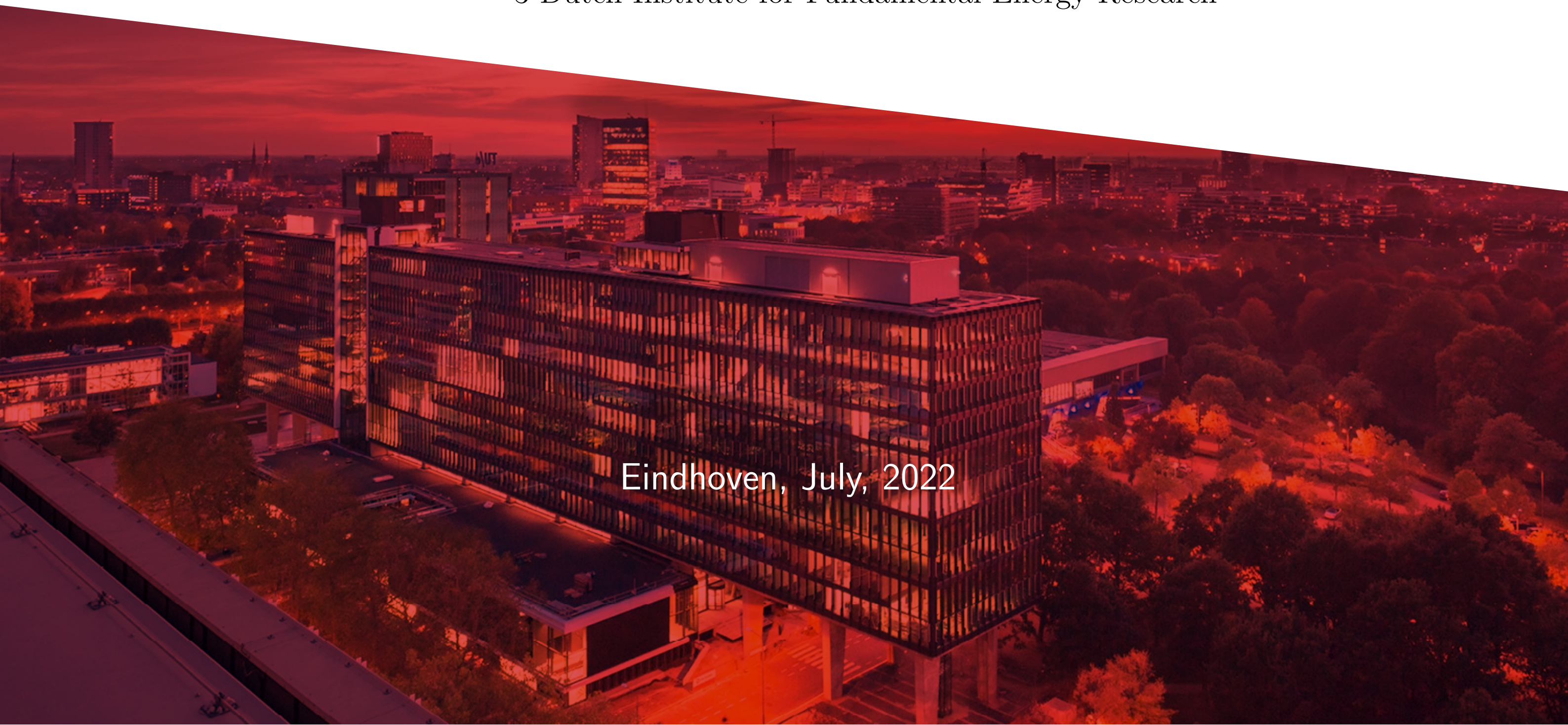
Master Thesis

Author:
Lennart van Rijn[1,2]

Supervisors:
Dr. J.H.E. Proll[1]
Dr. V. Menkovski[2]

Committee Members:
Dr. A.C. Jalba[2]
Dr. J. Citrin[1,3]
Dr. S. C. Hess[2]

1 Department of Applied Physics, Eindhoven University of Technology
2 Department of Mathematics and Computer Science, Eindhoven University of Technology
3 Dutch Institute for Fundamental Energy Research



Eindhoven, July, 2022

On a personal note

The last year has been a very interesting one, so to say. First of all because it seemed impossible to find a fitting project for my unusual combination of master's programs. Machine learning and Applied Physics were not the best of friends, and right at the moment I was considering dropping one of the programs I found this project with Josefine. For which I am truly grateful to have provided me with this opportunity. Besides this opportunity, she has guided me through the world of nuclear fusion, which was entirely new to me. I know it must have been challenging to guide a student with almost no knowledge of the matter. My sincere thanks!

Besides Josefine, I would also like to thank the rest of the stellarator group. The discussions during the 'donderdagmiddag borrel' and the dinners/cocktail tasting sessions were great. Some extra highlighting is deserved for my fellow graduating students with whom I've stood in the graduating trenches for an entire year. The synchronization of our biological coffee clocks after a year is truly remarkable. To all of you, you've created a very stimulating and enjoyable working environment, which was great to be a part of.

Also, thanks to Vlado for the discussions trying to solve my modeling challenges. Your dedication to understanding the underlying physical problem for all your students is impressive. This high-risk, high-reward project was not what I expected when I came to you for your supervision, but it definitely was very entertaining and educational. Furthermore, I would like to thank Andrei Jalba, Jonathan Citrin, and Sibylle Hess for being part of my graduation committee.

A special place in my thank words must be dedicated to Håkan and Yoeri. Håkan, I would like to thank you for supporting my machine learning ideas, even though you must have thought I was abusing your code for no good purpose. Also, many thanks for always taking the time to identify all the weird outcomes my methods unraveled. Yoeri, massive thanks for your implementation help, your very thorough feedback, and never-ending discussions about posterior collapse, even during the weekends. We have not won this battle, but it was most definitely fun to try.

Lastly, I want to thank my family and friends for their support and interest. I know I have acted like a cart riding the thesis roller coaster where results or the lack thereof might have affected my mood. Your support was endless, as was your patience in listening to my constant struggles.

DUM SPIRO SPERO!

Lennart van Rijn Eindhoven, July 2022

Abstract

Using first principle models to construct and optimize stellarator geometries is a computationally heavy - forbidding task. A fully optimized geometry would result in lower transport losses and can make the difference in the pursuit of feasible nuclear fusion. Because complete turbulence models do not exist yet and the extensive geometry parameter space, a data-driven approach is required. Deep learning methods hold the promise to deliver surrogate models that can explore this space much faster.

In this work, we study two generative machine learning models based on the Variational autoencoder (VAE), which are trained on simulated data of the Wendelstein 7-X stellarator. These models are leveraged to adjust the flux surfaces and the magnetic field on those surfaces with the goal of minimizing the particle and heat losses. The minimization algorithm finds a step-by-step optimization scheme based on the strength of the relations between the transport losses and input variables in the data. These relations could all be explained by known physical laws.

In addition to the generative capabilities of the models, we explore the autonomous design choices of the latent space by the models. During the investigation of the latent space, we identify a set of issues regarding the training procedure of VAEs on sufficiently complex data. Specifically, the effect of posterior collapse (a known limitation of VAEs) was a prevalent challenge in our implementation. In an effort to overcome the posterior collapse, many adaptations have been made. These did, however, not fix the posterior collapse in our project, but could be used as handles for future work.

List of Figures

Contents

Chapter 1

Introduction

1.1 Motivation & context

In 2014, the fifth assessment report of the IPCC (Intergovernmental Panel on Climate Change) alarmed about the consequences of climate change and the possible mitigation pathways consisting of reducing CO2 emissions to near zero[1]. In response, in 2015, President Obama announced drastic measures to reduce CO2 emissions and transition to a sustainable energy system[2]. Later that year, world leaders agreed in the Paris Agreement to limit global warming to 1.5 °C. This goal called for fast and drastic reductions in CO2 emissions. To accomplish these reductions, most countries rely heavily on solar and wind. However, these technologies consist of power fluctuations and geographic limitations. Nuclear fusion proposes a more reliable and less climate-sensitive energy source compared to many sustainable energy technologies and thereby could offer a valuable asset for the energy transition[3, 4].

Furthermore, nuclear fusion is a relatively clean technology as it does not produce toxic or greenhouse gasses. Secondly, fusion does not produce high-activity, long-lived nuclear waste as is produced in fission reactions. The activation of components in a fusion reactor is low enough for the materials to be recycled or reused within 100 years compared to 10.000 years needed for nuclear fission waste[5, 6]. Furthermore, the fuels used in fusion reactions are widely available and nearly inexhaustible. Deuterium can be distilled from all forms of water, while tritium will be produced during the fusion reaction as fusion neutrons interact with lithium in the device's walls[7]. Lastly, fusion does not use methods or materials with a possibility of a nuclear disaster or nuclear weapons.

Nuclear fusion is a process that naturally only occurs in stars. Fusion is the process of particles fusing to create energy. In stars, gravity is used to overcome particles' repulsive effects on each other. However, on Earth, other methods are required to overcome the repulsive effects. These methods use a magnetic field in a toroidal shape to avoid losses at the ends of the magnetic field. This magnetic field needs a twist to counteract the particle drifts caused by the curvature and the gradient of the magnetic field due to its toroidal shape. There are two main designs to create this twist, the tokamak and the stellarator. The tokamak uses currents in the plasma to create the twist, while the stellarator only uses external coils to create the necessary twist. Due to the external coils, many different magnetic field configurations are possible in a stellarator. However, not all of these configurations perform equally well. The gradient and curvature drifts are still present, and together with the effect of particle trapping, this can lead to unwanted transport of particles and heat out of the plasma, the so-called neoclassical transport. This neoclassical transport depends strongly on the exact shape of the magnetic field. This project focuses on the possible configurations of the Wendelstein 7-X stellarator (W7-X), located in Greifswald, Germany.

The range of possible configurations is enormous and, therefore, hard to optimize. In this work, we propose to address this challenge by developing a data-driven simulator based on deep generative models. Deep generative models can generate new data based on the data it has seen. A variational autoencoder (VAE) is a type of generative model able to do this by mapping the data to an intermediate latent representation cleverly. The complexity of neural network based models such as the VAE generally scale linearly with the number of parameters and the dimensionality of the input data. This is many orders of magnitude faster than the conventional methods where complex differential equations are numerically integrated. Furthermore, a VAE allows for optimization of the configuration using the fact that its components, neural networks, are differentiable and the discovery of hidden variables in the intermediate latent space.

This thesis presents a feasibility study demonstrating the capabilities of a variational autoencoder-like model on a simulated data set containing neoclassical transport. This project aims to show accurate simulating capabilities and discover known physical principles in the latent space.

1.2 Research questions

In the first part of this project, we implemented generative modeling using a variational autoencoder. The goal of this part is to show that a variational autoencoder can capture the distribution of the data, which leads to the following research question:

Can a variational autoencoder be used to generate synthetic results comparable to conventional simulation methods?

Which contains the relevant sub-questions:

- How accurate are the generated synthetic results?
- What roles do the technical limitations of using a variational autoencoder, such as the posterior collapse or blurry generation, play in the success of the solution?

Once the first questions have been answered, we identify new tasks for the generative model. The main idea of this project is to not only replace the simulation methods with faster machine learning alternatives but to use this model to optimize the fusion device and gain new insights. This leads to a second research question:

Which new tasks are possible using the generative model after it has proven to be an accurate simulator?

To answer the second main research question, several sub-questions need to be considered:

- Can we use the generative model to find magnetic configurations that minimize the transport losses?
- What adaptations to the generative model need to be made to allow for new tasks?
- Can the representation learned by the VAE (using the latent variables) be translated back to physical quantities?

1.3 Contributions

Answering the research questions leads to the main contributions of this thesis:

- We introduce several generative models designed to optimize the magnetic field configuration. Using these models, we demonstrate the ability to find multiple physical laws in the data. These identified physical laws can then be exploited to minimize the transport losses.
- We provide proof of principle methods for different tasks using generative models in nuclear fusion, including:
  - Simulating new, unseen data for a given set of input parameters. Thereby finding an alternative to the conventional expensive simulation methods.
  - Exploring latent space in search for relevant hidden variables and known physical principles.

This work is an exploratory project, where the application of generative models to the problem of simulating particle transport in stellarators is studied. During the project, the models suffered from a well-known variational autoencoder problem called the posterior collapse. This work provides some guidelines for overcoming the posterior collapse. However, due to the posterior collapse, not all of the original goals of this project were achieved. Nevertheless, this project delivers a number of outcomes that can inform subsequent efforts in this direction.

1.4 Overview

This thesis is for a double master's in applied physics and data science. Care is thus taken to provide all the necessary theories for both degrees. In Chapter 2, the basics of nuclear fusion are discussed by explaining the basic equations, the equation governing magnetic confinement, and the description of neoclassical transport and magnetic coordinates. Then, relevant machine learning theory is discussed in Chapter 3 by going through different model designs. Hereafter, an overview of literature related to this work is given in Chapter 4. Chapter 5 describes the data used in this project, after which the used models and the loss function are described in Chapter 6. The outcomes of this project and the discussion thereof are found in Chapter 7. Finally, Chapter 8 concludes this thesis.

Chapter 2

Background nuclear fusion and neoclassical transport

2.1 Nuclear fusion basics

Nuclear fusion is a reaction in which two or more atoms are combined to form one or more different atoms and other subatomic particles (neutrons and protons). The difference in total mass between the products and the reactants tells if energy is absorbed or released following the famous $E = mc^2$ equation, where E is the released or needed energy, m is the mass and c is the speed of light.

Normally, Deuterium(D) and Tritium(T) are used in nuclear fusion reactions performed on earth. Deuterium and Tritium are both isotopes of hydrogen, where the nucleus of Deuterium contains one proton and one neutron and the nucleus of Tritium contains one proton and two neutrons. The reaction of these two atoms produces one Helium atom and one free neutron:

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

The released energy per reaction is $E = 17.59\,\mathrm{MeV}$. Nuclear fusion is one of the most energetic reactions known to occur. As a comparison, The D-T reaction produces $3.4 \times 10^{8}\,\mathrm{MJ}$ of energy per kg of fuel, while the combustion of a kg of gasoline only produces 44 MJ. The nuclear fission process already mastered on earth produces almost the same amount of energy per kg of fuel. However, this process produces highly radioactive materials with half-lives of over 10^{4} years.

Regardless of the fusion method, the goal of all fusion power devices is to reach a state called ignition, a self-sustaining reaction. This occurs when the energy being given off by the fusion reactions heats the fuel mass more rapidly than various loss mechanisms cool it. The plasma conditions necessary for ignition to occur is described by the Lawson criterion, a lower bound on the fusion triple product, $nT\tau_E > 3 \times 10^{21}\,\mathrm{m^{-3}\,keV\,s}$, where n is the density in $\mathrm{m^{-3}}$, T is the temperature in keV and τ_E is the energy confinement time in seconds. With the densities and temperatures reached in experiments, a energy confinement time in the order of seconds is needed.

The energy confinement time is a measure of the insulation of the fusion plasma. It must be large enough to ensure that the energy released by the fusion products is equal to or larger than the energy lost. This confinement time is heavily dependent on the optimization of the magnetic field because of the influence the magnetic field has on the motion of the charged particles making up the plasma.

2.2 Particle motion in electro-magnetic fields

To achieve fusion and the Lawson criterion, the fuel needs to be heated to millions of Kelvins. Furthermore, all particles will be charged, i.e. ions or electrons. Therefore it is fundamental to understand how these particles move about. There is a difference between considering the movement of individual particles or a group of particles. In this section, only the movement of individual particles is described. Within this domain there is only one important force, the Lorentz force $\boldsymbol{F}_L = q(\boldsymbol{E} + \boldsymbol{v} \times \boldsymbol{B})$ with q the charge of the particle, $\boldsymbol{v}$ the velocity, $\boldsymbol{E}$ the electric field and $\boldsymbol{B}$ the magnetic field.

Gyration

Due to this Lorentz force, charged particles gyrate in a homogeneous magnetic field. The particles gyrate around a field line while the motion along this field line is not affected by the field. This gyration is called the cyclotron motion and it must be noted that positively and negatively charged particles gyrated the other way round. Equating the Lorentz force to the centripetal force leads to the following for electrons:

$$F = \frac{mv_\perp^2}{\rho_c} = qv_\perp B \tag{2.2}$$

where q is the charge of the particle and m is the mass of the particle. using the relation between angular velocity and orbit velocity it can be found that the electron cyclotron frequency and electron cyclotron radius are respectively given by

$$\omega_{ce} = \frac{eB}{m_e} \tag{2.3}$$

$$\rho_{ce} = \frac{m_e v_\perp}{eB} \tag{2.4}$$

Gyration is very fundamental to magnetically confined plasmas. Since the frequency depends only on the magnetic field, it is a strong resonance. This makes it possible to heat the plasma by using high power waves at the frequency resonant with the electron gyration.

E × B-drift

A charged particle moving in an electromagnetic field exhibits a drift in addition to its gyration and any acceleration due to a component of the electric field parallel to the magnetic field. This drift motion has velocity $\boldsymbol{v}_{E\times B} = \frac{\boldsymbol{E}\times\boldsymbol{B}}{B^2}$ and is therefore known as the 'E-cross-B' drift. This drift is dependent on the charge of the particle, as a result, both ions and electrons drift in the same direction with the same speed. This drift is shown in Figure 2.1B. If the force is not by an electric field for example gravity, the drift will be in opposite directions as can be seen in Figure 2.1C.

Grad-B drift

So far, only homogeneous fields have been used. However, these fields are rarely homogeneous and toroidal machines can by definition not have a homogeneous field. If there is a gradient in the magnetic field perpendicular to the direction of the field, the gyrating motion of the particles change. The cyclotron radius of charged particles depends on B, as can be seen in Equation 2.4, being smaller where the field is stronger. This results in a drift perpendicular to the gradient and the $\boldsymbol{B}$-field as can be seen in Figure 2.1D. Note that the drifts of the charged particles are in opposite directions, which in turn leads to an $\boldsymbol{E}$-field and thus $\boldsymbol{E} \times \boldsymbol{B}$ drifts. The grad-B velocity of this drift reads

$$\boldsymbol{v}_{\boldsymbol{\nabla} B} = \frac{1}{2}\rho_c v_\perp \frac{\boldsymbol{B} \times \boldsymbol{\nabla B}}{B^2} \tag{2.5}$$

Curvature-B drift

Just like the magnetic fields are rarely homogeneous, they are also rarely completely straight. For a charged particle to follow a curved field line, it needs a drift velocity out of the plane of curvature to provide the necessary centripetal force. This velocity is given by

$$\boldsymbol{v}_{curvB} = \frac{mv_{\parallel}^2}{eB}\frac{\boldsymbol{R}_c \times \boldsymbol{B}}{R_c^2 B} \tag{2.6}$$

where $\boldsymbol{R}_c$ is the radius of curvature pointing outwards, away from the center of the circular arc which best approximates the curve at that point. The curvature drift can also be defined using the curvature $\boldsymbol{\kappa}$ as $\boldsymbol{b}\boldsymbol{\nabla}\boldsymbol{b}$ where $\boldsymbol{b}$ is a unit vector in the direction of the magnetic field. This results[8] in

$$\boldsymbol{v}_{curvB} = \frac{\boldsymbol{b}}{\omega_{ce}} \times v_{\parallel}^2\boldsymbol{\kappa} \tag{2.7}$$

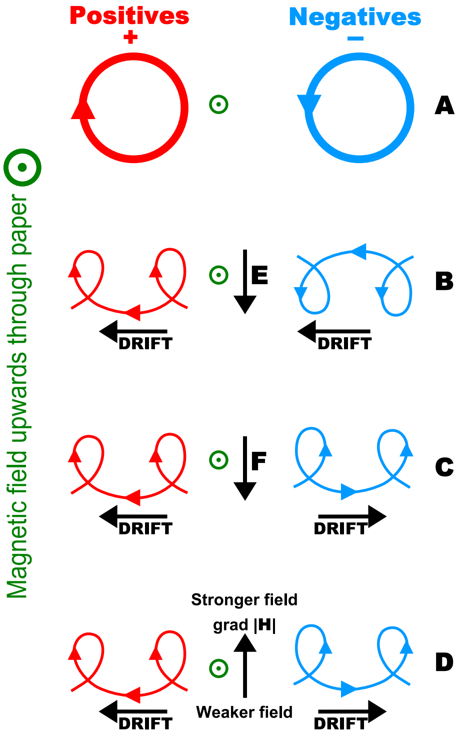


Figure 2.1: Charged particle orbits in a magnetic field. (A) No disturbing force, (B) With an electric field E, (C) With an independent force, F (e.g. gravity), and (D) In an inhomogeneous magnetic field, grad H. The parallel motion of the particles along the field lines is not shown in this figure. Source: Wikimedia

2.3 Magnetic confinement fusion

One of the most promising approaches to attain nuclear fusion on Earth is to use magnetic confinement fusion. Magnetic confinement fusion uses the theory of charged particles in electromagnetic fields explained in Section 2.2 to reach a high level of confinement. Magnetic confinement relies on the use of twisted magnetic toroidal fields to confine the particles and keep them from colliding with the walls. The twisting of the magnetic field is needed to compensate the drifts. When for example an electron is on the top half of the toroidal field, it will experience a grad-B drift downwards away from the magnetic field line it was following. Once the particle follows the twisted magnetic field line to the lower half of the torus the drift will still be downwards and the particle will drift back to the magnetic field line it was following. This cancellation is not perfect, but it provides enough confinement for the fuel to remain in the reactor for the needed amount of time. The two designs that solved this twisting challenge are the tokamak and the stellarator.

Tokamak

The tokamak was invented in 1952 by two Russian scientists, Tamm and Sakharov. The concept uses coils in the poloidal direction (short way round) to create a strong magnetic field. As stated above, only a toroidal field is not enough. Therefore an additional magnetic field is created by a transformer where the plasma acts as the winding. This causes a current through the plasma. As can be seen in Figure 2.2, the tokamak is almost axisymmetric. There is a slight ripple caused by the finite number of coils, this is not visible in the schematic. The almost axisymmetric property of the tokamak gives it high confinement properties compared to unoptimized non-axisymmetric devices. There are however also some disadvantages to this design. The plasma currents needed for the second magnetic field causes instabilities or disruptions which can damage the reactor. Furthermore, the transformer prevents the tokamaks from running continuously, which would be a nice property for reactors. Both of these disadvantages are absent in the other design, the stellarator.

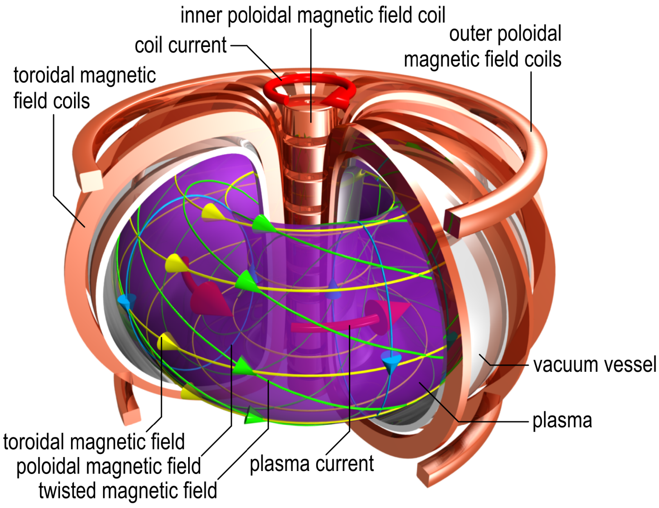


Figure 2.2: A schematic overview of the tokamak. Adapted from [8]

Stellarator

The stellarator was invented by Lyman Spitzer in 1951. To get the necessary twist he suggested twisting the torus into the shape of the number 8. This creates a twist in the magnetic field without needing the plasma current causing the main disadvantages of the tokamak. The figure 8 stellarator has already been archived but the idea remained. Nowadays, the twist is achieved by additional sets of coils or by designing special coils. As can be seen in Figure 2.3, the axisymmetric property is no longer present in stellarators. This led to the very poor confinement of the first stellarators. These confinement properties can be enhanced once another form of symmetry is created. This leads to concepts such as quasi-symmetry, quasi-isodynamicity, and omnigeity. These concepts will be discussed later in Section 2.4.3.

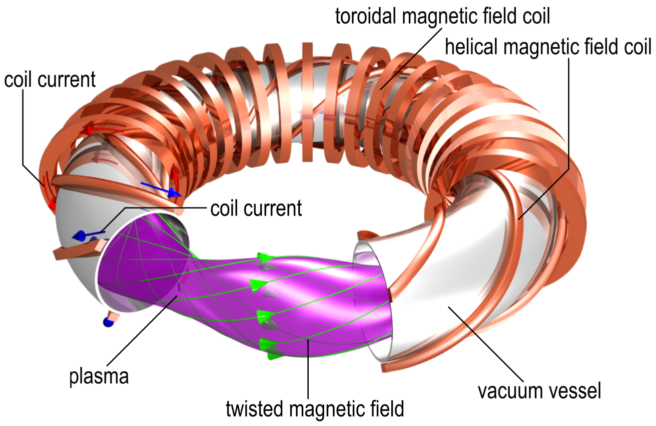


Figure 2.3: A schematic overview of the classical stellarator. Adapted from [8]

2.3.1 Coordinate systems

The toroidal geometries of the fusion devices can be described using different coordinate systems. First of all, it can be described using the classical cylindrical coordinates (R, ϕ, Z) where ϕ increases from 0 to 2π. A visualization of this coordinate system can be found in Figure 2.4.

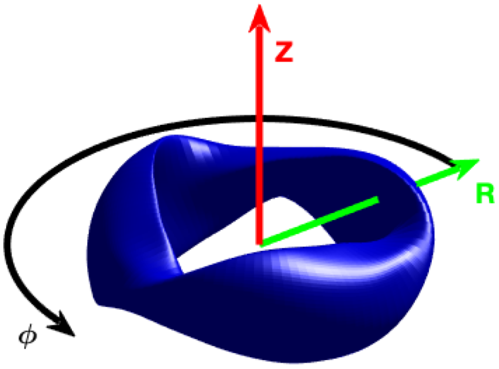


Figure 2.4: The standard cylindrical coordinate system: R measures distance from the $\hat{Z}$ axis and ϕ is the standard angle of the cylindrical coordinate system such that $\hat{R} \times \hat{\phi} = \hat{Z}$. Adapted from [9].

Moreover, as the desired geometries consist of toroidal surfaces, a toroidal coordinate system is defined. The term toroidal will refer to the long way around the torus, while poloidal refers to the direction the short way around the torus. These directions can be found in Figure 2.5

2.3.2 Magnetic field lines and flux surfaces

The magnetic field is mathematically described as a vector field. The flow lines of this vector field are called the magnetic field lines and are often used for visualization and interpretation of physical phenomena. The movement of charged particles is related to the geometry of these magnetic field lines as described in Section 2.2. In that section, it was shown that particles can drift across field lines along their motion parallel to the field lines. As the movement parallel to the field line is not disturbed, the temperature tends to equilibrate along a field line.

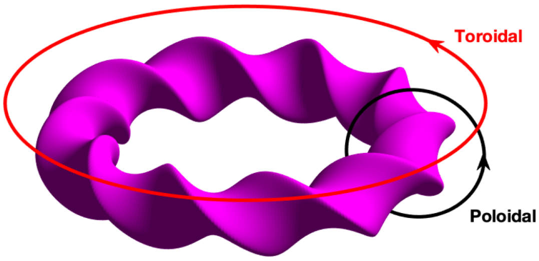


Figure 2.5: The position on a toroidal surface is often described by two angles. A poloidal angle increases from 0 to 2π on any closed poloidal loop (black), the short way around the torus. A toroidal angle increases from 0 to 2π on any closed toroidal loop (red), the long way around the torus. Adapted from [9].

In the magnetic confinement fusion devices discussed above, it is necessary to maintain a hot plasma core while the walls need to remain cold. Therefore, field lines are not allowed to connect the plasma core with the walls. If field lines are not allowed to intersect the walls, they should stay inside the toroidal volume. This can be accomplished if a field line can lie on a closed surface within the volume. A flux surface is a smooth surface such that at every point on the surface $\boldsymbol{B} \cdot \hat{n} = 0$, where $\hat{n}$ is a normal vector to the surface. So, in particular, no magnetic field line crosses a magnetic surface: the field is tangent to the flux surface. There may exist a set of toroidal surfaces within a given volume. All of these surfaces may be nested around a single closed field line, called a magnetic axis. As already stated, a twist in the field lines is needed for confinement resulting in field lines which twist about flux surfaces. The twist in the field lines is quantified by the rotational transform, ι, which indicates the number of poloidal turns of a field line around the magnetic axis for each toroidal turn.
To maintain a temperature gradient within the confinement region, magnetic fields that minimize the volume occupied by island structures need to be designed, as the temperature is equilibrated rapidly within these structures. Ideally, the magnetic field lies on continuously nested surfaces. The flux surfaces and magnetic islands are shown with three Poincaré plots of Wendelstein 7-X in Figure 2.6.

Equilibrium condition

As stated above due to the fast movement along the field lines, the ions and electrons are in near-Maxwellian distribution, which implies that the plasma has the pressure of an ideal gas namely $p = nT$, where $n = n_e + n_i$ is the sum of the number of electrons and the number of ions and T is the temperature. The force of this pressure is balanced by the electromagnetic force. Therefore, the equilibrium condition states

$$\boldsymbol{\nabla} p = \boldsymbol{J} \times \boldsymbol{B} \tag{2.8}$$

This has a few implications namely that the magnetic field and the current are perpendicular to the pressure gradient $\boldsymbol{\nabla} p$. From this, it can be derived that $\boldsymbol{B} \cdot \boldsymbol{\nabla} p = 0$ and this implies that the magnetic field lies on surfaces of constant plasma pressure. These surfaces are the flux surfaces mentioned earlier.

This can be used to extend the toroidal coordinate system described earlier. In that example, only points on a surface could be explained. Using the nested flux surfaces, points in space can be described. This results in the flux coordinates (s, θ, ϕ), where ϕ is an angle which increases by 2π upon a toroidal loop, θ is an angle which increases by 2π upon a poloidal loop, and s is a flux surface label.

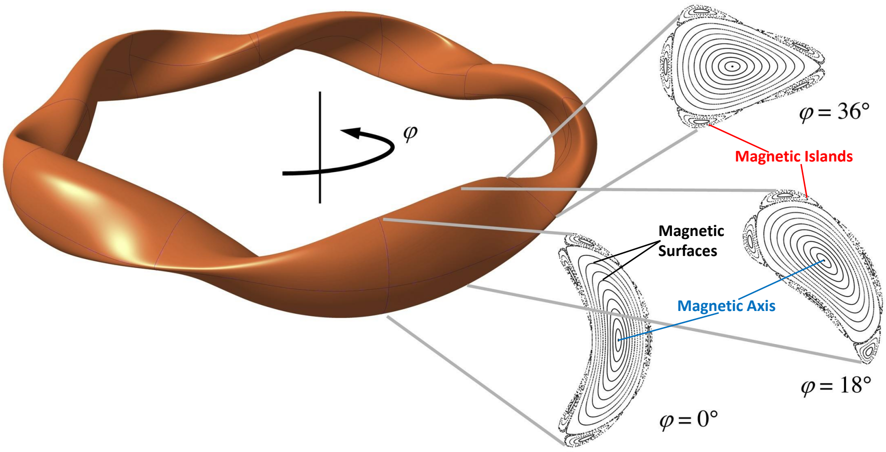


Figure 2.6: Shape of the Last Closed Flux Surface (LCFS) in W7-X standard magnetic configuration. The explanation of this configuration can be found in Appendix B.1. In this configuration, $\iota/(2\pi) = 1 = 5/5$ at the plasma edge, and the separatrix of five natural magnetic islands is therefore defining the LCFS. This can be seen in the Poincaré plots shown on the right for the bean-shaped ($\varphi = 0°$), the $\varphi = 18°$ and the triangular ($\varphi = 36°$) cross section. Adapted from [10]

2.3.3 Boozer coordinates

To describe the equilibrium of nested surfaces it is useful to introduce magnetic coordinates, where one coordinate is constant on the constant-pressure surfaces and the field lines are straight in terms of the other coordinates[11]. The representation of $\boldsymbol{B}$ in magnetic coordinates is given by

$$\boldsymbol{B} = \boldsymbol{\nabla}\psi \times \boldsymbol{\nabla}\theta + \boldsymbol{\nabla}\phi \times \boldsymbol{\nabla}\chi \tag{2.9}$$

where ψ and χ are the toroidal and poloidal flux, and θ and ϕ are the poloidal and toroidal direction. Since the toroidal and poloidal magnetic fluxes are constant on surfaces of constant pressure, these surfaces are called flux surfaces. Functions that are constant on such surfaces, only depend on ψ and are independent of θ and ϕ, are called flux functions. Now the magnetic field can be written as

$$\boldsymbol{B} = I\boldsymbol{\nabla}\theta + G\boldsymbol{\nabla}\phi + K\boldsymbol{\nabla}\psi + \boldsymbol{\nabla}H \tag{2.10}$$

where $I(\psi)$ and $G(\psi)$ are flux functions representing the poloidal and toroidal current, K is a function of(ψ, θ, ϕ) and $H(\psi, \theta, \phi)$ is an integration constant.

One could choose different angles ϕ and θ for the magnetic coordinates, Boozer choose a useful set of magnetic coordinates[12]. This coordinate system is reached if $\theta = \theta' + \iota\omega$ and $\phi = \phi' + \omega$ where $\omega(\psi, \theta, \phi)$ is a well behaved and periodic in the poloidal and toroidal directions.

This results in an elegant covariant representation of the magnetic field

$$\boldsymbol{B} = I(\psi)\boldsymbol{\nabla}\theta + G(\psi)\boldsymbol{\nabla}\varphi + K(\psi, \theta, \varphi)\boldsymbol{\nabla}\psi \tag{2.11}$$

where the primes have been dropped for readability.

Besides the simple covariant representation in Boozer coordinates, the magnetic field lines are mapped to be straight lines once plotted in a θ, ϕ plane. For a more detailed description of Boozer coordinates and general coordinate systems, the appendix of the paper ”Physics of magnetically confined plasmas”[13] by A. Boozer is recommended.

These Boozer coordinates are especially useful for stellarators, as quasi-symmetric magnetic fields exhibit a symmetry in this coordinate system. Therefore, confinement properties can be inferred by simply considering the symmetry of the field strength. Using Boozer coordinates also makes it easier to distinguish magnetic field properties. This is visible in Figure 2.7 where the contours of $|B|$ from the The Helically Symmetric Experiment(HSX) are plotted in Boozer coordinates. It can be seen that the minima and maxima of the magnetic field close in on themselves when moving helically around the torus. This is a property of quasi-helical symmetry which reduces the transport as was explained above. A clear distinction can be made when looking at a θ, ζ plot from W7-X which is designed following the quasi-omnigenous principle. Compared with the magnetic field of HSX, the magnetic field strength of W7-X does not connect with itself moving around the torus as can be seen in Figure 2.8. Note that this plot only shows $1/5^{\text{th}}$ of the entire magnetic field due to the five-fold symmetry while the plot of HSX shows the magnetic field of one full toroidal rotation.

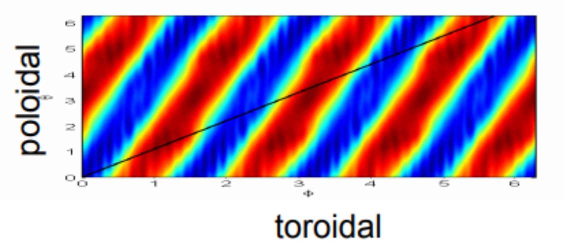


Figure 2.7: Contours of $|B|$ for a quasi-helically symmetric configuration in HSX where red is maximum field and blue is minimum field. The magnetic field is shown for one full toroidal rotation. A magnetic field line is shown in black with $\iota = 1.06$. Source: [14]

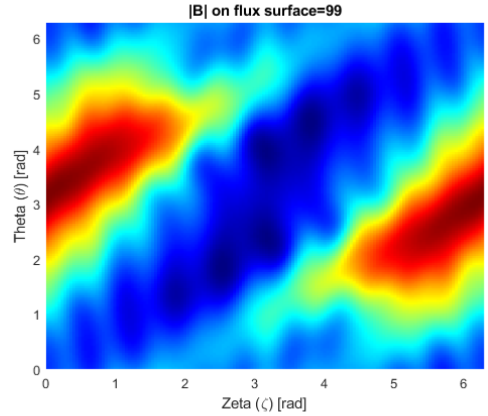


Figure 2.8: Contours of $|B|$ on the outermost flux surface for the standard configuration in W7-X. The magnetic field is shown for only $1/5^{th}$ of a toroidal rotation because of the five-fold symmetry in W7-X.

Fourier transform into coefficients

The use of Boozer coordinates and the needed periodicity in toroidal and poloidal direction allows the representation of quantities on a given radial surface in terms of trigonometric function (sine and cosine). This allows quantities in 3D to be represented by continuous functions in the poloidal and toroidal direction and a discrete set of points in the radial direction. Thus on any radial surface, the value of a parameter is known to machine accuracy at any point. In Boozer coordinates, any quantity f can be represented by

$$f(s,\theta,\zeta) = \sum_{n=-N}^{N}\sum_{m=0}^{M} f_{mn}(s)\cos(m\theta + nN_{FP}\zeta) \tag{2.12}$$

or

$$f(s,\theta,\zeta) = \sum_{n=-N}^{N}\sum_{m=0}^{M} f_{mn}(s)\sin(m\theta + nN_{FP}\zeta) \tag{2.13}$$

where s is a radial flux surface label, θ is the poloidal angle, ζ is the toroidal angle, and N_{FP} is the fundamental periodicity of the problem. The values N and M define the truncation of the mode spectrum.

The choice of trigonometric function for a given quantity is determined by the symmetry of the problem. For systems with stellarator symmetry (up-down in the phi=0 plane), the cylindrical radial coordinate (R) has an even symmetry (cosine) while the vertical coordinate (Z) has an odd symmetry (sine). These coordinates are visualized in Figure 2.4. In general, toroidal coordinates do not require this symmetry and quantities are functions of a series of both odd and even coefficients.

Code used: Variational Moments Equilibrium Code (VMEC)[15]

The code uses a variational method to find a minimum in the total energy of the system. The total plasma potential energy can be found by combining the MHD force balance equations. These are the two equations used to solve the equilibrium condition above, combined with $\boldsymbol{\nabla} \times \boldsymbol{B} = \mu_0 \boldsymbol{j}$ resulting in

$$W = \int \left(\frac{\|\boldsymbol{B}\|^2}{2\mu_o} + \frac{p}{\gamma - 1} \right) d^3x \tag{2.14}$$

to solve this, the code assumes that quantities may be Fourier expanded in terms of the poloidal and toroidal coordinates resulting in [16]

$$R(s,\theta,\zeta) = \sum_{n=-N}^{N}\sum_{m=0}^{M} R_{n,m}(s)\cos\left(m\theta - nN_{FP}\zeta\right) \tag{2.15}$$

$$Z(s,\theta,\zeta) = \sum_{n=-N}^{N}\sum_{m=0}^{M} Z_{n,m}(s)\sin\left(m\theta - nN_{FP}\zeta\right) \tag{2.16}$$

The magnetic field on these flux surfaces is defined in the same manner as

$$B(s,\theta,\zeta) = \sum_{n=-N}^{N}\sum_{m=0}^{M} B_{n,m}(s)\cos\left(m\theta - nN_{FP}\zeta\right) \tag{2.17}$$

Due to its speed in computing the MHD equilibrium in 3D, this code has become the standard code and practically all stellarator researchers use it.

2.4 Transport in fusion plasma

2.4.1 Diffusion

When microscopic particles randomly move in small steps the particle flux $\mathbf{\Gamma}$ is defined by Fick's law, stating that flux is proportional to a concentration gradient

$$\mathbf{\Gamma} = -D\boldsymbol{\nabla}n \tag{2.18}$$

where D is a diffusion coefficient. Since the change in time of density at a point in space is proportional to the divergence of the particle flux at that point,

$$\frac{\partial n}{\partial t} = -\boldsymbol{\nabla} \cdot \mathbf{\Gamma} \tag{2.19}$$

which is a continuity equation for particles. Putting these two equations together results in the diffusion equation for the evolution of particle density

$$\frac{\partial n}{\partial t} = D\boldsymbol{\nabla}^2 n \tag{2.20}$$

To understand the definition of the transport coefficients the intuitive derivation is used instead of the perturbation of the distribution function caused by gradients. This intuitive derivation is based on the random walk approach. For a simple 1-D walk this results in a Gaussian shape around the starting position. Diffusion can be interpreted as the sum of many random walks of step size Δr every time interval Δt,

$$D \sim \frac{(\Delta r)^2}{\Delta t} \tag{2.21}$$

This formula will be the pillar of this section and the only question that remains is, what needs to be filled in for the step size Δr and the time step Δt. As will become clear later on, for different types of transport, different step sizes and time steps need to be chosen. Fick's law is the chemical equivalent of Fourier's law, stating the heat flux through a material is proportional to the negative local temperature gradient as

$$\boldsymbol{q} = -k\boldsymbol{\nabla}T \tag{2.22}$$

where $\mathbf{q}$ is the local heat flux density, k is the material's conductivity, and T is the temperature.

2.4.2 Classical transport

Classical transport concerns itself with the diffusion of particles in a straight, magnetized cylinder due to collisions. When particles collide with other particles, their guiding center position changes. To determine the change in the guiding center, only electron-ion collisions need to be taken into account. In collisions between the same particles (electron-electron or ion-ion collisions), the guiding centers of the two particles move in opposite directions. Therefore this does not result in a net diffusion. Furthermore, as the direction of these particles are not truly random, the evolution of the density does not follow Equation 2.18.

Taking only unlike particle collisions into account, the diffusion coefficient can be determined by taking the average change in the guiding center after each collision as a step size (Δr) and the average time between collisions as the time step (Δt). In a straight cylinder, the time between collisions is approximately given by the electron-ion collision time which is $1/\nu_{ei}$ where

$$\nu_{ei} = \frac{4\pi n e^4}{m_e^2 V_{Te}^3} \ln \Lambda \tag{2.23}$$

in which $\Lambda = \frac{4\pi}{3}n\lambda_D^3$ is the number of particles in a Debye sphere and $V_{Te} = \sqrt{\frac{2k_BT_e}{m_e}}$ is the thermal velocity of an electron. The change in the guiding center position between collisions Δr is approximately the electron gyro-radius, ρ_e, which is given in Equation 2.4. Using these estimates for the step size and the time scale the classical diffusion coefficient is given by

$$D_{\text{class}} = \frac{(\Delta r)^2}{\Delta t} = \rho_e^2\nu_{ei} \tag{2.24}$$

2.4.3 Neoclassical transport

So far the transport model has been unrealistic. Due to budget limitations, an infinitely long cylinder is not a feasible plasma reactor. Therefore, in the neoclassical transport model, the toroidal magnetic field geometry is taken into account. In toroidal field geometry, the particle trajectories can be split into two groups, passing particles and trapped particles which need to be discussed first.

Passing and trapped particles

Orbits can be classified into two main categories, passing and trapped. A particle is considered a passing particle when the parallel velocity is large enough to follow the magnetic field lines. These passing particles are not considered to be a problem. On the other hand, if the parallel velocity is not high enough a particle is trapped. As can be seen in Figure 2.9 the B-field of a tokamak is one big well where a particle might get trapped if the parallel velocity is not high enough, in a tokamak this is called the banana orbit. Besides the big well, a stellarator has smaller helical wells where particles might get trapped, these particles are called helically trapped particles. The magnetic field strength plotted in Figure 2.9 is a simplified version of real stellarator fields. In practice, not all helical wells have the same depth. The criterion for trapped particles can be found in Equation 2.25. In contrast to the passing particles, the vertical drift of the helically trapped particles is not compensated which means these particles can escape the confinement region. The difference between passing and helically trapped particles in LHD is visualized in Figure 2.10.

$$E_{kin} < E_{magnetic} => \frac{v_{||}^2}{v^2} < 1 - \frac{B_{min}}{B_{max}} = \frac{dB}{B} = \varepsilon_h \tag{2.25}$$

where ε_h is the depth of the helical wells determined by the coil configuration. Because not all wells have the same depth, ε_h is a combined term of all separate wells.
Due to the complex magnetic configuration, different types of trapped particles are possible. Particles can be trapped but orbit a full toroidal turn between reflections. These particles have some characteristics of banana orbits from the tokamak but their reflection will happen at different positions due to the lack of axisymmetry. Lastly, helically trapped particles can also be trapped in the toroidal ripple via the poloidal drift component. These particles have a long duration of stay in the region of reflection which leads to a large radial displacement. Therefore these orbits are called super banana orbits.

After this classification of the different orbit types, the neoclassical transport can be analyzed for all these types of orbits, starting with passing particles. The transport resulting from passing particles is also referred to as Pfirsch-Schlüter transport.

A passing particle does stay on the flux surface on average, but locally it might deviate from it. This displacement of the guiding center, δ_p, from the flux surface is what is used as the step size Δr in Equation 2.21. If the electron-ion collision is longer than the transit time $\tau_{ei} > \tau_{tr}$ the particle can make a full orbit. This makes the stepping size $\delta_p = \rho_{L\theta}\varepsilon$ where the poloidal Lamor radius is given by $\rho_{L\theta} = \frac{mv}{qB_\theta}$. This results in a total Pfirsch-Schlüter diffusion constant

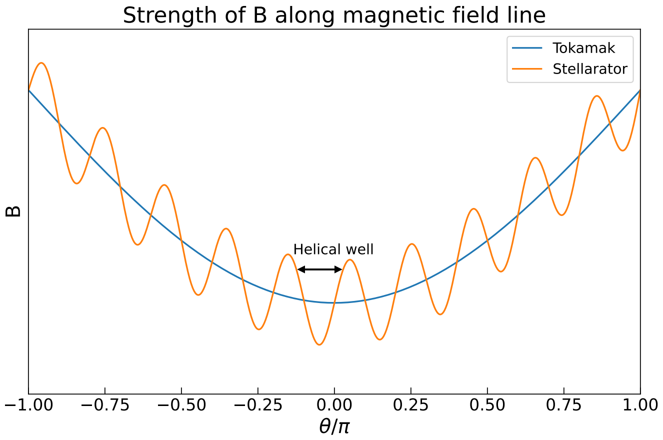


Figure 2.9: The variation in B, the magnitude of the magnetic field, along a field line of a theoretical Tokamak compared to a theoretical Stellarator. In practice, not all helical wells have the same depth.

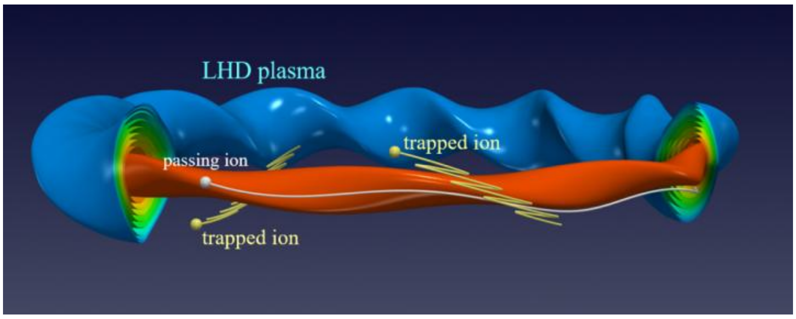


Figure 2.10: The passing ion (white sphere) moves in one direction. On the other hand, the trapped ion (yellow sphere) moves back and forth and the center of the back-and-forth motion also moves simultaneously in a helical direction, which is a characteristic of the LHD. The plasma pressure is constant on each colored surface, and the plasma pressure is high in the central region. Courtesy of NIFS

$$D_{PS} = \frac{1}{2\tau_{ei}} \left(\frac{2mv}{qB\iota} \right)^2 = D_{cl} \cdot \frac{1}{(\iota/2\pi)^2} \tag{2.26}$$

In stellarators, helically trapped particles are one of the biggest problems. If the collision time of a helically trapped particle is long enough for the particle to fulfill its trapped movement it will be lost due to the vertical drift. This process is not a diffusion process. Luckily, if $\tau_{eff} < \tau_{loss}$ the particles can be scattered back into passing trajectories. This allows the step size to be written as $\delta_p = v_{drift}\,\tau_{eff}$ with the effective collision time $\tau_{eff} = \tau_{ei}\,\varepsilon_h$. With these assumptions about the collision times, the diffusion coefficient can be written as

$$D = \sqrt{\varepsilon_h} \frac{(v_{drift}\tau_{eff})^2}{2\tau_{eff}} = \frac{\varepsilon_h^{3/2}}{2} v_d^2 \tau_{ei} \tag{2.27}$$

where the square root of the helical ripple is added to account for the trapped particles. It can easily be seen that this diffusion coefficient scales with τ_{ei} or $1/\nu$. This is why this diffusion coefficient is also referred to as the $1/\nu$ regime and $D_{1/\nu}$. Furthermore, if all variables are filled in, it shows that $D_{1/\nu} \propto T_e^{7/2}$ which is a lot worse than the corresponding transport coefficients in tokamaks and thus a problem for hot fusion plasmas in stellarators. As this transport is not ambipolar, a radial electric field is formed. This radial electric field causes $\boldsymbol{E} \times \boldsymbol{B}$ drifts which counteract the transport losses. A possible scenario might be that the ions are in the '$1/\nu$ regime' while the electrons are in another regime with lower losses, i.e. the Pfirsch-Schlüter regime, due to different collisionality and temperature. This would result in large losses of the ions and small losses of electrons. Due to this, the plasma charges up negatively which is called the ion root scenario, $E_r < 0$. If it is the other way around it is called the electron root, $E_r > 0$.

This radial electric field changes the T_e dependence of the diffusion coefficient drastically by changing the displacement due to the drift velocity. The poloidal transit frequency, in this case, is $\omega_{E\times B} = E_r/rB$ which will dominate the distance a particle can travel. This changes the step size to be $\Delta r = v_{drift}/\omega_{E\times B}$. Furthermore, the collision frequency needs to be adjusted for the number of trapped particles and becomes $\nu_{eff} \sim \nu/f^2$ where f is the fraction of participating particles. The step size is in general given by $\Delta t = v_{eff}^{-1}$ which would result in a diffusion coefficient of

$$D \sim f\frac{\Delta r^2}{\Delta t} \sim \frac{\nu v_{drift}^2}{f\omega_{E\times B}^2} \tag{2.28}$$

which diverges for $f \to 0$. However, the assumption was that the frequency responsible for stopping the particle from drifting and therefore the dominant one was that associated with the electric field. This creates an upper limit for the effective collision frequency $\nu_{eff} \leq \omega_{E\times B}$. With this reason $f \sim \sqrt{\nu/\omega_{E\times B}}$, leading to a final diffusion coefficient in a low collisionality regime of

$$D \sim \frac{\nu^{1/2} v_{drift}^2}{\omega_{E\times B}^{3/2}} \tag{2.29}$$

In this regime, the transport thus scales with $\sqrt{\nu}$ and decreases with an increasing electric field due to $E_r^{-3/2}$[17]. The scaling of the transport with the different levels of drift velocity is shown in Figure 2.11 by Beidler[18]. The different curves show the diffusion coefficient for different radial electric fields, where lower curves have a stronger electric field.

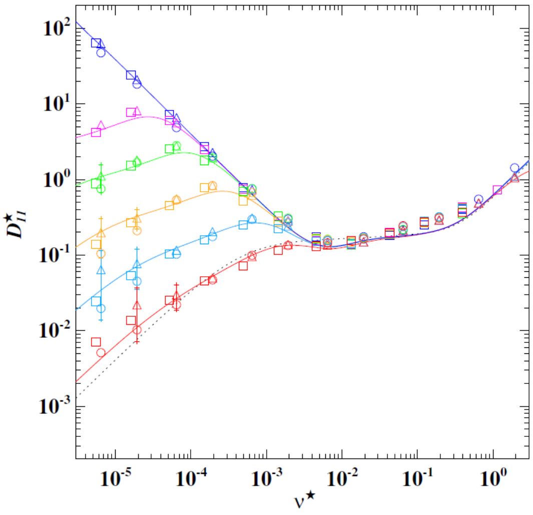


Figure 2.11: Diffusion coefficient as a function of collisionality for different values of the normalized drift velocity 1×10^{-3}(red), 3×10^{-4}(light blue), 1×10^{-4}(orange), 3×10^{-5}(green), 1×10^{-5}(purple) and zero (dark blue) for the standard configuration of W7-X at $\rho = 0.25$. Results for a comparable tokamak are given by the dotted line. Source: [18]

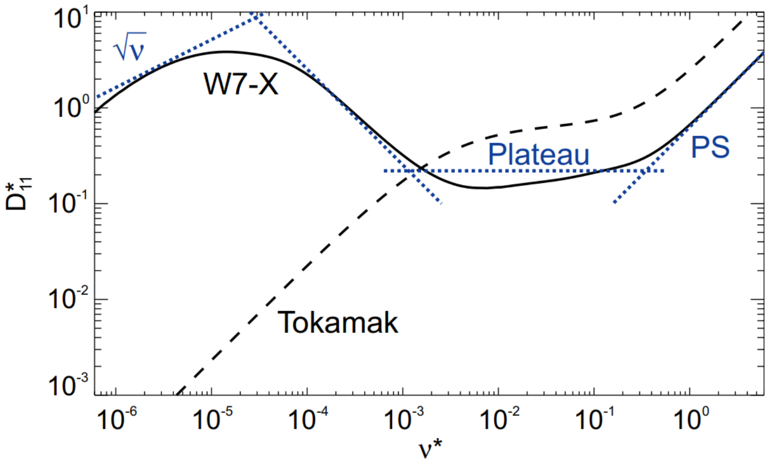


Figure 2.12: A diffusion coefficient versus collisionality for tokamaks (dotted curve) and the W7-X stellarator (solid curve). The asymptotic regimes are indicated by dotted straight lines. In order of increasing collisionality: the $\sqrt{\nu}$-regime, the $1/\nu$-regime, the plateau regime, and the Pfirsch–Schlüter regime. At very low collisionality (below the range shown) the transport again becomes proportional to ν. Source: [17]

All these coefficients can be combined to create a 'transport versus collisionality landscape' which is shown in Figure 2.12. As can be seen in that figure, the diffusion coefficient in the stellarator is much higher in the low collisionality regime. This predicted large energy losses in most stellarators even in the absence of plasma turbulence. Fortunately, these losses can be reduced by optimizing the geometry of the magnetic field[19]. One of those optimizations are geometries that are quasi-symmetric[20]. Quasi-symmetry refers to configurations for which the magnetic field strength, B, only varies on a surface through a given linear combination of Boozer angles. This implies that there is a symmetry coordinate on which the field strength does not depend.

While quasi-symmetry implies particle confinement in the absence of collisions or fluctuations, it is not a necessary condition to obtain confinement in a stellarator. Another more general property is omnigeneity, meaning that the time-averaged magnetic drift of a magnetic surface vanishes for all particles. A fully omnigenous configuration means the magnetic field is only dependent on the flux surface label. Following Helander[19], this is impossible to achieve for all radii. Luckily, a device does not have to be completely omnigenous to be optimized. The Wendelstein 7-X stellarator is a machine which has been optimised to be quasi-isodynamic (or also "quasi-omnigeneous") with consequently low neoclassical transport[21].

2.4.4 Drift-kinetic equation

The random walk approach used above gives an order-of-magnitude approximation of the neoclassical transport, but to calculate it more accurately, one needs to solve the so-called drift kinetic equation. The neoclassical transport is related to the fast motion along the field line and the slower drifts perpendicular to it. An anisotropy of the distribution function in gyro angle is related to the classical transport (which as seen earlier can be described as a random walk with the step length of the gyro radius). For the purpose of describing neoclassical transport, it is thus assumed that the gyro-frequency is much larger than the collision frequency, and the Fokker-Planck equation

$$\frac{\partial f_a}{\partial t} + \boldsymbol{v} \cdot \boldsymbol{\nabla} f_a + \frac{e_a}{m_a}(\boldsymbol{E} + \boldsymbol{v} \times \boldsymbol{B}) \cdot \frac{\partial f_a}{\partial \boldsymbol{v}} = C_a\left(f_a\right) \tag{2.30}$$

will therefore be averaged over the gyro-phase. In deriving the drift kinetic equation for a magnetized plasma, a small gyro-radius ordering is assumed, $\delta = \rho/L \ll 1$, where ρ is the gyro radius and L a scale length of the gradients of background parameters (magnetic field,density,temperature etc.). Another assumption is that there are no fluctuations $\partial/\partial t \sim \delta^2 v_T/L$ where the thermal velocity $v_T = (2T/m)^{1/2}$, and consistently with this that the $E \times B$ velocity is small $E/B = \mathcal{O}(\delta v_T)$ The resulting drift kinetic equation is given by

$$\frac{\partial f_a}{\partial t} + \left(v_\| \boldsymbol{b} + v_{\mathrm{d}a}\right) \cdot \nabla f_a = C_a\left(f_a\right) \tag{2.31}$$

Code used: Drift Kinetics Equation Solver (DKES)[22, 23] and Neotransp[24]

DKES solves the drift kinetic equation for a given radius, collisionality and normalised radial electric field, producing normalised versions of the mono-energetic transport coefficients described by Beidler[18]. These calculations are very computationally intensive, which is why a database is saved for only some combinations of the inputs.

Then, Neotransp is used to interpolate between the values produced by DKES to a value it needs to perform the Maxwellian-weighted integrals for L_{ij} given by Beidler[18]. These L_{ij} are then used to calculate the radial fluxes, given the gradients of density and temperature and the radial electric field.

Chapter 3

Background machine learning

3.1 Generative modeling

As mentioned in Chapter 1, this project aims to implement a generative model. Machine learning models can be classified into two types of models, discriminative and generative models. First, a brief recap of the more familiar discriminative model is given. The training of discriminative machine learning models is done by learning parameters that maximize the conditional probability $\mathbb{P}(\mathbf{y} \mid \mathbf{x})$ given a set of data instances $\mathbf{x}$ and a set of labels $\mathbf{y}$. This method focuses the model on predicting labels of the data.

A generative model on the other hand captures the joint probability $\mathbb{P}(\mathbf{x}, \mathbf{y})$, or just $\mathbb{P}(\mathbf{x})$ if there are no labels of the training set. This represents a powerful model, as this allows for several applications not feasible for discriminative models. Generative models can generate realistic data by sampling from the joint probability. Furthermore, because probability distributions are used, rather than point-wise predictions, the uncertainty of the predictions can be obtained. This is an important feature because it allows for uncertainty quantification.

Two well-known versions of powerful generative models are the Variational Autoencoder(VAE)[25] and the Generative Adversarial Network (GAN)[26] both introduced in 2014. Both models use a so-called latent space which is a multidimensional space spanned by a multidimensional latent variable.
to me this paragraph is very abstract, also it doesnt become clear why P(y—x) in the discriminative model isn't also a distribution function, but P(x) is, so I think you need to add that generative models also choose to use probability distributions

Latent variables

In a latent variable model, complex observable variables $\mathbf{x}$ are assumed to be represented by a lower-dimensional latent variable $\mathbf{z}$. As the number of latent dimensions is smaller than the number of complex observable variables, $\mathbf{z}$ is used to compress data. The models in this project are considered successful if they can reconstruct the original data from the compressed latent variables as will be explained in the next section. Therefore, it must learn to store only relevant information and disregard the noise as there is no room in the compressed space. This is one benefit of compressing into latent variables. It gets rid of any extraneous information, and only focuses on the most important features[1]. This 'compressed state' is the latent space of our data.

[1] This is generally true, but does not always hold. If there are enough latent dimensions compared to the data complexity the noise can be captured as well.

3.2 Variational autoencoder

In this work, a generative network based on the Variational Autoencoder(VAE) is implemented. The VAE is a probabilistic version of the classic autoencoder. We first explain the autoencoder framework, after which we explain the variational autoencoder.

3.2.1 Autoencoder

An autoencoder[27] is a deep learning model that learns to reconstruct data through a compressed representation, i.e. a bottleneck. To achieve this, it uses two connected neural networks called the encoder and decoder. The task of the encoder is to take an input $\mathbf{x}$ and encode it into a point $\mathbf{z}$ in latent space. Data points that are similar in the original data set do not have to placed at similar positions in the latent space.

Then, $\mathbf{z}$ is used as input by the decoder. The decoder operates as an opposite of the encoder as its output will be a reconstruction $\hat{\mathbf{x}}$ of the initial input $\mathbf{x}$. During training the reconstruction error between $\mathbf{x}$ and its reconstruction $\hat{\mathbf{x}}$ is minimized. Minimizing this error requires optimization of both the encoder and the decoder. A schematic overview of an autoencoder is given in Figure 3.1.

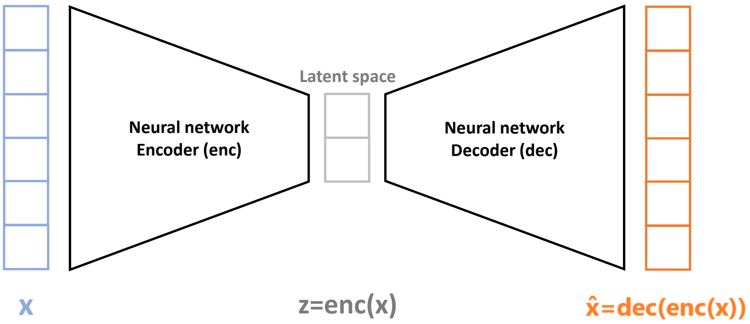


Figure 3.1: A schematic overview of an autoencoder. The encoder and decoder are neural networks. The input data $\mathbf{x}$ is shown in blue, the latent space $\mathbf{z}$ in grey and the output $\hat{\mathbf{x}}$ in orange. Underneath the reconstruction of input $\mathbf{x}$ through the model is shown. Adapted from Joseph Rocca[28].

The question that remains is how this model can generate content. The data has been encoded into points in latent space and the decoder is trained to reconstruct this data with precision. The meaning of the space between two points in latent space is not known and there is no regularization of this space. It is almost impossible to ensure that the encoder will organize the latent space in a way compatible with a meaningful generative process. To deal with this problem, the autoencoder was upgraded to a variational autoencoder.

3.2.2 VAE

A VAE is similar to the autoencoder described above. Instead of working with a point in latent space, each dimension of the latent variable represents a probability distribution. The probability distributions to be represented are chosen to be normal distributions. These distributions are parameterized by the encoder which provides a mean μ and standard deviation σ for the given input data $\mathbf{x}$. By applying the loss function described in the next section to this, the true posterior $p(\mathbf{z} \mid \mathbf{x})$ is approximated by an approximate posterior $q_\phi(\mathbf{z} \mid \mathbf{x})$. This approximate posterior is a neural network with trainable parameters ϕ. Afterward, a realization of the distribution is made by sampling, i.e. $\mathbf{z} \sim \mathcal{N}(\mu, \sigma)$ where μ and σ are taken from the encoder. This sample is then used by the decoder to make a reconstruction $\hat{\mathbf{x}}$ of the original input $\mathbf{x}$ as $\hat{\mathbf{x}} = p_\theta(\mathbf{x} \mid \mathbf{z})$ with θ the parameters of the decoder neural network. This is schematically visualized in Figure 3.2.

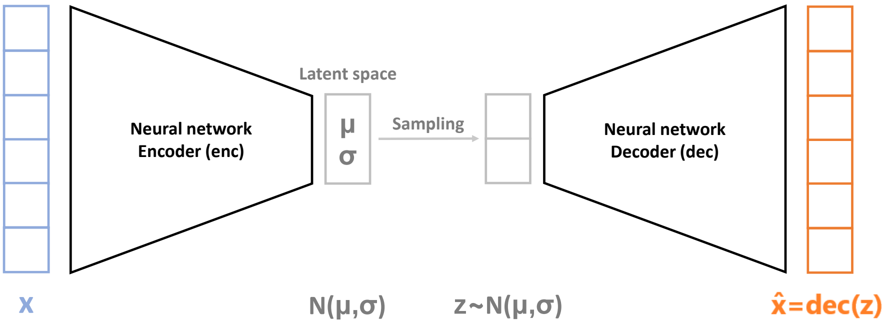


Figure 3.2: A schematic overview of a variational autoencoder. The encoder and decoder are neural networks. The input data $\mathbf{x}$ is shown in blue, the latent space $\mathbf{z}$ in grey and the output $\hat{\mathbf{x}}$ in orange. Underneath the reconstruction of input $\mathbf{x}$ through the model is shown. Adapted from Joseph Rocca[28].

The latent space is regularized by imposing a prior over the latent distribution $p(\mathbf{z})$, which is typically chosen to be a multidimensional Gaussian $\mathcal{N}(0, 1)$. This is taken into account in the loss function which is discussed in Section 3.2.3. Because of this imposed prior, the approximate distribution is somewhat known. This allows the model to be used in a generative fashion. By taking a sample from the imposed prior and running it through the decoder, a new data point can be created.

3.2.3 Loss function

There are many different forms of loss functions, but the standard loss function of a VAE consists of two main terms as

$$\mathcal{L}_{VAE}(\mathbf{x}) = d(\mathbf{x}, \hat{\mathbf{x}}) + D_{KL}(q_\phi(\mathbf{z} \mid \mathbf{x}) \,||\, p(\mathbf{z})) \tag{3.1}$$

where the first term is the reconstruction term, the same as in the plain autoencoder, a distance measure d between the original input and the reconstruction. Minimizing this term is required to produce good reconstructions of the original data. The second term is to restrict the latent space to a chosen prior distribution and is called the regularisation term. Usually, the KL-divergence[29, 30] between the prior and the approximate posterior found by the model is chosen as a regularisation term. This KL-divergence is a way of measuring the matching between two distributions given by

$$D_{KL}(p \,||\, q) = \sum_x p(x) \log\left(\frac{p(x)}{q(x)}\right) \tag{3.2}$$

The regularisation term prevents the model to encode data far apart in the latent space and pushes the approximate posterior towards the prior. On the other hand, the reconstruction term tries to separate the data to minimize reconstruction errors. This regularization term thus counteracts the reconstruction term, resulting in a higher reconstruction error on the training data. The trade-off between these two terms can be adjusted and have a great impact on the performance of the model[31]. Therefore, balancing these two terms is an important step in designing the model which will be explained in Section 3.2.4.

In a VAE, the chosen prior is often a standard Gaussian ($\mathcal{N}(0, 1)$). If this is the case, the KL-divergence in Equation 3.1 can be written as

$$D_{KL}(q_\phi(\mathbf{z} \mid \mathbf{x}) \,||\, \mathcal{N}(0, 1)) = \frac{\sigma_q^2 + \mu_q^2}{2} - \frac{1}{2} - \log(\sigma_q) \tag{3.3}$$

where μ_q is the standard deviation and σ_q the mean of the approximate posterior in one dimension. For a multidimensional latent space, a summation of KL-divergences over all dimensions is required. The derivation of Equation 3.3 can be found in Appendix A.

Reparameterization

Usually, the latent distribution output of the encoder is a normal distribution with mean μ and standard deviation σ: $\mathbf{z} \sim \mathcal{N}(\mu, \sigma^2)$. Taking a sample from this distribution to reconstruct the input introduces randomness in our model. During backpropagation, gradients are computed through the model to update all the weights. Updating the gradients through the randomness introduced by the sampling is however impossible.

This problem is solved by applying the reparameterization trick[32]. The reparameterization trick rewrites the representation of $\mathbf{z}$ as a random variable into the following

$$\mathbf{z} \sim \mathcal{N}(\mu, \sigma^2) \rightarrow \mathbf{z} = \mu + \sigma \odot \epsilon, \text{ where } \epsilon \in \mathcal{N}(0, 1) \tag{3.4}$$

In the reparameterized version of $\mathbf{z}$, ϵ is the only stochastic thing, μ and σ are not stochastic anymore. This allows for backpropagation through μ and σ which is what we want.

3.2.4 Annealing

In the section above, the KL-divergence and the reconstruction error both have the same weight in the total loss function. One can play with the weight of the KL-divergence as a design choice, which is better known as the β-VAE [33]. In this case, the loss function of Equation 3.1 is adapted to be

$$\mathcal{L}_{VAE}(\mathbf{x}) = d(\mathbf{x}, \hat{\mathbf{x}}) + \beta\, D_{KL}(q_\phi(\mathbf{z} \mid \mathbf{x}) \,||\, p(\mathbf{z})). \tag{3.5}$$

In the original β-VAE paper, a β value of 250 is used, compared to $\beta = 1$ in the original VAE. Using high values of β can lead to an increased disentanglement of the latent space[34, 35]. Using lower values of β on the other hand, is a known solution to a well-known machine learning issue, the posterior collapse.

Posterior collapse

The posterior collapse[36, 37] in VAEs arises when the variational distribution closely matches the uninformative prior for a subset of latent variables. If the posterior is not collapsed, $\mathbf{z}_k$ (k is the dimension of the variable) is sampled from a distribution $q_\phi(\mathbf{z}_k|\boldsymbol{x}) = \mathcal{N}(\mu_k, \sigma_k^2)$ where μ_k and σ_k are stable functions of the input $\mathbf{x}$. Therefore the encoder distills useful information from the data $\mathbf{x}$ into μ_k and σ_k.

A posterior is considered to be collapsed when the signal from the data input $\mathbf{x}$ to posterior parameters is too noisy or too weak. This results in a decoder that starts ignoring $\mathbf{z}$ samples drawn from the posterior $q_\phi(\mathbf{z}|\mathbf{x})$ [38].

If the signal is too noisy it results in unstable μ_k and σ_k and thus the sampled $\mathbf{z}$'s are also unstable, which forces the decoder to ignore them. This can better be described by the output of the decoder $\hat{\mathbf{x}}$ becoming almost independent of the sampled $\mathbf{z}$, which produces generic outputs $\hat{\mathbf{x}}$ that are crude representatives of all seen $\mathbf{x}$'s.

A signal which is too weak translates to a posterior

$$q_\phi(\mathbf{z} \mid \mathbf{x}) \simeq q_\phi(\mathbf{z}) = \mathcal{N}(a, b) \tag{3.6}$$

where μ and σ of posterior become almost disconnected from input $\mathbf{x}$. This means μ and σ collapse to constant values a and b channeling a weak signal from different inputs to the decoder. As a result, the decoder tries to construct $\hat{\mathbf{x}}$ by ignoring useless $\mathbf{z}$'s which are sampled from $\mathcal{N}(a, b)$.

Besides looking at the structure of the data, the posterior collapse can also occur due to the design of the model. If the decoder is sufficiently powerful, it needs less informative dimensions to solve the training objective[39].

Annealing schedules

There are different ways of handling the β coefficient. To clarify, when $\beta = 1$ the model acts as the plain VAE, and for $\beta = 0$ the model turns into an autoencoder. The value for β does not have to remain the same throughout training. Adjusting the value of β during training is called annealing.

A common annealing schedule is the Monotonic Annealing Schedule[40]. It sets $\beta = 0$ at the beginning of training and gradually increases β until at a given epoch[2] $\beta = 1$ is reached. This schedule has become the standard for training VAEs on texts and has been adopted in many Natural Language Processing[41] tasks.

A new annealing schedule proposed by Fu et al. is the Cyclical Annealing schedule[42]. In this schedule, the β coefficient starts at 0 and β is then increased to $\beta = 1$ at a fast pace where it will remain for some epochs. This encourages the model to converge towards the VAE objective and infers its first raw latent distribution. The optimization is then continued at $\beta = 0$ again, which perturbs the VAE objective and dislodges it from the convergence which allows the model to use the entire latent space. This process is repeated several times to achieve better convergences. Formally, β has the following form:

$$\beta_t = \begin{cases} f(\tau), & \tau \leq R \\ 1, \tau & \tau > R \end{cases} \text{ with } \quad \tau = \frac{\operatorname{mod}(t-1, \lceil T/M \rceil)}{T/M} \tag{3.7}$$

where t is the iteration number, T is the total number of training iterations, f is a monotonically increasing function, M is the number of cycles, and R is the fraction of a cycle used to increase β.

An example of a monotonic annealing schedule and a cyclical annealing schedule are shown respectively in Figure 3.3(a) and Figure 3.3(b).

[2] An epoch in machine learning means one complete pass of the training dataset through the algorithm.

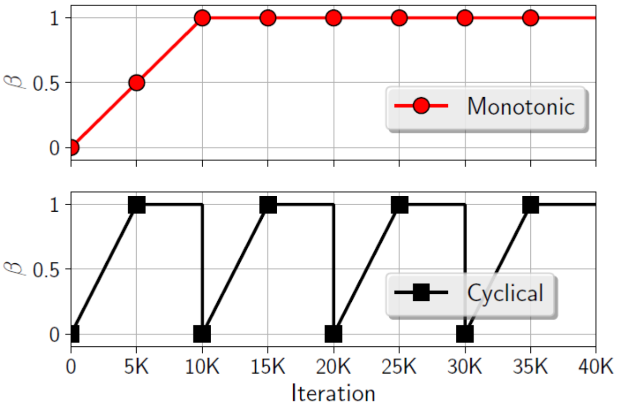


Figure 3.3: Comparison between (a) traditional monotonic and (b) proposed cyclical annealing schedules. In this figure, $M = 4$ cycles are illustrated, $R = 0.5$ is used for increasing within each cycle. Adapted from [42]

3.3 Conditional variational autoencoder

The VAE has the possibility of generating new data by taking a sample from the latent space. However, this sample will be an arbitrary sample from the data set, and thus there is no control over what part of the dataset is sampled. This lack of control limits the potential of VAEs.

To gain some control over the generative process the VAE can be extended to a Conditional Variational Autoencoder(CVAE)[43]. The CVAE allows extra input conditions besides the input data and the latent space. This leaves the latent space to contain all relevant information about the data that is unrelated to the conditions, while the generated samples are forced to have properties described by the conditions. A schematic overview of a conditional variational autoencoder is given in Figure 3.5. As can be seen, new samples can be generated for some given condition **y** by supplying the decoder with both **y** and a sample from the prior of the latent space.

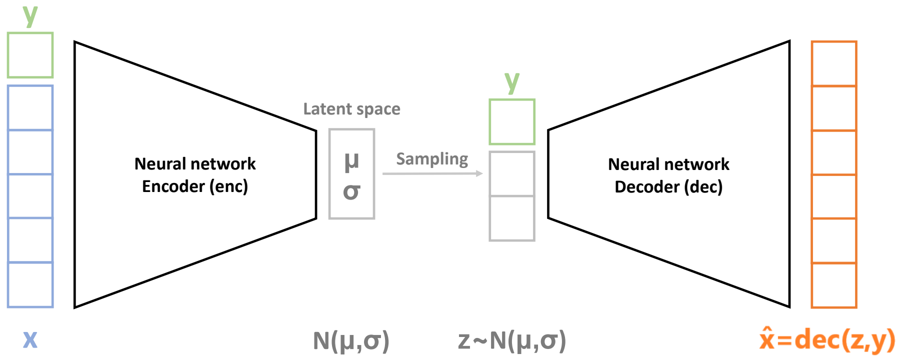


Figure 3.4: A schematic overview of a conditional variational autoencoder. The encoder and decoder are neural networks. The input data **x** is shown in blue, the latent space **z** in grey, the conditions **y** in green and the output $\hat{\mathbf{x}}$ in orange. Underneath the reconstruction of input **x** through the model is shown.

3.4 Domain Invariant Variational Autoencoder

The Domain Invariant Variational Autoencoder (DIVA)[44] is another adaptation of the normal VAE. The idea behind DIVAs is to split the latent space into separate sub-spaces. These sub-spaces are split according to the respective ”domains” by using a conditional prior based on the domain and using an auxiliary regression network to re-predict the domain information from the latent representation. This would allow for a separation of the correlations based on the plasma parameters or the magnetic field properties in our case. Later, this separation can be exploited to dive into these latent spaces individually to better understand the learned behavior.

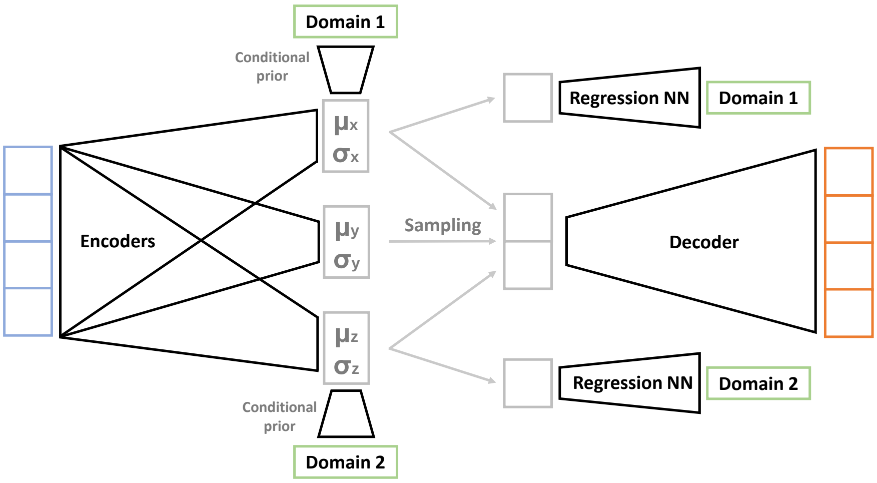


Figure 3.5: A schematic overview of a Domain Invariant Variational Autoencoder. The input data $\mathbf{x}$ is shown in blue, the latent space $\mathbf{z}$ in grey and the output $\hat{\mathbf{x}}$ in orange. The domains used for the latent spaces are shown in green and are used for the conditional priors. The encoders, decoder, conditional priors, and the regression NN are all neural networks.

Chapter 4

Related Work

Over the past few years, the field of nuclear fusion has been increasingly incorporating machine learning methods to analyze the growing amount of data. The newest and most prominent use of deep learning shows autonomous control of the coils in a tokamak using reinforcement learning[45]. They show the ability to produce different plasma configurations on the Tokamak à Configuration Variable(TCV). Besides this, deep learning models have been used to speed up predictions of tokamak core profiles[46, 47]. More closely related to this work, machine learning has been used to solve VMEC equilibrium evaluations[48] where a 6 order of magnitude runtime reduction is achieved.

Besides the aforementioned deep learning applications, the model used in this work is not commonly used in the field of nuclear fusion. One application uses a VAE to predict disruptions in a tokamak using seven input which are then mapped into a 2-dimensional latent space from which a disruptivity score can be computed[49]. If the disruption score is too high, a control sequence can be launched to stop the disruption and the model can identify the cause. They, however use time signals which is not comparable to this project. Another application of a VAE in the field of disruption prevention is by Ferreira et al.[50]. They, however, use a VAE for anomaly detection, to find patterns in the data leading to plasma disruptions. This application makes sense because a VAE can give an estimate of the likelihood of the data i.e. producing a low likelihood for anomalous patterns[51]. This results in a high VAE loss, which can then be coupled to an anomaly score indicating unusual samples. This property of a VAE has been exploited at CERN[52, 53] as well. Here the anomaly would indicate a collision showing behavior not incorporated in the current models for particle physics, and thus a potentially interesting sample. Although the cited works use the same modeling method, the goals of these works do not line up with the goal of this project. These articles make very little use of the generative capabilities of this modeling method, which in this project is an important property of the VAE.

The generative property of a VAE has been exploited in some instances in the physics domain. The problem of complex many-body physics is one of the fields where VAEs have proven their worth[54, 55] as the computational costs scale exponentially with the size of the system for numerical simulations. Besides the VAE, generative adversarial networks(GANs)[56] are often used when generative capabilities are required. These GANs are widely used in the field of particle physics[57, 58, 59] but also in modeling non-fusion transport phenomena[60], fluid simulations[61], and complex turbulence models[62, 63]. Some of these applications use physics-informed loss functions to satisfy boundary conditions. However, in this work, the GAN is not applicable due to two reasons. Firstly, it is known that GANs are harder to train due to their generator-discriminator structure[64]. The second and most important reason is that the interpretability of the GAN is low[65] and VAEs have both an inference and generative model. As these concepts are essential in this project, the VAE has been picked as the better choice.

Lastly, as mentioned above, this research touches on the subject of latent space exploration. The exploration of latent space is not yet a big field of research, but some progress has been made. Up until now, a large number of projects on latent space exploration have been done using data containing faces[66], other objects[67, 68], or physics-related images[69]. This makes sense as a shift in latent space can easily be visualized by comparing the two images. Some projects are trying latent space exploration on non image data sets and with success[70]. David Winant and his colleagues successfully explored a non VAE latent space representation of ECG signals. Besides the interpolation itself, work is being done on how interpolation in latent space needs to be performed[71, 72].

To the best of my knowledge, the work described in this thesis is the first application of generative models aimed at generating new magnetic field configurations in nuclear fusion machines. There are, like already mentioned, deep learning applications focused on computing MHD equilibria[48] and on controlling the magnetic field[45] but these applications do not serve the same purpose as this work where generative models are used stand-alone, in a purely data-driven approach, to construct new configurations.

Chapter 5

Data

5.1 Simulating transport

The model is trained on a data set constructed by simulating neoclassical transport. This is done using the Neotransp simulation code of Håkan Smith from the Max Planck Institute for Plasma Physics. His code uses a database calculated by DKES and interpolates this to all given input parameters as explained in Section 2.4.4. The transport variables which are calculated by Neotransp are limited to the particle flux densities Γ_e, Γ_i, the heat flux Q_e, Q_i, the energy flux S_e, S_i, and the neoclassical diffusion coefficient D_e, D_i for both the electrons and the ions. The input parameters for this code are the magnetic geometry and plasma parameters: densities n_e and n_i, temperatures T_e and T_i, the logarithmic density gradients and the logarithmic temperature gradients. The logarithmic gradients $1/L_n$ and $1/L_T$ are defined by dividing the gradient by its value as $\boldsymbol{\nabla} n/n$ and $\boldsymbol{\nabla} T/T$ respectively. In this case, the simulations are done as simple as possible, with only electrons, Hydrogen ions, and no impurities. Furthermore, all parameters are set to be equal for electrons and ions.

To generate the full data set, a parameter sweep is performed per flux surface for multiple geometry configurations. The ranges used for the parameter sweep can be found in Table 5.1. This range is based on a reconstruction of an experiment in W7-X[73].

Variable	**Min**	**Max**
Densities $n_{e,i}$	$0.1 \times 10^{20}\,\mathrm{m}^{-3}$	$1.5 \times 10^{20}\,\mathrm{m}^{-3}$
Temperatures $T_{e,i}$	$0.1\,\mathrm{keV}$	$5\,\mathrm{keV}$
Logarithmic density gradients $1/L_n$	$-0.05\,\mathrm{m}^{-1}$	$-6\,\mathrm{m}^{-1}$
Logarithmic temperature gradients $1/L_T$	$-0.1\,\mathrm{m}^{-1}$	$-20\,\mathrm{m}^{-1}$

Table 5.1: The range of all input plasma parameters used in Neotransp

This range is then divided into 10 linear steps between the lowest and highest values resulting in 10^4 possible combinations per flux surface. The number of flux surfaces used in a simulation is based on the number of nested flux surfaces specified in a geometry file. Computation of transport quantities on these flux surfaces in Neotransp is a local process. Therefore, gradients of a variable do not have to line up with the value of that variable on the next flux surface. While calculating the transport quantities, both the electron and ion root solutions are accepted if the radial electric field was between $E_r = -18\,\mathrm{kV\,m^{-1}}$ and $E_r = 7\,\mathrm{kV\,m^{-1}}$.

These simulations are performed on the national supercomputer Snellius using a small NWO computation grant. All output is stored in .Zarr format to reduce the amount of storage needed.

5.2 Preprocessing

After all simulations are completed, the data needs to be preprocessed. The computed transport quantities are combined with the according plasma parameters and the Fourier components of the flux surface belonging to that specific computation. Taking all available Fourier components would lead to an unnecessary amount of extra variables and the higher Fourier components do not contribute a meaningful amount to the total magnetic field or flux surface. Therefore, only the the Fourier components of $m \leq 11$ and $|n| \leq 12$ are taken into account.

Due to the big range in plasma input parameters, some combinations of plasma parameters at specific flux surfaces did not result in transport quantities. This might happen if Neotransp can not find a radial electric field determined by the ambipolarity constraint within the allowed range. As the radial electric field is a necessary to determine the transport, these combinations of plasma parameters result in empty outputs and need to be filtered out of the data set. Other specific input combinations result in negative particle flux densities. As these are only a few instances ($< 0.05\%$) and the reason for their occurrence could not be identified, these instances are filtered out for practical reasons. Besides the negative flux density cases, there are some instances in which the heat flux of ions and electrons is negative while the particle flux is positive. In these instances the energy flux is positive but as the heat flux is the energy flux minus heat convection, the heat flux can still be negative. These instances are also taken out to simplify the data set to overcome the posterior collapse as will be explained later in Section 7.2.4. This results in four constraints on the data set $\Gamma_{e,i} > 0$ and $Q_{e,i} > 0$

Also, some configurations result in extremely large maximum flux values. It was found that these configurations have a high value for the effective helical ripple ε_h compared to the other configurations. Following Equation 2.27, this high effective ripple explains the higher fluxes. The configurations are described by Joachim Geiger's three letter codes: [UEM, PKM, XAM, XHM, SIN, THE, UFM, PIS, NIT, SMM, UMM, VUM, TGM, WLT, YXT, SBM, UMM, RLM, QMF]. The first letter of the code represents the mirror ratio B_{01}/B_{00} on the axis, the second is the iota at the edge and the third is the difference between the planar coil currents leading to inward/outward shift. The full explanation of these three letters codes can be found in Appendix B.1.

Afterward, the data undergoes scaling to make sure it can be handled by the model. The distribution within the data differs greatly per variable. For example, the input parameters are given in linear steps while the range in for example flux is a lot wider. The input parameters can therefore easily be scaled to the 0-1 range by simple transformations. On the contrary, all transport variables are highly skewed as can be seen in Figure 5.1a. The skewed data of variable $\mathbf{x}$ is scaled to the 0-1 range by first taking the $\log_{10}(\mathbf{x})$ and then scale the outcomes linearly to the range of $\min(\log_{10}(\mathbf{x}))$, $\max(\log_{10}(\mathbf{x}))$. This results in the scaling from Figure 5.1a to Figure 5.1b. It can be seen that all flux variables follow the same trend which makes sense as the heat flux $Q_{e,i}$ and the energy flux $S_{e,i}$ are dependent on the particle flux $\Gamma_{e,i}$. Lastly, The Fourier components are scaled linearly to the min and max of that specific component over all geometries and all flux surfaces. The min and max of every variable in the total data set can be found in Table 1 in Appendix B.2. Further details on the implementation of the prepocessing steps, file formats and the generator functions can be found in Appendix B.3

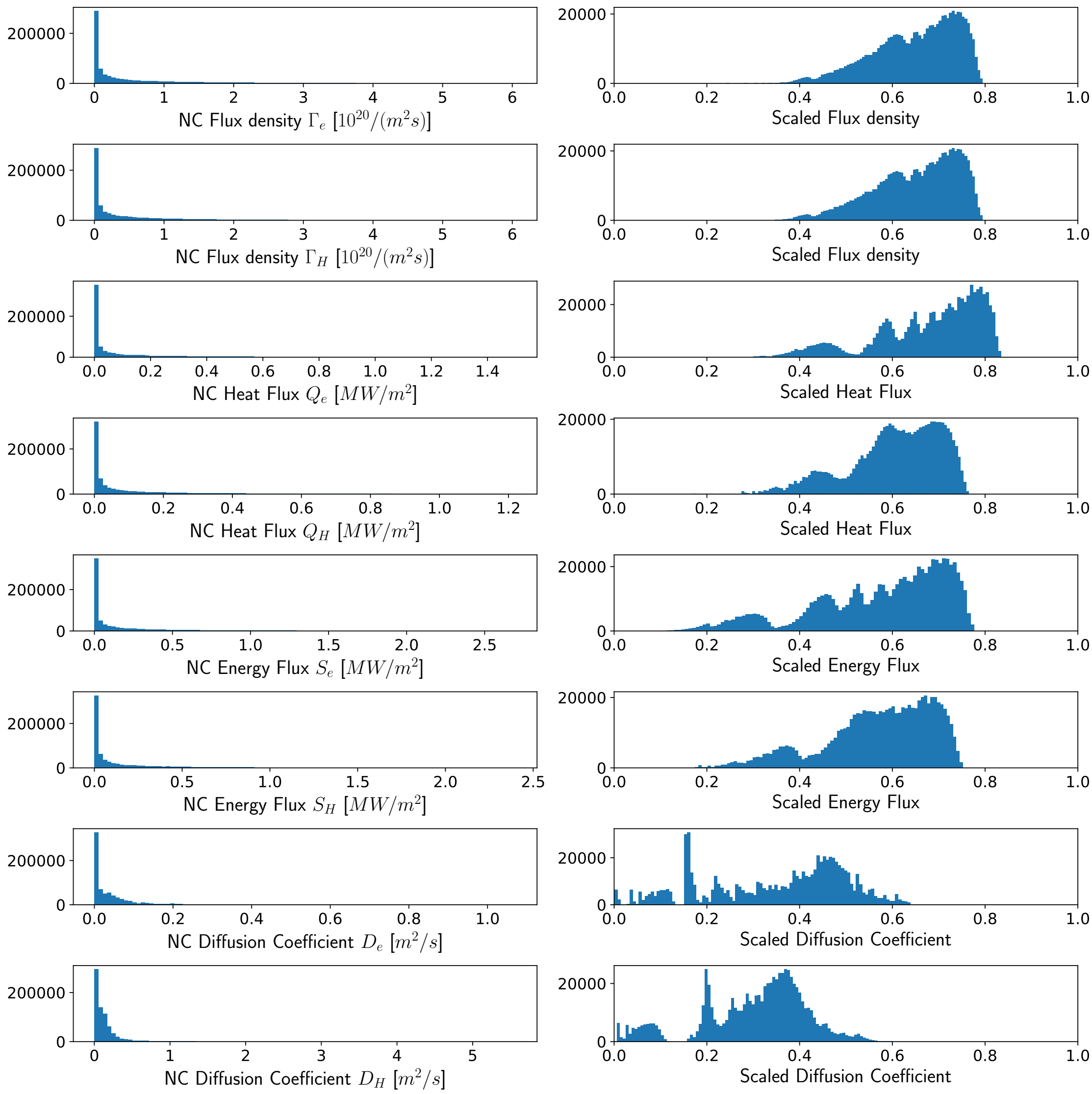


Figure 5.1: Eight different neoclassical transport variables computed using Neotransp for geometry reference 17, plotted in a histogram with 100 bins. The left column shows the absolute values and the right shows the values after applying a $\log_{10}$ and linear scaling.

Chapter 6

Methodology

6.1 Model designs

In this project, three different models have been designed. These will be referred to as **Model 1.0**, **Model 1.1** and **Model 2**. The models described in this chapter are the general versions of the models used in this project. While in Chapter 7, different variants of these models will be used to find results and shortcomings.

Model 1.0

The first model is based on the VAE framework. The geometry data $\mathbf{G}$, the transport quantities $\mathbf{X}$, and the plasma parameters $\mathbf{P}$ are all encoded into one latent space $\mathbf{z}$. One encoder q_ϕ approximates the posterior for $\mathbf{z}$ with trainable parameters denoted as ϕ after which the values for $\mathbf{z}$ are sampled from the approximate distribution. The prior $p(\mathbf{z})$ is set to a Gaussian distribution.

To generate new data, we sample from the prior distribution of $\mathbf{z}$. This sample results in a prediction for the plasma parameters $\hat{\mathbf{P}}$, the geometry $\hat{\mathbf{G}}$ and the transport $\hat{\mathbf{X}}$ using the decoder neural network $\widehat{\mathbf{pgx}}_{\theta_x}(\mathbf{z})$ with trainable parameters θ_x. A schematic overview of Model 1.0 can be found in Figure 6.1.

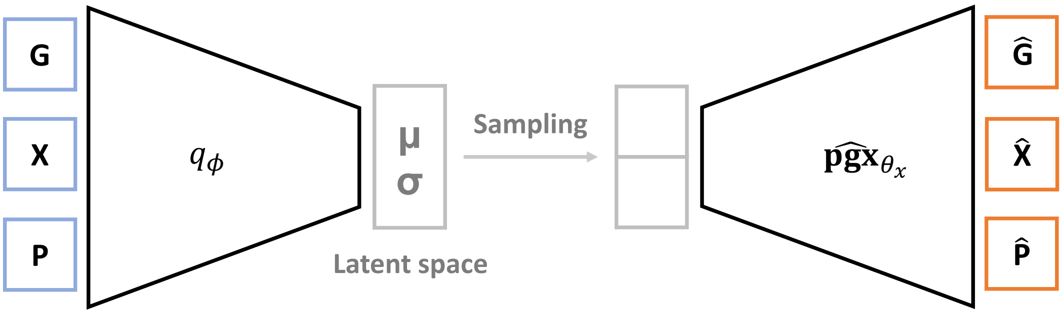


Figure 6.1: A schematic overview of Model 1.0 based on the VAE framework. The input consist of 3 categories, the Geometry data $\mathbf{G}$, the transport quantities $\mathbf{X}$ and the plasma parameters $\mathbf{P}$. The encoder and decoder are given by q_ϕ and $\widehat{\mathbf{pgx}}_{\theta_x}(\mathbf{z})$ respectively.

Model 1.1

For Model 1.1, Model 1.0 has been adjusted to contain 2 latent spaces. One of these latent spaces, $\mathbf{z_p}$, is used to encode only the plasma parameters $\mathbf{P}$. The other latent space, $\mathbf{z_g}$, is used to encode both the Fourier components of the geometry $\mathbf{G}$, and the transport quantities $\mathbf{X}$. This has been done to make sure the big influence of the plasma parameters would not overrule the entire geometry latent space as will be explained in Chapter 7. In the encoding process, two neural networks, q_{ϕ_p} and q_{ϕ_g} approximate the posteriors using trainable parameters ϕ_p and ϕ_g respectively. The values for $\mathbf{z_p}$ and $\mathbf{z_g}$ are then sampled from the approximate distributions.

The generative process of Model 1.1 is similar to the that of Model 1.0. The main difference is that two standard Gaussian priors $p(\mathbf{z_p})$ and $p(\mathbf{z_g})$ are sampled to find a prediction of the plasma parameters $\hat{\mathbf{P}}$, the geometry $\hat{\mathbf{G}}$ and the transport $\hat{\mathbf{X}}$ instead of one. The prediction are calculated by the same decoder neural network. A schematic overview of Model 1.1 can be found in Figure 6.2.

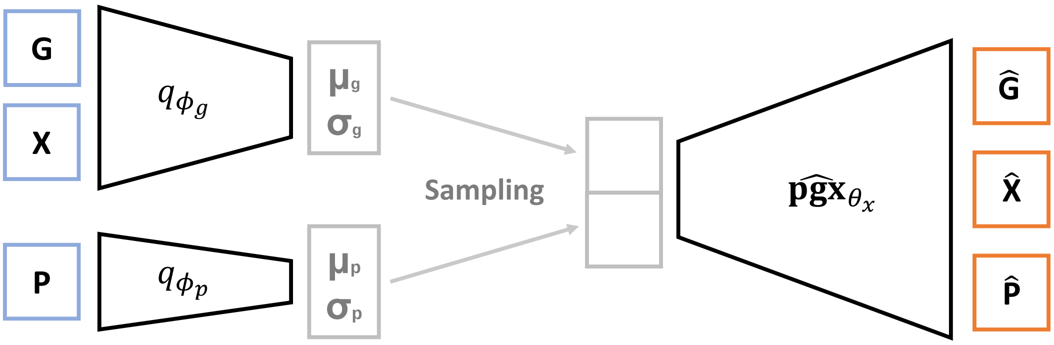


Figure 6.2: A schematic overview of Model 1.1. The input consist of 3 categories, the Geometry data $\mathbf{G}$, the transport quantities $\mathbf{X}$ and the plasma parameters $\mathbf{P}$. The encoders are given by by q_{ϕ_p} and q_{ϕ_g}. These encode the data into two separate latent spaces. The decoder is given by $\widehat{\mathbf{pgx}}_{\theta_x}(\mathbf{z})$.

Model 2

The second model is more sophisticated than the first model and is based on the DIVA framework. The second model uses two similar encoders $q_{\phi_{x1}}$ and $q_{\phi_{x2}}$ to encode all transport quantities, $\mathbf{X}$, into two latent spaces $\mathbf{z_{xp}}$ and $\mathbf{z_{xg}}$ with trainable parameters ϕ_{x1} and ϕ_{x2} respectively. The values for $\mathbf{z_{xp}}$ and $\mathbf{z_{xg}}$ are then sampled from these approximate posteriors. The geometry $\mathbf{G}$ and plasma parameters $\mathbf{P}$, function as a condition to shape the according latent spaces.

In the generative process, the conditional priors $p_{\theta_p}(\mathbf{z_{xp}}|\mathbf{p})$ and $p_{\theta_g}(\mathbf{z_{xg}}|\mathbf{g})$ can be sampled and concatenated to get a reconstruction of the transport quantities by running them through a decoder network $\hat{\mathbf{x}}_{\theta_x}(\mathbf{z_{xp}}, \mathbf{z_{xg}})$. This also works the other way around; By sampling from the approximate distributions and running the sample through the regression networks $\hat{\mathbf{p}}_{\theta_p}(\mathbf{z_{xp}})$ and $\hat{\mathbf{g}}_{\theta_g}(\mathbf{z_{xg}})$, a prediction of the plasma parameters and the geometry can be constructed for the sampled transport values. A schematic overview of Model is shown in Figure 6.3

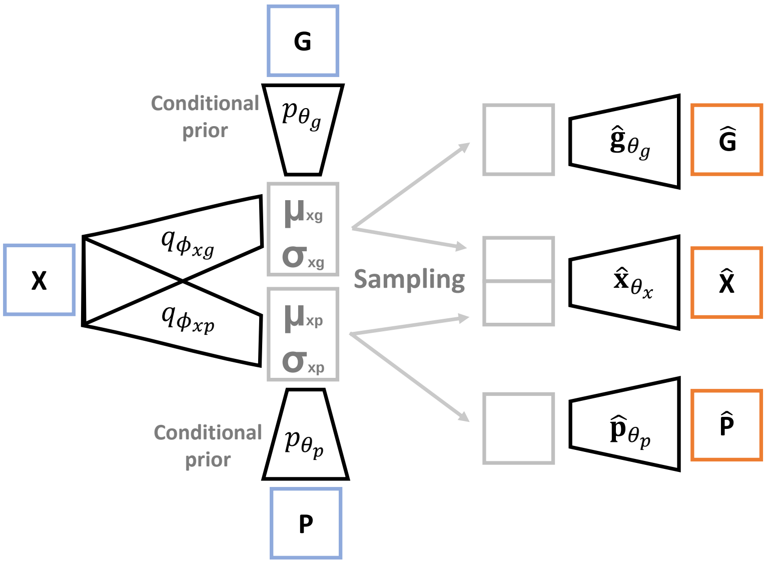


Figure 6.3: A schematic overview of Model 2 based on the DIVA framework. The input consist of 3 categories, the Geometry data $\mathbf{G}$, the transport quantities $\mathbf{X}$ and the plasma parameters $\mathbf{P}$. Two encoders $q_{\phi_{x1}}$ and $q_{\phi_{x2}}$ encode $\mathbf{X}$ into separate latent spaces. Two conditional priors p_{θ_p} and p_{θ_g} are used to shape the respective latent spaces given input $\mathbf{P}$ and $\mathbf{G}$. Three regression networks $\hat{\mathbf{g}}_{\theta_g}$, $\hat{\mathbf{x}}_{\theta_x}$ and $\hat{\mathbf{p}}_{\theta_p}$ can be used to construct predictions from a sample.

General setup

The neural networks making up all models are implemented using Tensorflow/Keras[74]. In general, both models follow a scheme of fully connected layers with the Rectified Linear Unit (ReLU) activation[75]. These fully connected layers are alternated by Batch normalization[76] layers. A batch normalization layer applies a transformation that maintains the mean output close to 0 and the output standard deviation close to 1. This has a stabilizing effect on the learning process and reduces the number of required training epochs.

During training, the Batch normalization layer transforms the data by returning

$$\mathbf{x}_{out} = \frac{\gamma(\mathbf{x}_{batch} - \overline{\mathbf{x}_{batch}})}{\sqrt{\sigma^2_{batch} + \epsilon}} + \beta \tag{6.1}$$

where $\mathbf{x}_{batch}$ is the input, $\overline{\mathbf{x}_{batch}}$ is the average of that batch, γ is a learned scaling factor, ϵ is a small constant and β is a learned offset factor. During inference, however, the layer normalizes the output using a moving average of the mean and the standard deviation of the batches it has seen during training. Further details on the implementation and architecture of all models can be found in Appendix C.

6.2 Model training and loss functions

The first models are trained in an unsupervised manner using no conditions. The second model on the other hand is trained in a supervised fashion allowing the use of conditional priors. The loss function is in general based on reconstruction losses, using the mean squared error between the reconstruction and the original data, and the KL-divergence as regularization terms.

Model 1.0

The loss for the first unsupervised Model 1.0 is composed of three components and is given in Equation 6.2

$$
\begin{aligned}
\mathcal{L}_1(\mathbf{p}, \mathbf{g}, \mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{p},\mathbf{g},\mathbf{x})}[\mathrm{mse}(\mathbf{p}, \mathbf{g}, \mathbf{x},\ \widehat{\mathbf{pgx}}_{\theta_x}(\mathbf{z}))] && \text{(6.2a)} \\
+\, \beta\, D_{KL}(q_{\phi_p}(\mathbf{z} \mid \mathbf{p}, \mathbf{g}, \mathbf{x}) \| \, p(\mathbf{z})) && \text{(6.2b)}
\end{aligned}
$$

where Monte Carlo sampling is used for the expectations and the reparametrization trick is used for the latent variables $\mathbf{z}$. The term given in 6.2a is the reconstruction loss, where the mean squared error is taken between the original input data and the reconstruction that is given by the decoder. Term 6.2bis a regularization term using the KL-Divergence with a standard Gaussian prior as given by Equation 3.3. This regularization is needed to allow for sampling of the latent space, when the goal is to generate new data. The regularization term is weighted by a factor β following the Cyclic annealing schedule given in Section 3.2.4.

Model 1.1

The loss for the second unsupervised Model 1.1 is composed of three components and is given in Equation 6.3

$$
\begin{aligned}
\mathcal{L}_1(\mathbf{p}, \mathbf{g}, \mathbf{x}) = \mathbb{E}_{q_{\phi_p}(\mathbf{z_p}|\mathbf{p}), q_{\phi_g}(\mathbf{z_g}|\mathbf{g},\mathbf{x})}[\mathrm{mse}(\mathbf{p}, \mathbf{g}, \mathbf{x},\ \widehat{\mathbf{pgx}}_{\theta_x}(\mathbf{z_p}, \mathbf{z_g}))] && \text{(6.3a)} \\
+\, \beta\, D_{KL}(q_{\phi_p}(\mathbf{z_p} \mid \mathbf{p}) \| \, p(\mathbf{z_p})) && \text{(6.3b)} \\
+\, \beta\, D_{KL}(q_{\phi_g}(\mathbf{z_g} \mid \mathbf{g}, \mathbf{x}) \| \, p(\mathbf{z_g})) && \text{(6.3c)}
\end{aligned}
$$

where again Monte Carlo sampling and the reparametrization trick are used for both latent variables $\mathbf{z_p}$ and $\mathbf{z_g}$. The term given in 6.3a is the reconstruction loss, which now depends on multiple encoders and latent spaces compared to Model 1.0. Terms 6.3b and 6.3c are the regularization terms using the KL-Divergence with a standard Gaussian prior. Both regularization term are weighted by β following the Cyclic annealing schedule.

Model 2

The loss for the supervised model is composed of five components and is given in Equation 6.4.

$$
\begin{aligned}
\mathcal{L}_2(\mathbf{x}) = \mathbb{E}_{q_{\phi_{xp}}(\mathbf{z_{xp}}|\mathbf{x}), q_{\phi_{xg}}(\mathbf{z_{xg}}|\mathbf{x})}[\mathrm{mse}(\mathbf{x}, \hat{\mathbf{x}}_{\theta_x}(\mathbf{z_{xp}}, \mathbf{z_{xg}})] && \text{(6.4a)} \\
+\, \mathbb{E}_{q_{\phi_{xp}}(\mathbf{z_{xp}}|\mathbf{x})}[\mathrm{mse}(\mathbf{p}, \hat{\mathbf{p}}_{\theta_p}(\mathbf{z_{xp}})] && \text{(6.4b)} \\
+\, \mathbb{E}_{q_{\phi_{xg}}(\mathbf{z_{xg}}|\mathbf{x})}[\mathrm{mse}(\mathbf{g}, \hat{\mathbf{g}}_{\theta_g}(\mathbf{z_{xg}})] && \text{(6.4c)} \\
+\, \beta\, \gamma_1\, D_{KL}(q_{\phi_{xp}}(\mathbf{z_{xp}} \mid \mathbf{x}) \| \, p_{\theta_p}(\mathbf{z_{xp}} \mid \mathbf{p})) && \text{(6.4d)} \\
+\, \beta\, \gamma_2\, D_{KL}(q_{\phi_{xg}}(\mathbf{z_{xg}} \mid \mathbf{x}) \| \, p_{\theta_g}(\mathbf{z_{xg}} \mid \mathbf{g})) && \text{(6.4e)}
\end{aligned}
$$

where again Monte Carlo sampling and the reparametrization trick are used for the latent variables. The first terms (6.4a - 6.4c) are the reconstruction losses for the transport values, the plasma parameters, and the geometry variables, respectively. These reconstruction terms all use the mean squared error. The last terms (6.4d) and (6.4a) are the KL-divergence with a conditional prior. These terms are important for both the encoders into the latent spaces and the conditional prior itself. Again, the regularization terms are weighted by β following a short Cyclic annealing schedule. These terms are also weighted by two different γ values to force the model to use both latent spaces and also to steer away from a posterior collapse.

6.2.1 Model comparison

Model 1.1 is the final result of an ongoing trial and error process starting from Model 1.0, where much has been learned. Model 2 is however carefully designed taking the faults of Model 1 into account. Therefore, Model 2 possesses some advantages over the first models that need to be highlighted.

First of all, Model 1.1 can not function as a simulator because the transport values are entangled with the geometry. Therefore, to give meaningful distributions in latent space, a geometry must always be accompanied by transport values, which is the part to be simulated. In the second model, a prior can be determined based on the geometry alone without the transport values, creating the possibility for a simulator. The task of simulating new data is important as a sanity check for the model which will be explained in the next section.

Furthermore, the latent spaces in the second model are shaped by the conditional priors. This would theoretically give the model a better chance to find relations between the transport values and the plasma parameters or geometry components compared to simply running them through an encoder together.

6.3 Model evaluation

During training the model uses the loss functions described above to evaluate the performance of the model. In general, the loss function needs to be minimized for the optimal result. However, in our case, these metrics do not fully represent the functioning of the model. These metrics can be minimized by ungeneralized models which is not useful in our case. A way to determine if the model is actually capable of capturing the essence of the data is to use the model as a simulator.

Using the conditional prior in Model 2, one is able to simulate transport values for given geometry and plasma parameters. The conditional priors over the plasma parameters and the geometry produce a distribution over the latent space variables for these given input parameters. The mean of this distribution is the most likely value for the given input parameters and can be used by the decoder to find the corresponding mean transport values. One could also sample from the latent distributions provided by the conditional priors. If these samples are then decoded, one obtains a distribution over the transport values comparable to a mean value with error bars. This idea of the error bars fits nicely with the original codes as DKES also produces error bounds for the transport coefficients due to the variational principle in the code. These error bars are then omitted from the results given by DKES.

Chapter 7

Results and Discussion

7.1 Transport minimization

An important property of neural network applications is that we can automatically differentiate the neural network parameters with respect to its outputs. These principles can be exploited so gradients from individual reconstruction variables to the latent space can be computed, allowing for the minimization of that specific variable by moving in latent space. Using these gradient descent techniques[77], a minimization of the transport quantities was pursued. This resulted in a step-by-step improvement scheme for the model and trivial but promising results.

7.1.1 Temperature and density dominance

The first model used in this project was a plain VAE with a latent space spanning 16 dimensions as described in Section 3.2.1. After training, a minimization of the neoclassical electron flux density Γ_e was performed. This resulted in a change in latent space mainly governed by a decrease in temperature T and density n towards zero. Other variables also changed, but the decrease in temperature and density dominated the minimization.

These first trivial minimization solutions are fully explainable using an intuitive approach. First of all, taking away the temperature removes any kinetic theory of the particles in the system. Particles can not move around anymore, so there can be no transport. Also, setting the density of the particles to zero would result in no transport as there will be no particles or energy to be lost. Both of these can also be explained by checking the principles of transport described in Section 2.4. Looking at the electron-ion collision frequency described in Equation 2.23 one can find that the electron-ion collision frequency, ν_{ei}, goes to 0 for density $n \to 0$. This results in no random walks and thus a diffusion coefficient of 0. The influence of the temperature can not be explained using the same equation as this equation assumes a small angle of deflection due to high velocities and thus higher temperatures.

7.1.2 Gradients effect

After adjusting the minimization scheme to prevent setting the temperature and density to zero, new ways to reduce the transport needed to be found. At this stage, the model starts to lower the logarithmic gradients of both the temperature and the density. Again, these adjustments dominated the move in latent space over more minor adjustments in other variables.

The logarithmic gradients $1/L_n$ and $1/L_T$, are normalized versions of the gradients where the gradient of a variable is divided by the variable itself, i.e. $\nabla n/n$ and $\nabla T/T$. The only option to decrease the logarithmic gradient is to reduce the gradient of that variable. The decrease in density gradient ∇n as a solution to our minimization problem can be confirmed by looking at the definition of the particle flux in Equation 2.18. Reducing the gradient of the density equally reduces the particle flux. The decrease in temperature gradient as a solution can be explained by Fourier's law in Equation 2.22. Decreasing the temperature gradient will reduce the heat flux. As the heat flux can also be defined as the average energy lost per particle $\langle E \rangle$, multiplied by the particle flux Γ, and the average energy is a function of the temperature $\langle E \rangle \sim T$, the heat flux Q goes as $Q \sim T \times \Gamma$. The model was not allowed to reduce the temperature, so the particle flux must also go down to reduce the heat flux.

Due to the enormous contributions of the plasma parameters compared to the geometry components, the model needed to be changed to allow further investigation. This required a change in modeling from the classical VAE-like Model 1.0 to Model 1.1, both described in Section 6.1. Using Model 1.1 allowed separating the geometry components from the plasma parameters and encode them in separate latent spaces. This allows for minimization by making changes in only one latent space while the other remains unchanged.

7.1.3 Innermost flux surface

After changes to the plasma parameters were prevented, only changes in the geometry could be made. This resulted in an interesting gradient dominating the minimization step. Moving in latent space towards the lowest flux results in a decrease in r, the flux surface label of the simulation, and thus the position where all other values are measured. The model, therefore, states that the radial transport is lower on flux surfaces closer to the magnetic axis.

This observation normally finds its explanation in an argument using the fact that the temperature and the density peak in the center of the plasma. This can be seen in Figure 7.1 where the density and temperature profiles for a discharge in W7-X are plotted. At low r, it can be seen that the gradients for both go to 0. For low gradients, an explanation of low transport has already been given in Section 7.1.2.

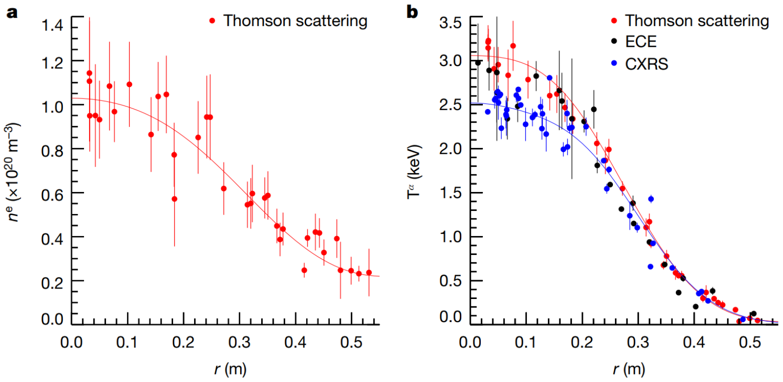


Figure 7.1: Density and temperature profiles for a typical discharge in W7-X. Fits to the experimental data are depicted as solid lines.[73].

In the data set used in this project, this argument does not work. Due to the parameter sweep performed to create the entire data set, many data points located at the inner flux surfaces have non-zero gradients. This behavior can be explained by inspecting the magnetic field on the innermost flux surfaces.

For flux surfaces closer to the magnetic axis, the field lines on these surfaces become straighter. This is reflected in a lower magnetic curvature on inner flux surfaces. As can be seen in Figure 7.2, where the normal curvature component κ_N decreases for lower values of r as R_0 is the major radius and thus constant. This normal curvature component is defined as the radial projection of the curvature vector $\boldsymbol{\kappa}$[19]. A lower magnetic curvature $\boldsymbol{\kappa}$ results in lower curvature drifts following Equation 2.7. As the curvature drift is a drift contributing to the random walk step discussed in Section 2.4.3, a lower curvature drift results in lower transport. The reduction of flux surface labels is, however, not a feasible solution as the total needs to have a certain volume and the losses need to be minimized for the entire volume.

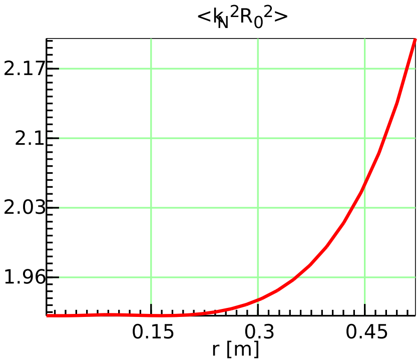


Figure 7.2: The RMS of the normal curvature component κ_N normalized by the major radius R_0 in the high mirror configuration of W7-X along the minor radius r.

7.1.4 Minor geometry changes

By adding a constraint that the minimization steps must remain on the specified radial position, the algorithm can only tweak the Fourier components R and Z defining the shape of the flux surface and B giving the magnetic field strength on that flux surface. This resulted in minor changes in the Fourier components making up the surface and magnetic field. Compared to all trivial minimization solutions discussed above, these are uncharted waters.

Starting from a random sample, these minor adjustments in the Fourier components can decrease neoclassical transport by up to 2%. The changes in Fourier components needed to realize this minimization of particle flux are too small to be compared in a visualization of the magnetic field. The changes made to the Fourier components can, however, be visualized. The relative changes made to the Fourier components spanning the surface, R and Z, and the magnetic field B for two minimization runs can be found in Figure 7.3 and 7.4. The figures show the relative change from two different randomly sampled starting points in the geometry latent space from $\mathcal{N}(0, 1)$. The sample from the plasma parameter latent space was chosen to be the mean along all dimensions for both cases. Applying the modifications shown in Figure 7.3 reduced the particle flux density by 0.315%. The changes suggested in Figure 7.4 result in a decrease of 1.63%. To reach a more significant reduction in transport, the minimization algorithm was allowed to take more steps along the steepest gradient. As can be seen by comparing the two figures, to reach a more significant decrease in transport, more components were adjusted, and these were also modified by a more significant amount. Furthermore, the relative change in Fourier components is larger for the higher Fourier components. Whereas the lower Fourier components, which contribute more to the general shape, only change by at most a few percent.

Two things must be noted while investigating these changes. The most significant changes might be because specific components were initialized closer to zero than other components. Therefore, the relative change to these components turn out to be bigger. Furthermore, it must also be noted that all Fourier components, R, Z and B are allowed to change. Therefore, the flux surface changes slightly and might not fulfill the equilibrium condition defined in Section 2.3.2 anymore. Lastly, the suggested improvements can not be verified. To verify these improvements, new DKES runs needed to be performed.

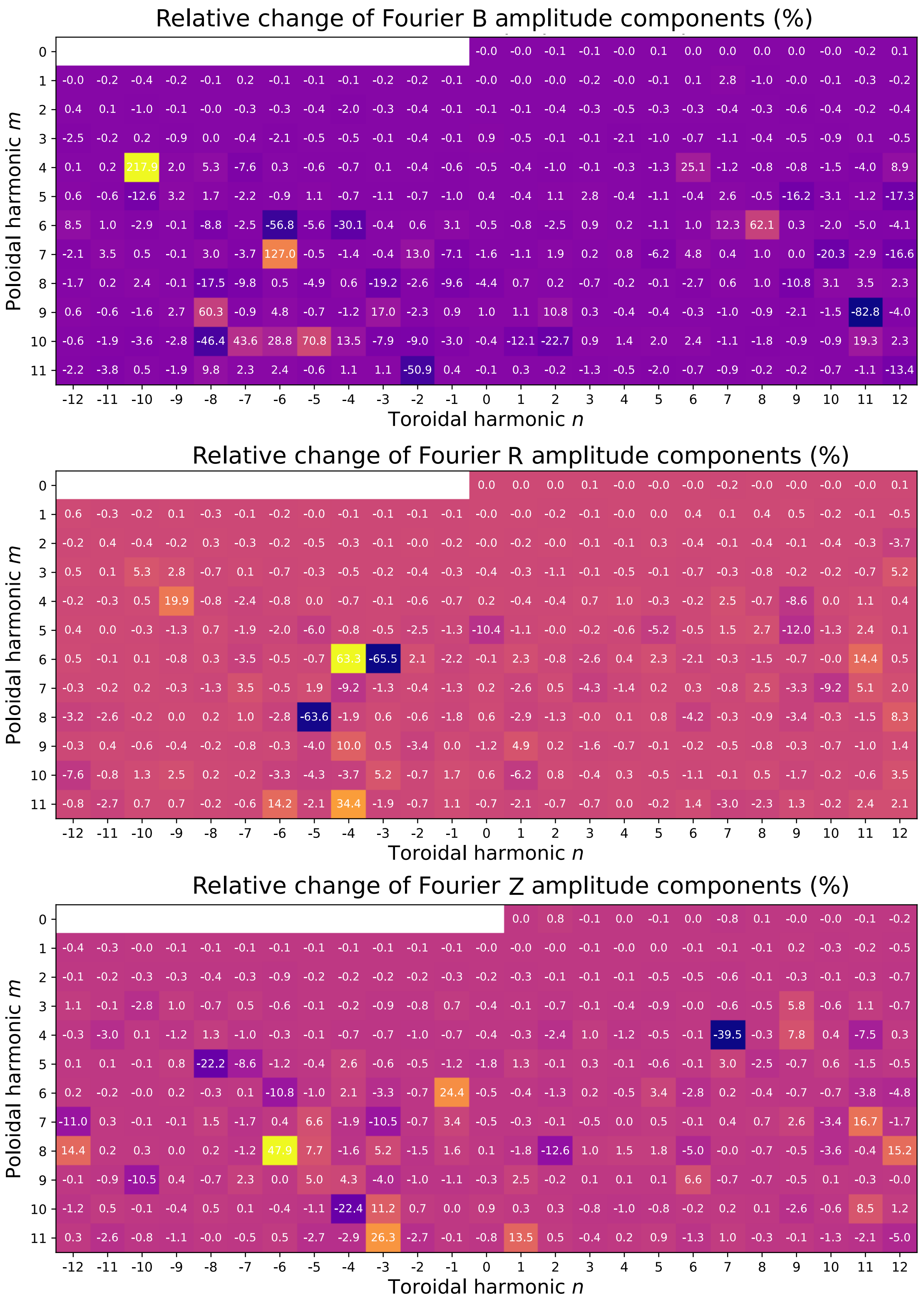
The small reduction in transport might be caused by the gradient descent method. This method follows the gradient in a certain direction and get stuck in a local minima. As, there is no clear idea on how the latent space is constructed, it is not clear if the algorithm is stuck in a local minima. The step size of the algorithm can be increased to get out of local minima. It can, however, also cause the algorithm to extrapolate into unreliable results.

7.1.5 Maximizing transport

In contrast to minimizing transport, one could also try to maximize transport. As the entire data set is filled with W7-X geometries designed to minimize transport, trying to move away from these optima might result in interesting behavior.

Starting from the same sample, the changes required to minimize the transport can be compared with the changes needed for maximization of transport. For a randomly sampled starting point in the geometry latent space from $\mathcal{N}(0,1)$ the minimization algorithm reduced the transport by 0.08%, and the maximization algorithm increased the transport by 0.12%. The respective relative changes made to the Fourier B components can be found in Figure 7.5a and 7.5b.

By comparing the adjustments made to maximize and minimize the transport, it can be noted that the significant changes are located at the same harmonics. This can not simply be explained by the fact that minimizing and maximizing require opposite gradients. Initially it might, starting from the same position in latent space, the steepest gradient descent and ascent might be in opposite directions. However, after one step, these points are in different places in latent space and can follow their own steepest gradients. After many steps, it is interesting to see that some components contribute more than others. It must be noted that if these processes start from a different starting point, other components are highlighted as the biggest changers. So there is, based on this, no evidence for a single component contributing a significant amount. One could systematically go through the minimization changes for lots of randomly sampled starting point in the geometry latent space. By analyzing these changes for all these starting points, a strong correlation might be found. This concept remains to be investigated.



Relative change of Fourier B amplitude components (%)

	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	10	11	12
0													-0.0	-0.0	-0.1	-0.1	-0.0	0.1	0.0	0.0	0.0	0.0	-0.0	-0.2	0.1
1	-0.0	-0.2	-0.4	-0.2	-0.1	0.2	-0.1	-0.1	-0.1	-0.2	-0.2	-0.1	-0.0	-0.0	-0.0	-0.2	-0.0	-0.1	0.1	2.8	-1.0	-0.0	-0.1	-0.3	-0.2
2	0.4	0.1	-1.0	-0.1	-0.0	-0.3	-0.3	-0.4	-2.0	-0.3	-0.4	-0.1	-0.1	-0.1	-0.4	-0.3	-0.5	-0.3	-0.3	-0.4	-0.3	-0.6	-0.4	-0.2	-0.4
3	-2.5	-0.2	0.2	-0.9	0.0	-0.4	-2.1	-0.5	-0.5	-0.1	-0.4	-0.1	0.9	-0.5	-0.1	-0.1	-2.1	-1.0	-0.7	-1.1	-0.4	-0.5	-0.9	0.1	-0.5
4	0.1	0.2	217.9	2.0	5.3	-7.6	0.3	-0.6	-0.7	0.1	-0.4	-0.6	-0.5	-0.4	-1.0	-0.1	-0.3	-1.3	25.1	-1.2	-0.8	-0.8	-1.5	-4.0	8.9
5	0.6	-0.6	-12.6	3.2	1.7	-2.2	-0.9	1.1	-0.7	-1.1	-0.7	-1.0	0.4	-0.4	1.1	2.8	-0.4	-1.1	-0.4	2.6	-0.5	-16.2	-3.1	-1.2	-17.3
6	8.5	1.0	-2.9	-0.1	-8.8	-2.5	-56.8	-5.6	-30.1	-0.4	0.6	3.1	-0.5	-0.8	-2.5	0.9	0.2	-1.1	1.0	12.3	62.1	0.3	-2.0	-5.0	-4.1
7	-2.1	3.5	0.5	-0.1	3.0	-3.7	127.0	-0.5	-1.4	-0.4	13.0	-7.1	-1.6	-1.1	1.9	0.2	0.8	-6.2	4.8	0.4	1.0	0.0	-20.3	-2.9	-16.6
8	-1.7	0.2	2.4	-0.1	-17.5	-9.8	0.5	-4.9	0.6	-19.2	-2.6	-9.6	-4.4	0.7	0.2	-0.7	-0.2	-0.1	-2.7	0.6	1.0	-10.8	3.1	3.5	2.3
9	0.6	-0.6	-1.6	2.7	60.3	-0.9	4.8	-0.7	-1.2	17.0	-2.3	0.9	1.0	1.1	10.8	0.3	-0.4	-0.4	-0.3	-1.0	-0.9	-2.1	-1.5	-82.8	-4.0
10	-0.6	-1.9	-3.6	-2.8	-46.4	43.6	28.8	70.8	13.5	-7.9	-9.0	-3.0	-0.4	-12.1	-22.7	0.9	1.4	2.0	2.4	-1.1	-1.8	-0.9	-0.9	19.3	2.3
11	-2.2	-3.8	0.5	-1.9	9.8	2.3	2.4	-0.6	1.1	1.1	-50.9	0.4	-0.1	0.3	-0.2	-1.3	-0.5	-2.0	-0.7	-0.9	-0.2	-0.2	-0.7	-1.1	-13.4

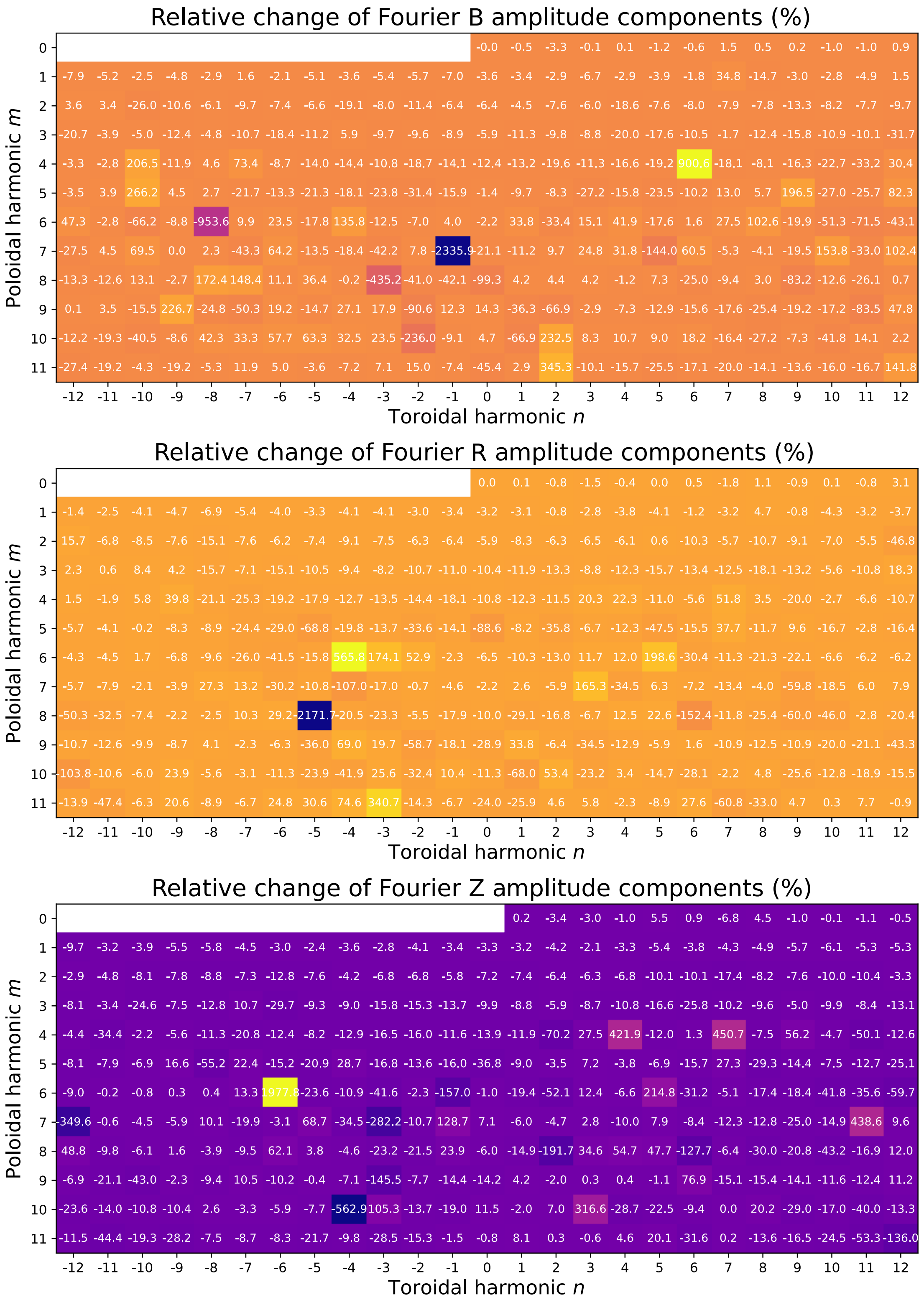
Relative change of Fourier R amplitude components (%)

	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	10	11	12
0													0.0	0.0	0.0	0.1	-0.0	-0.0	-0.0	-0.2	-0.0	-0.0	-0.0	-0.0	0.1
1	0.6	-0.3	-0.2	0.1	-0.3	-0.1	-0.2	-0.0	-0.1	-0.1	-0.1	-0.1	-0.0	-0.0	-0.2	-0.1	-0.0	0.0	0.4	0.1	0.4	0.5	-0.2	-0.1	-0.5
2	-0.2	0.4	-0.4	-0.2	0.3	-0.3	-0.2	-0.5	-0.3	-0.1	-0.0	-0.2	-0.0	-0.2	-0.0	-0.1	-0.1	0.3	-0.4	-0.1	-0.4	-0.1	-0.4	-0.3	-3.7
3	0.5	0.1	5.3	2.8	-0.7	0.1	-0.7	-0.3	-0.5	-0.2	-0.4	-0.3	-0.4	-0.3	-1.1	-0.1	-0.5	-0.1	-0.7	-0.3	-0.8	-0.2	-0.2	-0.7	5.2
4	-0.2	-0.3	0.5	19.9	-0.8	-2.4	-0.8	0.0	-0.7	-0.1	-0.6	-0.7	0.2	-0.4	-0.4	0.7	1.0	-0.3	-0.2	2.5	-0.7	-8.6	0.0	1.1	0.4
5	0.4	0.0	-0.3	-1.3	0.7	-1.9	-2.0	-6.0	-0.8	-0.5	-2.5	-1.3	-10.4	-1.1	-0.0	-0.2	-0.6	-5.2	-0.5	1.5	2.7	-12.0	-1.3	2.4	0.1
6	0.5	-0.1	0.1	-0.8	0.3	-3.5	-0.5	-0.7	63.3	-65.5	2.1	-2.2	-0.1	2.3	-0.8	-2.6	0.4	2.3	-2.1	-0.3	-1.5	-0.7	-0.0	14.4	0.5
7	-0.3	-0.2	0.2	-0.3	-1.3	3.5	-0.5	1.9	-9.2	-1.3	-0.4	-1.3	0.2	-2.6	0.5	-4.3	-1.4	0.2	0.3	-0.8	2.5	-3.3	-9.2	5.1	2.0
8	-3.2	-2.6	-0.2	0.0	0.2	1.0	-2.8	-63.6	-1.9	0.6	-0.6	-1.8	0.6	-2.9	-1.3	-0.0	0.1	0.8	-4.2	-0.3	-0.9	-3.4	-0.3	-1.5	8.3
9	-0.3	0.4	-0.6	-0.4	-0.2	-0.8	-0.3	-4.0	10.0	0.5	-3.4	0.0	-1.2	4.9	0.2	-1.6	-0.7	-0.1	-0.2	-0.5	-0.8	-0.3	-0.7	-1.0	1.4
10	-7.6	-0.8	1.3	2.5	0.2	-0.2	-3.3	-4.3	-3.7	5.2	-0.7	1.7	0.6	-6.2	0.8	-0.4	0.3	-0.5	-1.1	-0.1	0.5	-1.7	-0.2	-0.6	3.5
11	-0.8	-2.7	0.7	0.7	-0.2	-0.6	14.2	-2.1	34.4	-1.9	-0.7	1.1	-0.7	-2.1	-0.7	-0.7	0.0	-0.2	1.4	-3.0	-2.3	1.3	-0.2	2.4	2.1

Relative change of Fourier Z amplitude components (%)

	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	10	11	12
0														0.0	0.8	-0.1	0.0	-0.1	0.0	-0.8	0.1	-0.0	-0.0	-0.1	-0.2
1	-0.4	-0.3	-0.0	-0.1	-0.1	-0.1	-0.1	-0.1	-0.1	-0.1	-0.1	-0.1	-0.0	-0.0	-0.0	-0.1	-0.0	0.0	-0.1	-0.1	-0.1	0.2	-0.3	-0.2	-0.5
2	-0.1	-0.2	-0.3	-0.3	-0.4	-0.3	-0.9	-0.2	-0.2	-0.2	-0.2	-0.3	-0.2	-0.3	-0.1	-0.1	-0.1	-0.5	-0.5	-0.6	-0.1	-0.3	-0.1	-0.3	-0.7
3	1.1	-0.1	-2.8	1.0	-0.7	0.5	-0.6	-0.1	-0.2	-0.9	-0.8	0.7	-0.4	-0.1	-0.7	-0.1	-0.4	-0.9	-0.0	-0.6	-0.5	5.8	-0.6	1.1	-0.7
4	-0.3	-3.0	0.1	-1.2	1.3	-1.0	-0.3	-0.1	-0.7	-0.7	-1.0	-0.7	-0.4	-0.3	-2.4	1.0	-1.2	-0.5	-0.1	-39.5	-0.3	7.8	0.4	-7.5	0.3
5	0.1	0.1	-0.1	0.8	-22.2	-8.6	-1.2	-0.4	2.6	-0.6	-0.5	-1.2	-1.8	1.3	-0.1	0.3	-0.1	-0.6	-0.1	3.0	-2.5	-0.7	0.6	-1.5	-0.5
6	0.2	-0.2	-0.0	0.2	-0.3	0.1	-10.8	-1.0	2.1	-3.3	-0.7	24.4	-0.5	-0.4	-1.3	0.2	-0.5	3.4	-2.8	0.2	-0.4	-0.7	-0.7	-3.8	-4.8
7	-11.0	0.3	-0.1	-0.1	1.5	-1.7	0.4	6.6	-1.9	-10.5	-0.7	3.4	-0.5	-0.3	-0.1	-0.5	0.0	0.5	-0.1	0.4	0.7	2.6	-3.4	16.7	-1.7
8	14.4	0.2	0.3	0.0	0.2	-1.2	47.9	7.7	-1.6	5.2	-1.5	1.6	0.1	-1.8	-12.6	1.0	1.5	1.8	-5.0	-0.0	-0.7	-0.5	-3.6	-0.4	15.2
9	-0.1	-0.9	-10.5	0.4	-0.7	2.3	0.0	5.0	4.3	-4.0	-1.0	-1.1	-0.3	2.5	-0.2	0.1	0.1	0.1	6.6	-0.7	-0.7	-0.5	0.1	-0.3	-0.0
10	-1.2	0.5	-0.1	-0.4	0.5	0.1	-0.4	-1.1	-22.4	11.2	0.7	0.0	0.9	0.3	0.3	-0.8	-1.0	-0.8	-0.2	0.2	0.1	-2.6	-0.6	8.5	1.2
11	0.3	-2.6	-0.8	-1.1	-0.0	-0.5	0.5	-2.7	-2.9	26.3	-2.7	-0.1	-0.8	13.5	0.5	-0.4	0.2	0.9	-1.3	1.0	-0.3	-0.1	-1.3	-2.1	-5.0

Figure 7.3: The relative changes in Fourier components of B, R and Z, acquired by minimizing the neoclassical electron flux density Γ_e from a randomly sampled starting point. These changes resulted in a neoclassical electron flux density decrease of 0.315%. The colormap is dominated by a handful of outliers where blue represent the low values and yellow the high values.



Relative change of Fourier B amplitude components (%)

	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	10	11	12
0													-0.0	-0.5	-3.3	-0.1	0.1	-1.2	-0.6	1.5	0.5	0.2	-1.0	-1.0	0.9
1	-7.9	-5.2	-2.5	-4.8	-2.9	1.6	-2.1	-5.1	-3.6	-5.4	-5.7	-7.0	-3.6	-3.4	-2.9	-6.7	-2.9	-3.9	-1.8	34.8	-14.7	-3.0	-2.8	-4.9	1.5
2	3.6	3.4	-26.0	-10.6	-6.1	-9.7	-7.4	-6.6	-19.1	-8.0	-11.4	-6.4	-6.4	-4.5	-7.6	-6.0	-18.6	-7.6	-8.0	-7.9	-7.8	-13.3	-8.2	-7.7	-9.7
3	-20.7	-3.9	-5.0	-12.4	-4.8	-10.7	-18.4	-11.2	5.9	-9.7	-9.6	-8.9	-5.9	-11.3	-9.8	-8.8	-20.0	-17.6	-10.5	-1.7	-12.4	-15.8	-10.9	-10.1	-31.7
4	-3.3	-2.8	206.5	-11.9	4.6	73.4	-8.7	-14.0	-14.4	-10.8	-18.7	-14.1	-12.4	-13.2	-19.6	-11.3	-16.6	-19.2	900.6	-18.1	-8.1	-16.3	-22.7	-33.2	30.4
5	-3.5	3.9	266.2	4.5	2.7	-21.7	-13.3	-21.3	-18.1	-23.8	-31.4	-15.9	-1.4	-9.7	-8.3	-27.2	-15.8	-23.5	-10.2	13.0	5.7	196.5	-27.0	-25.7	82.3
6	47.3	-2.8	-66.2	-8.8	-953.6	9.9	23.5	-17.8	135.8	-12.5	-7.0	4.0	-2.2	33.8	-33.4	15.1	41.9	-17.6	1.6	27.5	102.6	-19.9	-51.3	-71.5	-43.1
7	-27.5	4.5	69.5	0.0	2.3	-43.3	64.2	-13.5	-18.4	-42.2	7.8	-2335.9	-21.1	-11.2	9.7	24.8	31.8	-144.0	60.5	-5.3	-4.1	-19.5	153.8	-33.0	102.4
8	-13.3	-12.6	13.1	-2.7	172.4	148.4	11.1	36.4	-0.2	-435.2	-41.0	-42.1	-99.3	4.2	4.4	4.2	-1.2	7.3	-25.0	-9.4	3.0	-83.2	-12.6	-26.1	0.7
9	0.1	3.5	-15.5	226.7	-24.8	-50.3	19.2	-14.7	27.1	17.9	-90.6	12.3	14.3	-36.3	-66.9	-2.9	-7.3	-12.9	-15.6	-17.6	-25.4	-19.2	-17.2	-83.5	47.8
10	-12.2	-19.3	-40.5	-8.6	42.3	33.3	57.7	63.3	32.5	23.5	-236.0	-9.1	4.7	-66.9	232.5	8.3	10.7	9.0	18.2	-16.4	-27.2	-7.3	-41.8	14.1	2.2
11	-27.4	-19.2	-4.3	-19.2	-5.3	11.9	5.0	-3.6	-7.2	7.1	15.0	-7.4	-45.4	2.9	345.3	-10.1	-15.7	-25.5	-17.1	-20.0	-14.1	-13.6	-16.0	-16.7	141.8

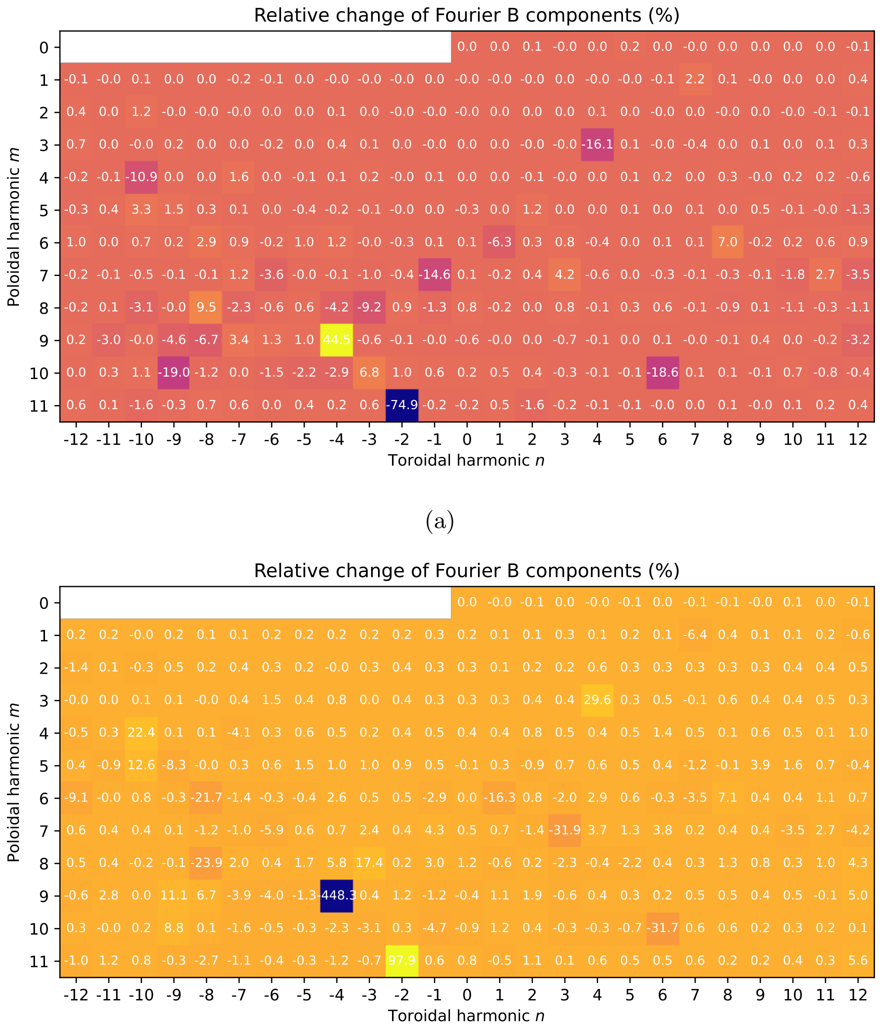
Relative change of Fourier R amplitude components (%)

	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	10	11	12
0													0.0	0.1	-0.8	-1.5	-0.4	0.0	0.5	-1.8	1.1	-0.9	0.1	-0.8	3.1
1	-1.4	-2.5	-4.1	-4.7	-6.9	-5.4	-4.0	-3.3	-4.1	-4.1	-3.0	-3.4	-3.2	-3.1	-0.8	-2.8	-3.8	-4.1	-1.2	-3.2	4.7	-0.8	-4.3	-3.2	-3.7
2	15.7	-6.8	-8.5	-7.6	-15.1	-7.6	-6.2	-7.4	-9.1	-7.5	-6.3	-6.4	-5.9	-8.3	-6.3	-6.5	-6.1	0.6	-10.3	-5.7	-10.7	-9.1	-7.0	-5.5	-46.8
3	2.3	0.6	8.4	4.2	-15.7	-7.1	-15.1	-10.5	-9.4	-8.2	-10.7	-11.0	-10.4	-11.9	-13.3	-8.8	-12.3	-15.7	-13.4	-12.5	-18.1	-13.2	-5.6	-10.8	18.3
4	1.5	-1.9	5.8	39.8	-21.1	-25.3	-19.2	-17.9	-12.7	-13.5	-14.4	-18.1	-10.8	-12.3	-11.5	20.3	22.3	-11.0	-5.6	51.8	3.5	-20.0	-2.7	-6.6	-10.7
5	-5.7	-4.1	-0.2	-8.3	-8.9	-24.4	-29.0	-68.8	-19.8	-13.7	-33.6	-14.1	-88.6	-8.2	-35.8	-6.7	-12.3	-47.5	-15.5	37.7	-11.7	9.6	-16.7	-2.8	-16.4
6	-4.3	-4.5	1.7	-6.8	-9.6	-26.0	-41.5	-15.8	565.8	174.1	52.9	-2.3	-6.5	-10.3	-13.0	11.7	12.0	198.6	-30.4	-11.3	-21.3	-22.1	-6.6	-6.2	-6.2
7	-5.7	-7.9	-2.1	-3.9	27.3	13.2	-30.2	-10.8	-107.0	-17.0	-0.7	-4.6	-2.2	2.6	-5.9	165.3	-34.5	6.3	-7.2	-13.4	-4.0	-59.8	-18.5	6.0	7.9
8	-50.3	-32.5	-7.4	-2.2	-2.5	10.3	29.2	-2171.7	-20.5	-23.3	-5.5	-17.9	-10.0	-29.1	-16.8	-6.7	12.5	22.6	-152.4	-11.8	-25.4	-60.0	-46.0	-2.8	-20.4
9	-10.7	-12.6	-9.9	-8.7	4.1	-2.3	-6.3	-36.0	69.0	19.7	-58.7	-18.1	-28.9	33.8	-6.4	-34.5	-12.9	-5.9	1.6	-10.9	-12.5	-10.9	-20.0	-21.1	-43.3
10	-103.8	-10.6	-6.0	23.9	-5.6	-3.1	-11.3	-23.9	-41.9	25.6	-32.4	10.4	-11.3	-68.0	53.4	-23.2	3.4	-14.7	-28.1	-2.2	4.8	-25.6	-12.8	-18.9	-15.5
11	-13.9	-47.4	-6.3	20.6	-8.9	-6.7	24.8	30.6	74.6	340.7	-14.3	-6.7	-24.0	-25.9	4.6	5.8	-2.3	-8.9	27.6	-60.8	-33.0	4.7	0.3	7.7	-0.9

Relative change of Fourier Z amplitude components (%)

	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	10	11	12
0														0.2	-3.4	-3.0	-1.0	5.5	0.9	-6.8	4.5	-1.0	-0.1	-1.1	-0.5
1	-9.7	-3.2	-3.9	-5.5	-5.8	-4.5	-3.0	-2.4	-3.6	-2.8	-4.1	-3.4	-3.3	-3.2	-4.2	-2.1	-3.3	-5.4	-3.8	-4.3	-4.9	-5.7	-6.1	-5.3	-5.3
2	-2.9	-4.8	-8.1	-7.8	-8.8	-7.3	-12.8	-7.6	-4.2	-6.8	-6.8	-5.8	-7.2	-7.4	-6.4	-6.3	-6.8	-10.1	-10.1	-17.4	-8.2	-7.6	-10.0	-10.4	-3.3
3	-8.1	-3.4	-24.6	-7.5	-12.8	10.7	-29.7	-9.3	-9.0	-15.8	-15.3	-13.7	-9.9	-8.8	-5.9	-8.7	-10.8	-16.6	-25.8	-10.2	-9.6	-5.0	-9.9	-8.4	-13.1
4	-4.4	-34.4	-2.2	-5.6	-11.3	-20.8	-12.4	-8.2	-12.9	-16.5	-16.0	-11.6	-13.9	-11.9	-70.2	27.5	421.9	-12.0	1.3	450.7	-7.5	56.2	-4.7	-50.1	-12.6
5	-8.1	-7.9	-6.9	16.6	-55.2	22.4	-15.2	-20.9	28.7	-16.8	-13.6	-16.0	-36.8	-9.0	-3.5	7.2	-3.8	-6.9	-15.7	27.3	-29.3	-14.4	-7.5	-12.7	-25.1
6	-9.0	-0.2	-0.8	0.3	0.4	13.3	1977.8	-23.6	-10.9	-41.6	-2.3	-157.0	-1.0	-19.4	-52.1	12.4	-6.6	214.8	-31.2	-5.1	-17.4	-18.4	-41.8	-35.6	-59.7
7	-349.6	-0.6	-4.5	-5.9	10.1	-19.9	-3.1	68.7	-34.5	-282.2	-10.7	128.7	7.1	-6.0	-4.7	2.8	-10.0	7.9	-8.4	-12.3	-12.8	-25.0	-14.9	438.6	9.6
8	48.8	-9.8	-6.1	1.6	-3.9	-9.5	62.1	3.8	-4.6	-23.2	-21.5	23.9	-6.0	-14.9	-191.7	34.6	54.7	47.7	-127.7	-6.4	-30.0	-20.8	-43.2	-16.9	12.0
9	-6.9	-21.1	-43.0	-2.3	-9.4	10.5	-10.2	-0.4	-7.1	-145.5	-7.7	-14.4	-14.2	4.2	-2.0	0.3	0.4	-1.1	76.9	-15.1	-15.4	-14.1	-11.6	-12.4	11.2
10	-23.6	-14.0	-10.8	-10.4	2.6	-3.3	-5.9	-7.7	-562.9	105.3	-13.7	-19.0	11.5	-2.0	7.0	316.6	-28.7	-22.5	-9.4	0.0	20.2	-29.0	-17.0	-40.0	-13.3
11	-11.5	-44.4	-19.3	-28.2	-7.5	-8.7	-8.3	-21.7	-9.8	-28.5	-15.3	-1.5	-0.8	8.1	0.3	-0.6	4.6	20.1	-31.6	0.2	-13.6	-16.5	-24.5	-53.3	-136.0

Figure 7.4: The relative changes in Fourier components of B, R and Z, acquired by minimizing the neoclassical electron flux density Γ_e from a randomly sampled starting point. These changes resulted in a neoclassical electron flux density decrease of 1.63%. The colormap is dominated by a handful of outliers where blue represent the low values and yellow the high values.



(a)

(b)

Figure 7.5: The relative changes in Fourier components of B, R and Z, for minimization (a) and maximization (b) of the neoclassical electron flux density Γ_e from the same randomly sampled starting point. These changes led to a decrease of 0.08% and an increase of 0.12%, respectively.

7.2 Delving into the posterior collapse

As already explained in Section 6.3, only using metrics is not enough to test the capabilities of a model. To be certain, simulations of new, unseen data using the generative model must be performed and checked. Moreover, using conventional methods like DKES to simulate new data is not easy. Therefore, an alternative to these conventional methods to simulate new data would be helpful.

Due to the difference in design described in Section 6.2.1, only Model 2 is able to function as a surrogate simulator. One of this model's biggest challenges is avoiding the posterior collapse described in Section 3.2.4. In general, the model is capable of reconstructing the training set, but generalization of the underlying problem is not present. Multiple additions and adaptations to the model or the data have been performed to overcome this problem.

7.2.1 Latent space dimensions

Model 2 started with low-dimensional latent spaces with the idea to make latent space exploration in a later stage of the project more manageable. After running model 2 on data sets of different sizes, posterior collapsing was identified. This was done by setting the input to the plasma parameter conditional prior to be constant and alternate the input to the geometry conditional prior. It was observed that alternating the input to the geometry conditional prior had almost no influence on the transport variables. Furthermore, looking into the mean values of the generated conditional prior, all values were extremely close to zero. The variances given by the conditional prior were also set close to zero.

As all information required to correctly model the transport variables during training was stored in the plasma latent space, it was decided to shrink this latent space. A lower amount of dimensions in this latent space would allow for less information to be stored in this latent space. The number of dimensions for the geometry latent space was increased to facilitate the compression of all information in the latent spaces. This should guide the model to let more information flow through this latent space. This resulted in a plasma latent space with 2 dimensions and a geometry latent space with 128 dimensions.

7.2.2 Loss function

With the changes mentioned above in latent space dimensions, the model has more possibilities to store information in the geometry latent space. This has been extensively tested on multiple data sets of different sizes. However, using the methods described above, posterior collapse was still present in the geometry latent space.

The usage of latent spaces and the storage of information in these latent spaces can be adjusted even more. As described in Section 3.2.4 the terms in the loss function of model 2 given by Equation 6.4 do not have to be weighed equally. First, the γ terms were added to guide the model in making certain choices. Lower values for γ increase the relative importance of the reconstruction term in the total loss function. For higher values of γ, the latent space is forced to follow a specific distribution, and because there are only a finite number of dimensions, these terms counteract each other. On top of this, because there are multiple latent spaces, using different weights for their regularization terms might guide the model even more. If one latent space is heavily regularized (high γ) and the other is not, storing more information in the latent space with the lower weight will be more favored. This is because deviating from the prior in the lower weighted latent space is far less penalized.

Moreover, the β term is added to try and stay clear of posterior collapse. The β term follows the Cyclic Annealing Schedule explained in Section 3.2.4. Due to the low amount of epochs used to train the model, the annealing schedule functions as an on/off switch. In a cycle of two epochs, it will first take the value 0 and the next epoch the value 1, toggling the KL-terms on and off. This allows the model to focus on the reconstruction and regularize the latent space afterward.

Applying these weights to the regularization terms drastically lowered the metrics during training. However, these terms did not stop a posterior collapse from occurring. Using a difference in the weights, such as $\gamma_1 = 10$ and $\gamma_2 = 0.01$ could still not prevent posterior collapse in the geometry latent space.

Also, the general idea of the loss function in a VAE-like model might need a change. The minimization of both the reconstruction term and the KL-terms might is a cause of posterior collapse. Minimizing the KL-terms towards zero as part of the main objective might force the model to find alternative methods with low metrics. Then the encoders die and cause posterior collapse.

7.2.3 Model architecture and training

Besides adjusting the latent space dimensions and the weights in the loss function, another significant factor in the modeling process is the layout of the model. In view of the posterior collapse, a few remarks must be made. In general, a model needs enough layers and nodes to deal with the data's complexity. However, in VAEs, if the decoders are very complex and powerful, it might compensate for bad regularization in the latent space[39]. This is a difficult factor to optimize as insightful theoretical literature on this topic is scarce. In this project, the complexity of the model was gradually decreased, but it did not result in avoiding posterior collapse.

Early stopping

As already stated, the metrics during training show that the reconstruction error of the training data is on an acceptable level. However, this does not generalize to a test set of unseen data. One way to overcome this might be to use early stopping rules. Early stopping is a form of regularization used to avoid overfitting when training a model with an iterative method. Up to a point, this improves the model's performance on data outside of the training set. Past that point, however, improving the model's fit to the training data comes at the expense of increased generalization error. Early stopping rules would stop the training once this tipping point is reached.

Due to the low amount of epochs used during training, this theory could easily be tested by running the model for a different amount of epochs. The problem of bad generalization and higher metric scores on the test set also arose when the model learns for a shorter amount of time. This might imply that the model overfits within the low amount of epochs already, or that the problem of bad generalization does not lie with training for a longer period but with posterior collapse.

7.2.4 Data reduction, scaling, and normalization

As described above, the model must be able to deal with the complexity of the data. This can be taken care of by adjusting the model's architecture, but one could also take another look at the complexity of the data. By investigating the results produced by Neotransp, some interesting cases were identified. The heat flux can be negative, while the particle and energy flux are positive. This results in a broader distribution with more variance. To help the model, these cases were filtered out of the data set.

Another option to reduce data complexity is using methods to transform the data. In Section 5.2 the methods of processing the data used to generate the results shown in this chapter are described. During this project, many different methods have been tried. Linear scaling, exponential scaling, normalization, and combinations of these methods have been tried. On top of these combinations, different methods have been used on different parts of the data. In this project, the most important factor was using exponential scaling to take out the large range in the flux as described in Section 5.2.

7.3 Latent space exploration

One of the promising ideas about these models is the exploration of latent space. As described in Section 3.1, the latent space is used to compress the data efficiently. To do this most efficiently, the model needs to make certain choices on how to map the input to the latent space. These choices can be investigated to find hidden relations in the data.

The first necessity before one can start exploring the latent space is a functioning model. The model needs to be able to generalize the behavior of the data and therefore have some understanding. From there on out, one can investigate why the model came up with these correlations to base its compression on. This first criterion has not been met for Model 2 as described in Section 7.2. Different exploration methods can be tried using Model 1.1 because the geometry latent space of this model is shaped following a Gaussian distribution $\mathcal{N}(0, 1)$ as a result of the KL-terms.

This exploration starts by walking along a single dimension in the geometry latent space measuring the changes in the neoclassical electron flux density to find which has the most influence while keeping the sample from the plasma latent space constant. To achieve this, a random sample from the plasma latent space is taken within 1σ. For the geometry latent space, a sample is hard coded by varying one dimension in latent space from -3 to 3 while keeping the other dimensions fixed at a value of 0. The results can be found in Figure 7.6.

A few things catch the eye when analyzing these results. First of all, the latent space variables have a nonlinear relation to the particle flux density. Also, it can be seen that some latent variables follow a similar trend. Lastly, the electron flux density values do not change much while walking along one dimension. Also, the flux density values always move in the same range for all dimensions. It is worth noting that these plots change for other samples from the parameter latent space. The trends per latent variables change but also the values of the transport density on the y-axis change.

The dependence of the plasma variables on the transport is still present in Model 1, as can be concluded from the significant changes in transport for different plasma parameter samples. Also, the changing of the trends of the latent variables shows that the latent space is highly nonlinear. If the difference between the minimal and maximal values along one dimension are computed, it is found that latent variable 5 has the most influence on the transport. This is true for any sample from the parameter space. By investigating the difference in geometry between the maximal and minimal transport by only changing latent variable 5, it can be found that latent variable 5 strongly affects the flux surface label. This was not allowed in the minimization algorithms used in Section 7.1.4, but as this is a linear sampling along an axis in latent space, this could not be avoided in this case. Therefore, analyzing the changes in Fourier components along axis 5 is not informative as these are all related to the flux surface label.

In the case explained above, only one latent variable is changed while the others are kept constant at 0. The same procedure is performed while keeping the other dimensions constant at 1 and -1. The results of these procedures can be found in appendix D. As can be seen by comparing these three figures, the behavior of some latent variables changes, but not for all. Latent variable

5, however, does not change at all and remains constant for all three explorations. This shows that the latent space is a complex space, and to get a clear overview of its workings, one must take steps along all dimensions simultaneously instead of keeping dimensions constant. This idea would, however, be tough to visualize as it would require visualizing anything in $n > 3$. Moreover, exploring the latent space of Model 1.1 will be dominated by the effects of the flux surface label found in latent variable 5. Additionally, sampling the geometry latent space of Model 2 would not be effective as the decoder effectively ignores it and the variance is small.

So far, there are a few papers showing that these methods work. Most papers use images as data sets because changes made in latent space are more easily identifiable for images, as described in Chapter 4. Latent space exploration, however, remains a difficult challenge, especially on complex data sets like the one used in this project.

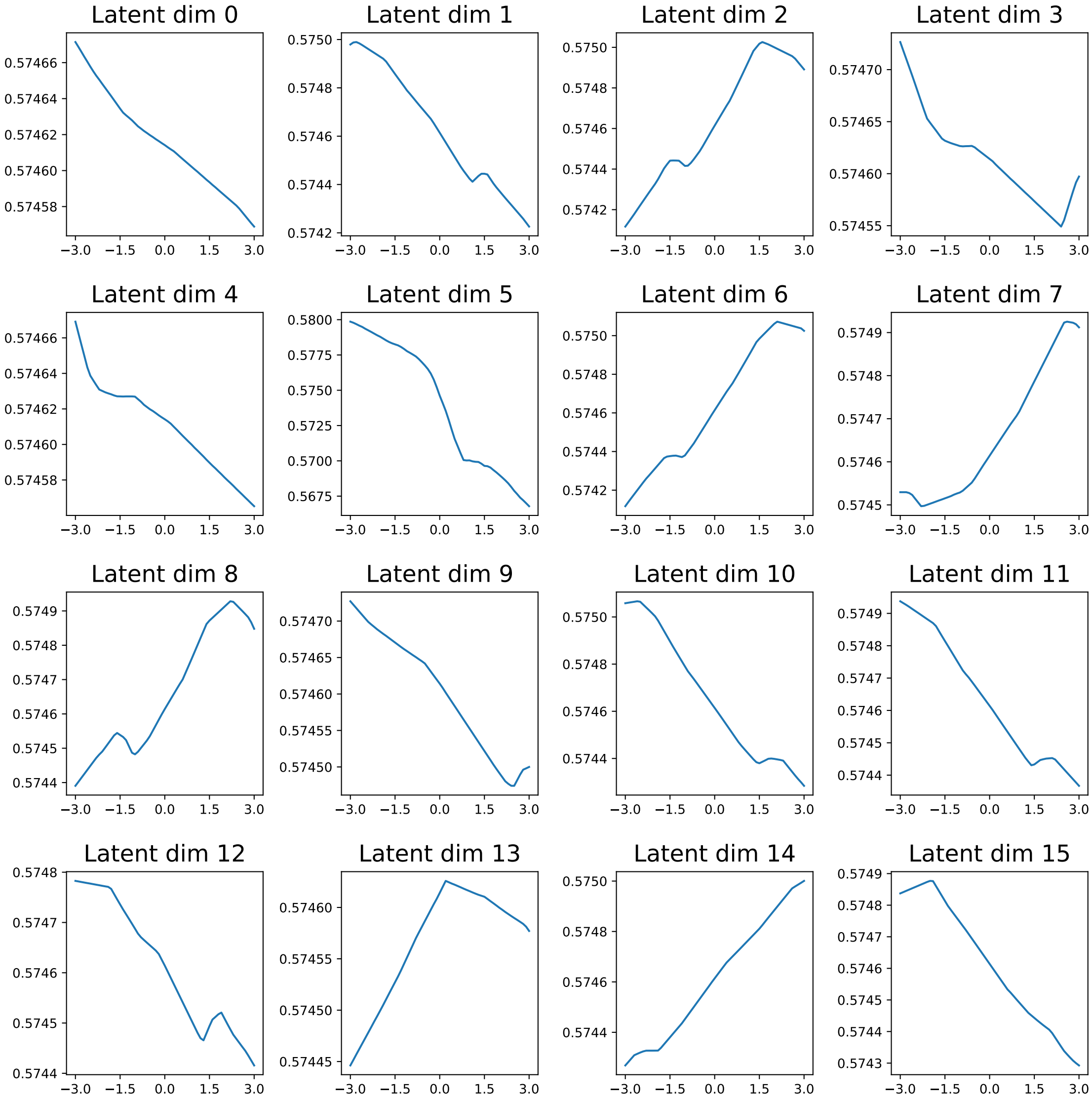


Figure 7.6: The scaled value of the neoclassical electron flux density along one latent space dimension from -3 to 3 while keeping the value of the other dimensions fixed at 0. for the plasma latent space, a random sample is picked from a Gaussian but within 1σ.

Chapter 8

Outlook and Conclusions

8.1 Conclusions

Looking back at the first research question: "*Can a variational autoencoder be used to generate synthetic results comparable to conventional simulation methods?*", it must be concluded based on our results, that posterior collapse is a major challenge. This is a common problem in many applications of variational autoencoders[36, 37, 38]. Numerous adaptations have been tried to overcome this problem, but without much success. Earlier models in this project did not suffer from posterior collapse as these were less regulated. These models show the ability to identify the strong relations hidden in the data set and can use these to minimize transport. However, due to other regularization conditions, new applications of these models were limited.

The second research question: "*Which new tasks are possible using the generative model after it has proven to be an accurate simulator?*", can not be fully answered. As the most promising model suffered from posterior collapse, we could not explore new possible tasks of this model. Because of this, earlier models needed to be used to explore new applications. One of these tasks is the minimization of transport. The earlier suggestions of the model to minimize transport could be substantiated by fundamental physical laws. The more complicated suggestions where transport is minimized by only varying the individual Fourier components of the magnetic field have not been theoretically verified. Furthermore, to verify these modifications, extra simulations using conventional methods must be done.

Moreover, the latent space of the earlier model has been partly explored. We showed that the latent space is highly complex, and to get a good overview, one must walk along all dimensions simultaneously. In this latent space, the most significant changes along one latent dimension were due to the change in flux surface label. We did not explore the latent space of Model 2 because of posterior collapse, making it unfit for the task in any case.

8.2 Future research

These unsupervised generative models can be helpful in the field of nuclear fusion. The applications discussed in this project might replace conventional, slower methods, and if one could explore the latent space in a structured way, this might result in the discovery of new physics. Or at least point them in the right direction. Especially when applied to an unsolved physical problem like turbulence. However, there is still work to be done. First and foremost, a solution to posterior collapse in this particular case needs to be found, or other models need that do not suffer from this problem need to be adopted.

Posterior collapse

Luckily, there are still many options to apply to the model or the data to try and overcome posterior collapse. A first modification can be done to the geometry data used in the project. "In this project, the truncation of the Fourier components was set to $m \leq 11$ and $|n| \leq 12$, but the contribution of some components is relatively small. They do, however contribute to the complexity of the data, which might not be in the model's best interest. Therefore, one should weigh the contributions of the higher Fourier components against the added complexity and find a reasonable trade off for the minimum number of Fourier components[78]. Moreover, it might be worth investigating if other possibilities exist to encode a geometry differently. Currently, the Fourier components result in a very evenly distributed data set as the same Fourier components for a surface are used for 10000 combinations of parameters. All these combinations with the same Fourier components result in different outcomes, which might be an extra challenge for the model. We chose this approach to allow for optimization of the transport and to prepare the method for experimental data, where varying plasma parameters are expected. To analyze the contributions of the geometry better, one should construct a data set where transport is computed for multiple geometries using the same plasma parameters.

Furthermore, modifications to the model's loss function can be made. Utilizing the influence of hyperparameters, a lot of different options can be tried. The general loss function could also be adjusted. An excellent overview of projects with solutions to their posterior collapse is given by 'sajadn' on GitHub[79]. Not all suggested solutions are applicable to this project, but more options are available. These could, however, not all be applied due to time constraints.

Adding physics

Other additions to this project are related to the use of physics in the model and the data. More variables are used in the simulation of DKES, those could be used in the ML model. One could for example add the local rotational transform ι and the poloidal current to the data set.

Furthermore, adding physics to the loss function might help the model overcome posterior collapse and make the results from a surrogate simulator more reliable and applicable[80]. Adding terms to the loss function to apply penalties if specific physical constraints are broken can assure that geometries suggested by the model follow the equilibrium condition. These suggestions were outside the scope of this project but might be a good addition for future bigger projects.

Latent space exploration

Lastly, as was already described in Chapter 4, there is still a lot to learn about the workings of the latent space. Gaining scientific knowledge by analyzing the workings of the model and the structure of the latent space is a promising research direction. To help this, one might apply more constraints to the latent space during training and, afterward, analyze how these might have contributed to shaping the latent space.

If this succeeds, we might gain insights into the variables govern the transport. This would not only be interesting for neoclassical transport, but could be expanded to other fields such as turbulent transport or even applications outside of fusion, such as fluid turbulence.

Bibliography

[1] R. K. Pachauri, M. R. Allen, V. R. Barros, J. Broome, W. Cramer, R. Christ, J. A. Church, L. Clarke, Q. Dahe, P. Dasgupta, N. K. Dubash, O. Edenhofer, I. Elgizouli, C. B. Field, P. Forster, P. Friedlingstein, J. Fuglestvedt, L. Gomez-Echeverri, S. Hallegatte, G. Hegerl, M. Howden, K. Jiang, B. Jimenez Cisneroz, V. Kattsov, H. Lee, K. J. Mach, J. Marotzke, M. D. Mastrandrea, L. Meyer, J. Minx, Y. Mulugetta, K. O'Brien, M. Oppenheimer, J. J. Pereira, R. Pichs-Madruga, G.-K. Plattner, Hans-Otto Pörtner, S. B. Power, B. Preston, N. H. Ravindranath, A. Reisinger, K. Riahi, M. Rusticucci, R. Scholes, K. Seyboth, Y. Sokona, R. Stavins, T. F. Stocker, P. Tschakert, D. van Vuuren, and J.-P. van Ypserle. *Climate Change 2014: Synthesis Report. Contribution of Working Groups I, II and III to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change.* IPCC, Geneva, Switzerland, 2014. (Cited on page 1.)

[2] Climate change. `https://obamawhitehouse.archives.gov/president-obama-climate-action-plan`. (Cited on page 1.)

[3] Keii Gi, Fuminori Sano, Keigo Akimoto, Ryoji Hiwatari, and Kenji Tobita. Potential contribution of fusion power generation to low-carbon development under the paris agreement and associated uncertainties. *Energy Strategy Reviews*, 27:100432, 2020. (Cited on page 1.)

[4] Advantages of fusion. `https://www.iter.org/sci/Fusion`. (Cited on page 1.)

[5] Sehila M. Gonzalez de Vicente, Nicholas A. Smith, Laila El-Guebaly, Sergio Ciattaglia, Luigi Di Pace, Mark Gilbert, Robert Mandoki, Sandrine Rosanvallon, Youji Someya, Kenji Tobita, and David Torcy. Overview on the management of radioactive waste from fusion facilities: ITER, demonstration machines and power plants. *Nuclear Fusion*, 62(8):085001, may 2022. (Cited on page 1.)

[6] *Status and Trends in Spent Fuel and Radioactive Waste Management.* Number NW-T-1.14 (Rev. 1) in Nuclear Energy Series. INTERNATIONAL ATOMIC ENERGY AGENCY, Vienna, 2022. (Cited on page 1.)

[7] Joaquin Sánchez. Nuclear fusion as a massive, clean, and inexhaustible energy source for the second half of the century: brief history, status, and perspective. *Energy Science & Engineering*, 2(4):165–176, 2014. (Cited on page 1.)

[8] Josefine Proll. *Trapped-particle instabilities in quasi-isodynamic stellarators.* PhD thesis, 01 2013. (Cited on pages iii, iii, 7, 8, and 9.)

[9] Lise-Marie Imbert-Gerard, Elizabeth J. Paul, and Adelle M. Wright. An introduction to stellarators: From magnetic fields to symmetries and optimization, 2020. (Cited on pages iii, iii, 9, and 10.)

[10] M. Endler, J. Baldzuhn, C.D. Beidler, H.-S. Bosch, S. Bozhenkov, B. Buttenschön, A. Dinklage, J. Fellinger, Y. Feng, G. Fuchert, Y. Gao, J. Geiger, O. Grulke, D. Hartmann, M. Jakubowski, R. König, H.P. Laqua, S. Lazerson, P. McNeely, D. Naujoks, U. Neuner, M. Otte, E. Pasch, T. Sunn Pedersen, V. Perseo, A. Puig Sitjes, K. Rahbarnia, N. Rust, O. Schmitz,

A. Spring, T. Stange, A. von Stechow, Y. Turkin, E. Wang, and R.C. Wolf. Wendelstein 7-x on the path to long-pulse high-performance operation. *Fusion Engineering and Design*, 167:112381, 2021. (Cited on pages iii and 11.)

[11] M. D. Kruskal and R. M. Kulsrud. Equilibrium of a magnetically confined plasma in a toroid. *The Physics of Fluids*, 1(4):265–274, 1958. (Cited on page 11.)

[12] Allen H. Boozer. Plasma equilibrium with rational magnetic surfaces. *The Physics of Fluids*, 24(11):1999–2003, 1981. (Cited on page 11.)

[13] Allen H. Boozer. Physics of magnetically confined plasmas. *Rev. Mod. Phys.*, 76:1071–1141, Jan 2005. (Cited on page 11.)

[14] D.T. Anderson. Transport in quasisymmetric plasma: Results from hsx. In *Proceedings of 2008 Innovative Confinement Concepts Workshop, Reno NV*. UW-Madison, 2008. (Cited on pages iii and 12.)

[15] S. P. Hirshman and J. C. Whitson. Steepest-descent moment method for three-dimensional magnetohydrodynamic equilibria. *The Physics of Fluids*, 26(12):3553–3568, 1983. (Cited on page 13.)

[16] Samuel A. Lazerson, Joaquim Loizu, Steven Hirshman, and Stuart R. Hudson. Verification of the ideal magnetohydrodynamic response at rational surfaces in the vmec code. *Physics of Plasmas*, 23(1):012507, 2016. (Cited on page 13.)

[17] P Helander, C D Beidler, T M Bird, M Drevlak, Y Feng, R Hatzky, F Jenko, R Kleiber, J H E Proll, Yu Turkin, and P Xanthopoulos. Stellarator and tokamak plasmas: a comparison. *Plasma Physics and Controlled Fusion*, 54(12):124009, nov 2012. (Cited on pages iv, 17, and 18.)

[18] C.D. Beidler, K. Allmaier, M.Yu. Isaev, S.V. Kasilov, W. Kernbichler, G.O. Leitold, H. Maaßberg, D.R. Mikkelsen, S. Murakami, M. Schmidt, D.A. Spong, V. Tribaldos, and A. Wakasa. Benchmarking of the mono-energetic transport coefficients—results from the international collaboration on neoclassical transport in stellarators (ICNTS). *Nuclear Fusion*, 51(7):076001, jun 2011. (Cited on pages iii, 17, and 19.)

[19] P. Helander. Theory of plasma confinement in non-axisymmetric magnetic fields. *Reports on progress in physics. Physical Society*, 77 8:087001, 2014. (Cited on pages 18 and 43.)

[20] J M Canik. Reduction of neoclassical transport and observation of a fast electron driven instability with quasisymmetry in hsx, Jul 2006. (Cited on page 18.)

[21] M. Gasparotto, F. Elio, B. Heinemann, N. Jaksic, B. Mendelevitch, J. Simon-Weidner, and B. Streibl. The wendelstein 7-x mechanical structure support elements: Design and tests. *Fusion Engineering and Design*, 74(1):161–165, 2005. Proceedings of the 23rd Symposium of Fusion Technology. (Cited on page 18.)

[22] S. P. Hirshman, K. C. Shaing, W. I. van Rij, C. O. Beasley, and E. C. Crume. Plasma transport coefficients for nonsymmetric toroidal confinement systems. *The Physics of Fluids*, 29(9):2951–2959, 1986. (Cited on page 19.)

[23] W. I. van Rij and S. P. Hirshman. Variational bounds for transport coefficients in three-dimensional toroidal plasmas. *Physics of Fluids B: Plasma Physics*, 1(3):563–569, 1989. (Cited on page 19.)

[24] Hakan Smith. Neotransp, availabla at: https://gitlab. mpcdf.mpg.de/smithh/neotransp. (Cited on page 19.)

[25] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2014. (Cited on page 21.)

[26] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014. (Cited on page 21.)

[27] Mark A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, 37(2):233–243, 1991. (Cited on page 22.)

[28] Joseph Rocca. Understanding variational autoencoders (vaes), Sep 2019. (Cited on pages iv, iv, 22, and 23.)

[29] S. Kullback and R. A. Leibler. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79 – 86, 1951. (Cited on page 23.)

[30] Jonathon Shlens. Notes on kullback-leibler divergence and likelihood, 2014. (Cited on page 23.)

[31] Shengjia Zhao, Jiaming Song, and Stefano Ermon. Infovae: Information maximizing variational autoencoders, 2017. (Cited on page 23.)

[32] Diederik P. Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick, 2015. (Cited on page 24.)

[33] Irina Higgins, Loïc Matthey, Arka Pal, Christopher P. Burgess, Xavier Glorot, Matthew M. Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *ICLR*, 2017. (Cited on page 24.)

[34] Irina Higgins, Loic Matthey, Xavier Glorot, Arka Pal, Benigno Uria, Charles Blundell, Shakir Mohamed, and Alexander Lerchner. Early visual concept learning with unsupervised deep learning, 2016. (Cited on page 24.)

[35] Irina Higgins, Arka Pal, Andrei A. Rusu, Loic Matthey, Christopher P Burgess, Alexander Pritzel, Matthew Botvinick, Charles Blundell, and Alexander Lerchner. Darla: Improving zero-shot transfer in reinforcement learning, 2017. (Cited on page 24.)

[36] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning, 2017. (Cited on pages 24 and 53.)

[37] James Lucas, George Tuckerand Roger Grosse, and Mohammad Norouzi. Understanding posterior collapse in generative latent variable models, 2019. (Cited on pages 24 and 53.)

[38] Anirudh Goyal, Alessandro Sordoni, Marc-Alexandre Côté, Nan Rosemary Ke, and Yoshua Bengio. Z-forcing: Training stochastic recurrent networks, 2017. (Cited on pages 24 and 53.)

[39] Xi Chen, Diederik P. Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. Variational lossy autoencoder, 2016. (Cited on pages 25 and 49.)

[40] Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Józefowicz, and Samy Bengio. Generating sentences from a continuous space. *CoRR*, abs/1511.06349, 2015. (Cited on page 25.)

[41] Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. 2015. (Cited on page 25.)

[42] Hao Fu, Chunyuan Li, Xiaodong Liu, Jianfeng Gao, Asli Celikyilmaz, and Lawrence Carin. Cyclical annealing schedule: A simple approach to mitigating KL vanishing. *CoRR*, abs/1903.10145, 2019. (Cited on pages iv, 25, and 26.)

[43] Kihyuk Sohn, H. Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *NIPS*, 2015. (Cited on page 26.)

[44] Maximilian Ilse, Jakub M. Tomczak, Christos Louizos, and Max Welling. Diva: Domain invariant variational autoencoders, 2019. (Cited on page 27.)

[45] Jonas Degrave, Federico Felici, Jonas Buchli, Michael Neunert, Brendan Tracey, Francesco Carpanese, Timo Ewalds, Roland Hafner, Abbas Abdolmaleki, Diego de las Casas, Craig Donner, Leslie Fritz, Cristian Galperti, Andrea Huber, James Keeling, Maria Tsimpoukelli, Jackie Kay, Antoine Merle, Jean-Marc Moret, Seb Noury, Federico Pesamosca, David Pfau, Olivier Sauter, Cristian Sommariva, Stefano Coda, Basil Duval, Ambrogio Fasoli, Pushmeet Kohli, Koray Kavukcuoglu, Demis Hassabis, and Martin Riedmiller. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602(7897):414–419, Feb 2022. (Cited on pages 29 and 30.)

[46] 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):022310, 2020. (Cited on page 29.)

[47] J Citrin, C Bourdelle, F J Casson, C Angioni, N Bonanomi, Y Camenen, X Garbet, L Garzotti, T Görler, O Gürcan, F Koechl, F Imbeaux, O Linder, K van de Plassche, P Strand, and G Szepesi and. Tractable flux-driven temperature, density, and rotation profile evolution with the quasilinear gyrokinetic transport model QuaLiKiz. *Plasma Physics and Controlled Fusion*, 59(12):124005, nov 2017. (Cited on page 29.)

[48] Andrea Merlo, Daniel Böckenhoff, Jonathan Schilling, Udo Höfel, Sehyun Kwak, Jakob Svensson, Andrea Pavone, Samuel Aaron Lazerson, and Thomas Sunn Pedersen. Proof of concept of a fast surrogate model of the VMEC code via neural networks in wendelstein 7-x scenarios. *Nuclear Fusion*, 61(9):096039, aug 2021. (Cited on pages 29 and 30.)

[49] Y. Wei, J.P. Levesque, C.J. Hansen, M.E. Mauel, and G.A. Navratil. A dimensionality reduction algorithm for mapping tokamak operational regimes using a variational autoencoder (VAE) neural network. *Nuclear Fusion*, 61(12):126063, nov 2021. (Cited on page 29.)

[50] Diogo R. Ferreira, Pedro J. Carvalho, Carlo Sozzi, Peter J. Lomas, and JET Contributors. Deep learning for the analysis of disruption precursors based on plasma tomography, 2020. (Cited on page 29.)

[51] Jinwon An and Sungzoon Cho. Variational autoencoder based anomaly detection using reconstruction probability. 2015. (Cited on page 29.)

[52] Olmo Cerri, Thong Nguyen, Maurizio Pierini, Maria Spiropulu, and Jean-Roch Vlimant. Variational autoencoders for new physics mining at the large hadron collider. *Journal of High Energy Physics*, 2019, 11 2018. (Cited on page 29.)

[53] Adrian Alan Pol, Victor Berger, Gianluca Cerminara, Cecile Germain, and Maurizio Pierini. Anomaly detection with conditional variational autoencoders, 2020. (Cited on page 29.)

[54] Ilia A. Luchnikov, Alexander Ryzhov, Pieter-Jan Stas, Sergey N. Filippov, and Henni Ouerdane. Variational autoencoder reconstruction of complex many-body physics. *Entropy*, 21(11):1091, Nov 2019. (Cited on page 29.)

[55] Andrea Rocchetto, Edward Grant, Sergii Strelchuk, Giuseppe Carleo, and Simone Severini. Learning hard quantum distributions with variational autoencoders. *npj Quantum Information*, 4(1):28, Jun 2018. (Cited on page 29.)

[56] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014. (Cited on page 29.)

[57] Michela Paganini, Luke de Oliveira, and Benjamin Nachman. Accelerating science with generative adversarial networks: An application to 3d particle showers in multilayer calorimeters. *Physical Review Letters*, 120(4), jan 2018. (Cited on page 29.)

[58] Luke de Oliveira, Michela Paganini, and Benjamin Nachman. Learning particle physics by example: Location-aware generative adversarial networks for physics synthesis. *Computing and Software for Big Science*, 1(1), sep 2017. (Cited on page 29.)

[59] Pasquale Musella and Francesco Pandolfi. Fast and accurate simulation of particle detectors using generative adversarial networks. *Computing and Software for Big Science*, 2(1), nov 2018. (Cited on page 29.)

[60] Amir Barati Farimani, Joseph Gomes, and Vijay S. Pande. Deep learning the physics of transport phenomena, 2017. (Cited on page 29.)

[61] Byungsoo Kim, Vinicius C. Azevedo, Nils Thuerey, Theodore Kim, Markus Gross, and Barbara Solenthaler. Deep fluids: A generative network for parameterized fluid simulations. *Computer Graphics Forum*, 38(2):59–70, may 2019. (Cited on page 29.)

[62] Akshay Subramaniam, Man Long Wong, Raunak D Borker, Sravya Nimmagadda, and Sanjiva K Lele. Turbulence enrichment using physics-informed generative adversarial networks, 2020. (Cited on page 29.)

[63] C. Drygala, B. Winhart, F. di Mare, and H. Gottschalk. Generative modeling of turbulence. *Physics of Fluids*, 34(3):035114, mar 2022. (Cited on page 29.)

[64] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A. Bharath. Generative adversarial networks: An overview. *IEEE Signal Processing Magazine*, 35(1):53–65, 2018. (Cited on page 29.)

[65] Wonkwang Lee, Donggyun Kim, Seunghoon Hong, and Honglak Lee. High-fidelity synthesis with disentangled representation, 2020. (Cited on page 29.)

[66] Defang Li, Min Zhang, Weifu Chen, and Guocan Feng. Facial attribute editing by latent space adversarial variational autoencoders. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 1337–1342, 2018. (Cited on page 30.)

[67] Shengyu Meng. Exploring in the latent space of design: A method of plausible building facades images generation, properties control and model explanation base on stylegan2. In Philip F. Yuan, Hua Chai, Chao Yan, and Neil Leach, editors, *Proceedings of the 2021 DigitalFUTURES*, pages 55–68, Singapore, 2022. Springer Singapore. (Cited on page 30.)

[68] Erik Härkönen, Aaron Hertzmann, Jaakko Lehtinen, and Sylvain Paris. Ganspace: Discovering interpretable gan controls. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 9841–9850. Curran Associates, Inc., 2020. (Cited on page 30.)

[69] Kosei Dohi. Variational autoencoders for jet simulation, 2020. (Cited on page 30.)

[70] David Winant, Joachim Schreurs, and Johan A. K. Suykens. Latent space exploration using generative kernel pca, 2021. (Cited on page 30.)

[71] Lu Mi, Tianxing He, Core Francisco Park, Hao Wang, Yue Wang, and Nir Shavit. Revisiting latent-space interpolation via a quantitative evaluation framework, 2021. (Cited on page 30.)

[72] Mike Yan Michelis and Quentin Becker. On linear interpolation in the latent space of deep generative models, 2021. (Cited on page 30.)

[73] C. D. Beidler, H. M. Smith, A. Alonso, T. Andreeva, J. Baldzuhn, M. N. A. Beurskens, M. Borchardt, S. A. Bozhenkov, K. J. Brunner, H. Damm, M. Drevlak, O. P. Ford, G. Fuchert, J. Geiger, P. Helander, U. Hergenhahn, M. Hirsch, U. Höfel, Ye. O. Kazakov, R. Kleiber, M. Krychowiak, S. Kwak, A. Langenberg, H. P. Laqua, U. Neuner, N. A. Pablant, E. Pasch, A. Pavone, T. S. Pedersen, K. Rahbarnia, J. Schilling, E. R. Scott, T. Stange, J. Svensson, H. Thomsen, Y. Turkin, F. Warmer, R. C. Wolf, D. Zhang, and the W7-X Team. Demonstration of reduced neoclassical energy transport in wendelstein 7-x. *Nature*, 596(7871):221–226, Aug 2021. (Cited on pages iv, 31, and 42.)

[74] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning, 2016. (Cited on page 37.)

[75] Xavier Glorot, Antoine Bordes, and Y. Bengio. Deep sparse rectifier neural networks. volume 15, 01 2010. (Cited on page 37.)

[76] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015. (Cited on page 37.)

[77] Claude Lemaréchal. Cauchy and the gradient method, 1847. (Cited on page 41.)

[78] Haifeng LIU, Akihiro SHIMIZU, Mitsutaka ISOBE, Shoichi OKAMURA, Shin NISHIMURA, Chihiro SUZUKI, Yuhong XU, Xin ZHANG, Bing LIU, Jie HUANG, Xianqu WANG, Hai LIU, Changjian TANG, Dapeng YIN, Yi WAN, and CFQS team. Magnetic configuration and modular coil design for the chinese first quasi-axisymmetric stellarator. *Plasma and Fusion Research*, 13:3405067–3405067, 2018. (Cited on page 54.)

[79] Sajadn. https://github.com/sajadn/posterior-collapse-list: A curated list of techniques to avoid posterior collapse, Nov 2019. (Cited on page 54.)

[80] Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Rätsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. 2018. (Cited on page 54.)

[81] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014. (Cited on page 65.)

Appendices

A KL-divergence for Gaussian distributions

Here a simple case of the KL-divergence for two Gaussian distributions $p(x)$ with a mean and standard deviation μ_p and σ_p, and $q(x)$, with a mean and standard deviation μ_q and σ_q is given.

By definition, the KL divergence is given by:

$$D_{KL}(p(x)\,||\,q(x)) = \int p(x)\,\log\left(\frac{p(x)}{q(x)}\right)\,dx \tag{A.1}$$

and the Gaussian distributions are given by:

$$\begin{aligned} p(x) &= \frac{1}{\sigma_p\sqrt{2\pi}}\,\exp\left(\frac{-(x-\mu_p)^2}{2\sigma_p^2}\right) \\ q(x) &= \frac{1}{\sigma_q\sqrt{2\pi}}\,\exp\left(\frac{-(x-\mu_q)^2}{2\sigma_q^2}\right) \end{aligned} \tag{A.2}$$

Splitting the logarithmic fraction allows us to solve them individually. This transforms Equation A.1 into:

$$D_{KL}(p(x)\,||\,q(x)) = \int p(x)\,\log p(x)\,dx \tag{A.3a}$$

$$-\int p(x)\,\log q(x)\,dx \tag{A.3b}$$

First, we'll solve the first term by substituting the definition of the Gaussian distribution into Equation A.3a resulting in:

$$\begin{aligned} &\int p(x)\,\log\left(\frac{1}{\sigma_p\sqrt{2\pi}}\,\exp\left(\frac{-(x-\mu_p)^2}{2\sigma_p^2}\right)\right)\,dx \\ = &\int p(x)\,\log\left(\frac{1}{\sigma_p\sqrt{2\pi}}\right)\,dx + \int p(x)\,\log\left(\exp\left(\frac{-(x-\mu_p)^2}{2\sigma_p^2}\right)\right)\,dx \end{aligned} \tag{A.4}$$

Splitting the logarithmic of the first term and cancelling the log and e in the second term results in the following:

$$\int p(x)\,\log 1\,dx - \int p(x)\,\log\!\left(\sigma_p\sqrt{2\pi}\right)dx + \int p(x)\left(\frac{-(x-\mu_p)^2}{2\sigma_p^2}\right)\,dx \tag{A.5}$$

The first term goes to zero as $\log 1 = 0$, for the second term we utilise the fact that the integration of a probability density function is 1 as:

$$\int p(x)\,dx = 1 \tag{A.6}$$

which transforms Equation A.5 into

$$\begin{aligned} &-\log\Big(\sigma_p\sqrt{2\pi}\Big) + \int p(x)\left(\frac{-(x-\mu_p)^2}{2\sigma_p^2}\right)dx \\ &= -\log\Big(\sigma_p\sqrt{2\pi}\Big) - \frac{1}{2\sigma_p^2}\int p(x)(x-\mu_p)^2\,dx \end{aligned} \tag{A.7}$$

Now we insert the definition of variance:

$$\int p(x)(x-\mu_p)^2\,dx = \sigma_p^2 \tag{A.8}$$

This results in the following for Equation A.3a

$$\int p(x)\,\log p(x)\,dx = -\log\Big(\sigma_p\sqrt{2\pi}\Big) - \frac{1}{2} = -\frac{1}{2}(1+\log\big(2\pi\sigma_p^2\big)) \tag{A.9}$$

Following the same principles for Equation A.3b as with Equation A.3a results in:

$$\begin{aligned} &\frac{1}{2}\log\big(2\pi\sigma_q^2\big) - \int p(x)\left(\frac{-(x-\mu_q)^2}{2\sigma_q^2}\right)dx \\ &= \frac{1}{2}\log\big(2\pi\sigma_q^2\big) + \frac{\int p(x)\,x^2\,dx - \int p(x)\,2x\,\mu_q\,dx + \int p(x)\,\mu_q^2\,dx}{2\sigma_q^2} \end{aligned} \tag{A.10}$$

Letting the $\langle\rangle$ denote the expectation operator under p, the above can be written as:

$$\frac{1}{2}\log\big(2\pi\sigma_q^2\big) + \frac{\langle x^2\rangle - 2\langle x\rangle\mu_q + \mu_q^2}{2\sigma_q^2} \tag{A.11}$$

Furthermore, we know that $var(x) = \langle x^2\rangle - \langle x\rangle$, so $\langle x^2\rangle = \sigma_p^2 + \mu_p^2$ and substituting this into Equation A.11 results in:

$$\frac{1}{2}\log\big(2\pi\sigma_q^2\big) + \frac{\sigma_p^2 + \mu_p^2 - 2\mu_p\mu_q + \mu_q^2}{2\sigma_q^2} = \frac{1}{2}\log\big(2\pi\sigma_q^2\big) + \frac{\sigma_p^2 + (\mu_p-\mu_q)^2}{2\sigma_q^2} \tag{A.12}$$

Adding Equation A.9 and A.12 for the total KL-divergence gives:

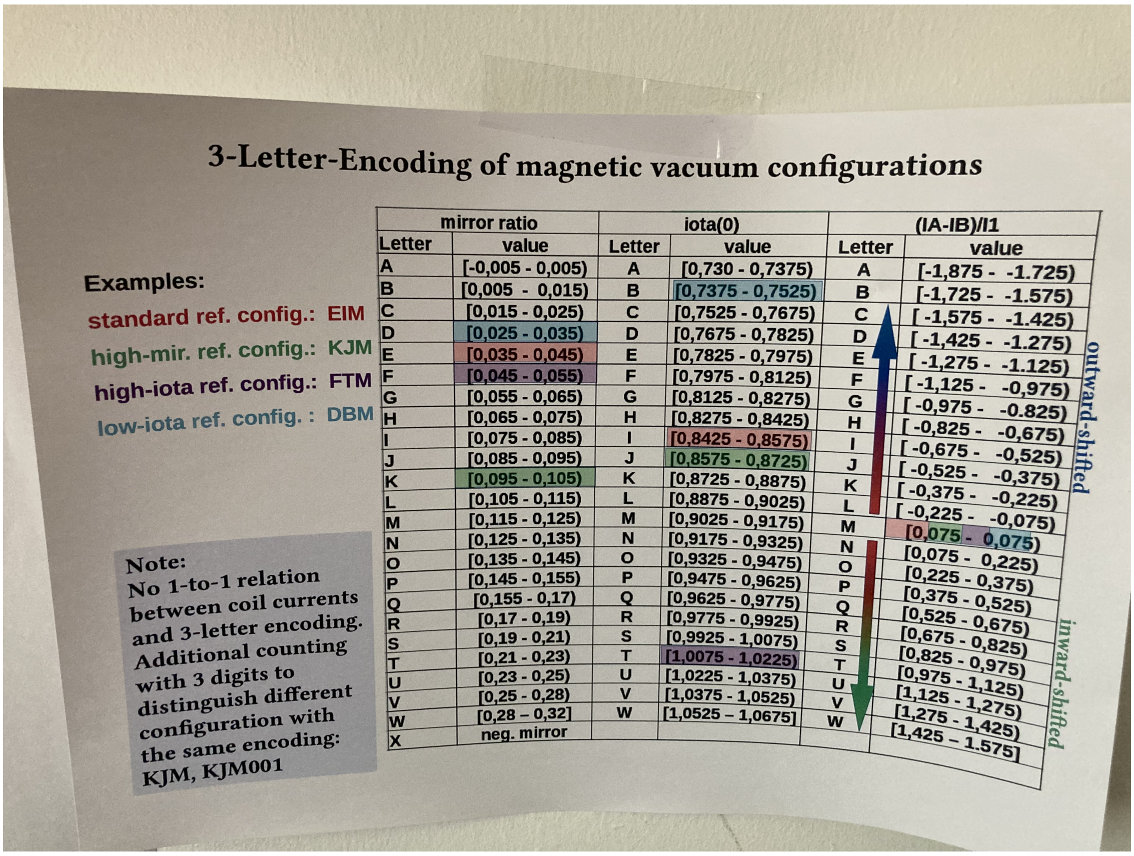
$$D_{KL}(p(x)\,||\,q(x)) = \frac{1}{2}\log\big(2\pi\sigma_q^2\big) + \frac{\sigma_p^2 + (\mu_p-\mu_q)^2}{2\sigma_q^2} - \frac{1}{2}(1+\log\big(2\pi\sigma_p^2\big)) \tag{A.13}$$

which can be simplified to

$$D_{KL}(p(x)\,||\,q(x)) = \log\frac{\sigma_q}{\sigma_p} + \frac{\sigma_p^2 + (\mu_p-\mu_q)^2}{2\sigma_q^2} - \frac{1}{2} \tag{A.14}$$

B Data appendix

B.1 Three letter codes



3-Letter-Encoding of magnetic vacuum configurations

Examples:
standard ref. config.: EIM
high-mir. ref. config.: KJM
high-iota ref. config.: FTM
low-iota ref. config. : DBM

mirror ratio		iota(0)		(IA-IB)/I1	
Letter	value	Letter	value	Letter	value
A	[-0,005 - 0,005)	A	[0,730 - 0,7375)	A	[-1,875 - -1.725)
B	[0,005 - 0,015)	B	[0,7375 - 0,7525)	B	[-1,725 - -1.575)
C	[0,015 - 0,025)	C	[0,7525 - 0,7675)	C	[-1,575 - -1.425)
D	[0,025 - 0,035)	D	[0,7675 - 0,7825)	D	[-1,425 - -1.275)
E	[0,035 - 0,045)	E	[0,7825 - 0,7975)	E	[-1,275 - -1.125)
F	[0,045 - 0,055)	F	[0,7975 - 0,8125)	F	[-1,125 - -0,975)
G	[0,055 - 0,065)	G	[0,8125 - 0,8275)	G	[-0,975 - -0.825)
H	[0,065 - 0,075)	H	[0,8275 - 0,8425)	H	[-0,825 - -0,675)
I	[0,075 - 0,085)	I	[0,8425 - 0,8575)	I	[-0,675 - -0,525)
J	[0,085 - 0,095)	J	[0,8575 - 0,8725)	J	[-0,525 - -0,375)
K	[0,095 - 0,105)	K	[0,8725 - 0,8875)	K	[-0,375 - -0,225)
L	[0,105 - 0,115)	L	[0,8875 - 0,9025)	L	[-0,225 - -0,075)
M	[0,115 - 0,125)	M	[0,9025 - 0,9175)	M	[0,075 - 0,075)
N	[0,125 - 0,135)	N	[0,9175 - 0,9325)	N	[0,075 - 0,225)
O	[0,135 - 0,145)	O	[0,9325 - 0,9475)	O	[0,225 - 0,375)
P	[0,145 - 0,155)	P	[0,9475 - 0,9625)	P	[0,375 - 0,525)
Q	[0,155 - 0,17)	Q	[0,9625 - 0,9775)	Q	[0,525 - 0,675)
R	[0,17 - 0,19)	R	[0,9775 - 0,9925)	R	[0,675 - 0,825)
S	[0,19 - 0,21)	S	[0,9925 - 1,0075)	S	[0,825 - 0,975)
T	[0,21 - 0,23)	T	[1,0075 - 1,0225)	T	[0,975 - 1,125)
U	[0,23 - 0,25)	U	[1,0225 - 1,0375)	U	[1,125 - 1,275)
V	[0,25 - 0,28)	V	[1,0375 - 1,0525)	V	[1,275 - 1,425)
W	[0,28 – 0,32]	W	[1,0525 – 1,0675]	W	[1,425 – 1.575]
X	neg. mirror				

Note:
No 1-to-1 relation between coil currents and 3-letter encoding. Additional counting with 3 digits to distinguish different configuration with the same encoding: KJM, KJM001

Figure B.1: The meaning of Joachim Geiger's three letter code configurations of W7-X.

B.2 Data scaling values

The values used for data scaling after the unlogical cases are filtered out.

Name	**Min**	**Max**
Radius r	$0.0502\,\mathrm{m}$	$0.630\,31\,\mathrm{m}$
Normalized radius r/a	0.10101	0.998298
Normalized flux label s	0.01020	0.996600
Radial Electric field E_r	$-18\,\mathrm{kV\,m^{-1}}$	$7\,\mathrm{kV\,m^{-1}}$
Density n_e	$0.1 \times 10^{20}\,\mathrm{m^{-3}}$	$1.5 \times 10^{20}\,\mathrm{m^{-3}}$
Density n_H	$0.1 \times 10^{20}\,\mathrm{m^{-3}}$	$1.5 \times 10^{20}\,\mathrm{m^{-3}}$
Temperature T_e	$0.1\,\mathrm{keV}$	$5\,\mathrm{keV}$
Temperature T_H	$0.1\,\mathrm{keV}$	$5\,\mathrm{keV}$
Logarithmic density gradient $1/L_{n_e}$	$-6\,\mathrm{m^{-1}}$	$-0.05\,\mathrm{m^{-1}}$
Logarithmic density gradient $1/L_{n_H}$	$-6\,\mathrm{m^{-1}}$	$-0.05\,\mathrm{m^{-1}}$
Logarithmic temperature gradient $1/L_{T_e}$	$-20\,\mathrm{m^{-1}}$	$-0.1\,\mathrm{m^{-1}}$
Logarithmic temperature gradient $1/L_{T_H}$	$-20\,\mathrm{m^{-1}}$	$-0.1\,\mathrm{m^{-1}}$
NC Flux density Γ_e	$9.497 \times 10^{10}\,\mathrm{m^{-2}\,s^{-1}}$	$2.041 \times 10^{23}\,\mathrm{m^{-2}\,s^{-1}}$
NC Flux density Γ_H	$9.497 \times 10^{10}\,\mathrm{m^{-2}\,s^{-1}}$	$2.047 \times 10^{23}\,\mathrm{m^{-2}\,s^{-1}}$
NC Heat Flux Q_e	$1.803 \times 10^{-13}\,\mathrm{MW\,m^{-2}}$	$597.693\,\mathrm{MW\,m^{-2}}$
NC Heat Flux Q_H	$4.890 \times 10^{-10}\,\mathrm{MW\,m^{-2}}$	$868.596\,\mathrm{MW\,m^{-2}}$
NC Energy Flux S_e	$3.962 \times 10^{-9}\,\mathrm{MW\,m^{-2}}$	$1006.485\,\mathrm{MW\,m^{-2}}$
NC Energy Flux S_H	$1.402 \times 10^{-8}\,\mathrm{MW\,m^{-2}}$	$1166.116\,\mathrm{MW\,m^{-2}}$
NC Diffusion Coefficient D_e	$3.326 \times 10^{-5}\,\mathrm{m^2\,s^{-1}}$	$234.625\,\mathrm{m^2\,s^{-1}}$
NC Diffusion Coefficient D_H	$0.000\,962\,2\,\mathrm{m^2\,s^{-1}}$	$709.249\,\mathrm{m^2\,s^{-1}}$

Table 1: The minimal and maximal values for the plasma parameters and the outcomes of Neotransp after the cases described in Section 5.2 are filtered out. These minima and maxima are then used to scale the values as was described in the same section.

B.3 Data implementation

The preprocessed data is also saved in Zarr format as a preprocessed file of a single geometry is in the order of several GB if not compressed by Zarr. However, Zarr is only designed for efficient data storage of big arrays, not for fast readability which is required for deep learning models. To solve this problem, the Zarr files are combined and stored in HDF5 format. This format does not have to be opened all at once and can be streamed to a model, using generator functions. However, because of the data streaming, shuffling can not be done by the model itself and has to be done beforehand and by the generator function. Shuffling data reduces the variance and makes sure that the model remain general and overfits less. Also, it ensures that each data batch creates an independent change on the model, without being biased by the same points before them. Using this transition from big Zarr files to singular HDF5 files allows the possibility to play with the number of geometries trained in the model. Furthermore, to decrease the total number of data points for practical reasons, a random selection of data points per geometry file is added to the HDF5 file. Lastly, HDF5 also allows the data to be used in parallel by machine learning models.

Streaming the data into the model requires a generator function as mentioned earlier. This generator function takes over some actions which are commonly handled by the machine learning model itself. First of all, the generator function divides the data into a training and test set where 20% of the data is used as test data. Then, the generator function is responsible for creating batches. Finally, the data is split into three dictionaries to ensure it can be read by different input layers.

C Models

C.1 Model 1.0

The layout of Model 1.0 can be found in Figure C.1 and C.2. All the data is encoded using one singular encoder into a latent space with 24 dimensions. It must be noted that the geometry data used for this model were only the Fourier B components and not the R and Z components. Therefore the layers working with the geometry data have far less nodes compared to the next models.

The decoder given in Figure C.2 takes a sample from the one latent space and tries to reconstruct all the given input data. As can be seen for both the encoder and the decoder, Dense layers are alternated with Batch normalization layers. All Dense layers work with a ReLU activation function except the last layers of the encoder and the decoder. The last layer of the encoder has no activation function and the last layer of the decoder has a Sigmoid activation function because the data is given in the range 0-1. The model was trained for a low amount of epochs in the order of 20. All models use Adam optimization[81].

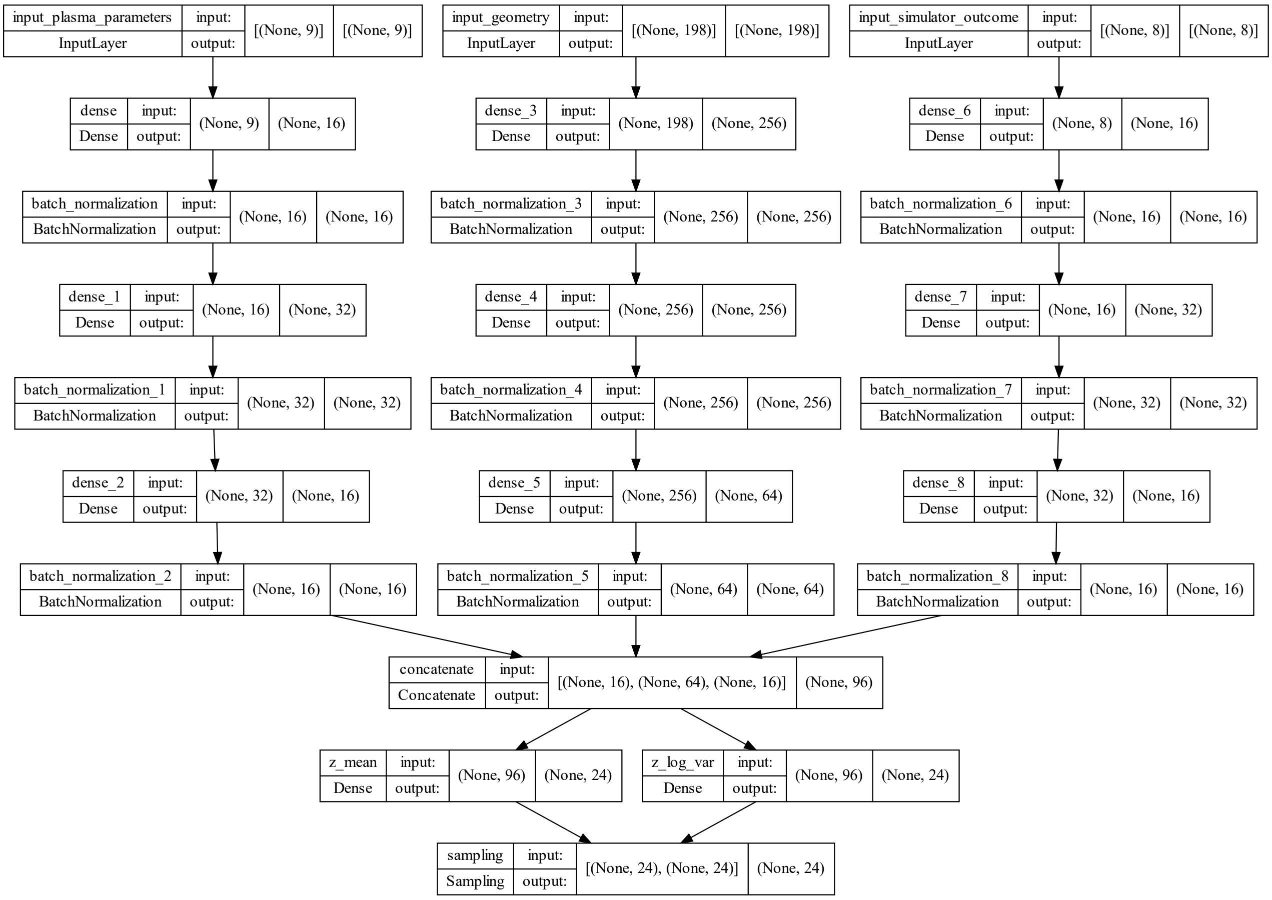

input_plasma_parameters | InputLayer | input: | output: | [(None, 9)] | [(None, 9)]
input_geometry | InputLayer | input: | output: | [(None, 198)] | [(None, 198)]
input_simulator_outcome | InputLayer | input: | output: | [(None, 8)] | [(None, 8)]
dense | Dense | input: | output: | (None, 9) | (None, 16)
dense_3 | Dense | input: | output: | (None, 198) | (None, 256)
dense_6 | Dense | input: | output: | (None, 8) | (None, 16)
batch_normalization | BatchNormalization | input: | output: | (None, 16) | (None, 16)
batch_normalization_3 | BatchNormalization | input: | output: | (None, 256) | (None, 256)
batch_normalization_6 | BatchNormalization | input: | output: | (None, 16) | (None, 16)
dense_1 | Dense | input: | output: | (None, 16) | (None, 32)
dense_4 | Dense | input: | output: | (None, 256) | (None, 256)
dense_7 | Dense | input: | output: | (None, 16) | (None, 32)
batch_normalization_1 | BatchNormalization | input: | output: | (None, 32) | (None, 32)
batch_normalization_4 | BatchNormalization | input: | output: | (None, 256) | (None, 256)
batch_normalization_7 | BatchNormalization | input: | output: | (None, 32) | (None, 32)
dense_2 | Dense | input: | output: | (None, 32) | (None, 16)
dense_5 | Dense | input: | output: | (None, 256) | (None, 64)
dense_8 | Dense | input: | output: | (None, 32) | (None, 16)
batch_normalization_2 | BatchNormalization | input: | output: | (None, 16) | (None, 16)
batch_normalization_5 | BatchNormalization | input: | output: | (None, 64) | (None, 64)
batch_normalization_8 | BatchNormalization | input: | output: | (None, 16) | (None, 16)
concatenate | Concatenate | input: | output: | [(None, 16), (None, 64), (None, 16)] | (None, 96)
z_mean | Dense | input: | output: | (None, 96) | (None, 24)
z_log_var | Dense | input: | output: | (None, 96) | (None, 24)
sampling | Sampling | input: | output: | [(None, 24), (None, 24)] | (None, 24)

Figure C.1: The full encoder side of Model 1.0. The input of this side is given by 3 input categories: geometry, simulator outcome and plasma parameters. The outcome of this side are the mean, ln($variance$) and a sample per dimension of the latent space.

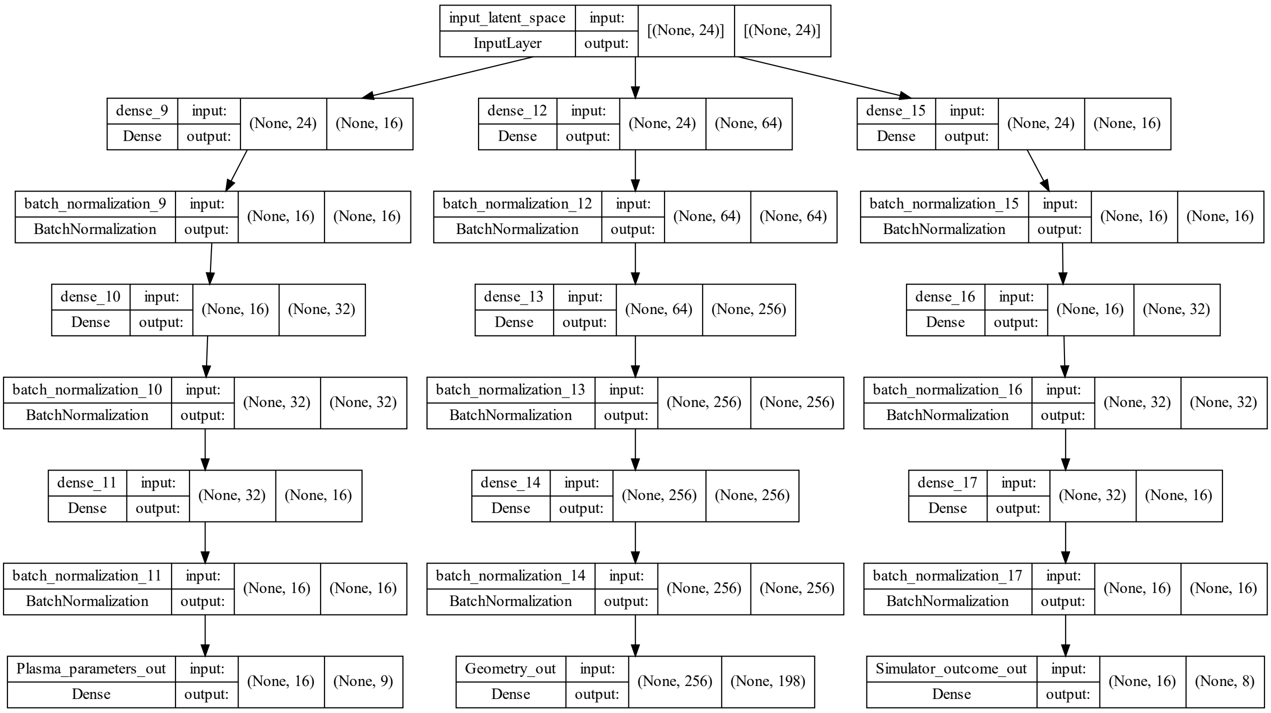

input_latent_space | InputLayer | input: | output: | [(None, 24)] | [(None, 24)]
dense_9 | Dense | input: | output: | (None, 24) | (None, 16)
dense_12 | Dense | input: | output: | (None, 24) | (None, 64)
dense_15 | Dense | input: | output: | (None, 24) | (None, 16)
batch_normalization_9 | BatchNormalization | input: | output: | (None, 16) | (None, 16)
batch_normalization_12 | BatchNormalization | input: | output: | (None, 64) | (None, 64)
batch_normalization_15 | BatchNormalization | input: | output: | (None, 16) | (None, 16)
dense_10 | Dense | input: | output: | (None, 16) | (None, 32)
dense_13 | Dense | input: | output: | (None, 64) | (None, 256)
dense_16 | Dense | input: | output: | (None, 16) | (None, 32)
batch_normalization_10 | BatchNormalization | input: | output: | (None, 32) | (None, 32)
batch_normalization_13 | BatchNormalization | input: | output: | (None, 256) | (None, 256)
batch_normalization_16 | BatchNormalization | input: | output: | (None, 32) | (None, 32)
dense_11 | Dense | input: | output: | (None, 32) | (None, 16)
dense_14 | Dense | input: | output: | (None, 256) | (None, 256)
dense_17 | Dense | input: | output: | (None, 32) | (None, 16)
batch_normalization_11 | BatchNormalization | input: | output: | (None, 16) | (None, 16)
batch_normalization_14 | BatchNormalization | input: | output: | (None, 256) | (None, 256)
batch_normalization_17 | BatchNormalization | input: | output: | (None, 16) | (None, 16)
Plasma_parameters_out | Dense | input: | output: | (None, 16) | (None, 9)
Geometry_out | Dense | input: | output: | (None, 256) | (None, 198)
Simulator_outcome_out | Dense | input: | output: | (None, 16) | (None, 8)

Figure C.2: The full decoder side of Model 1.0. The decoder takes a sample from the latent space and constructs reconstructions for all three data categories.

C.2 Model 1.1

The encoder of model 1.1 is shown in Figure C.3. The geometry and transport variables enter one neural network together ending in one latent space as can be seen in the left column. The right column is separated from the rest of the model to stop the information related to the transport to be stored in the plasma latent space at the end of the right column of Figure C.3.

The decoder of model 1.1 can be found in Figure C.4. The samples of both latent spaces together are used to reconstruct the transport values. These samples are also used to separately reconstruct the geometry and the plasma parameters.

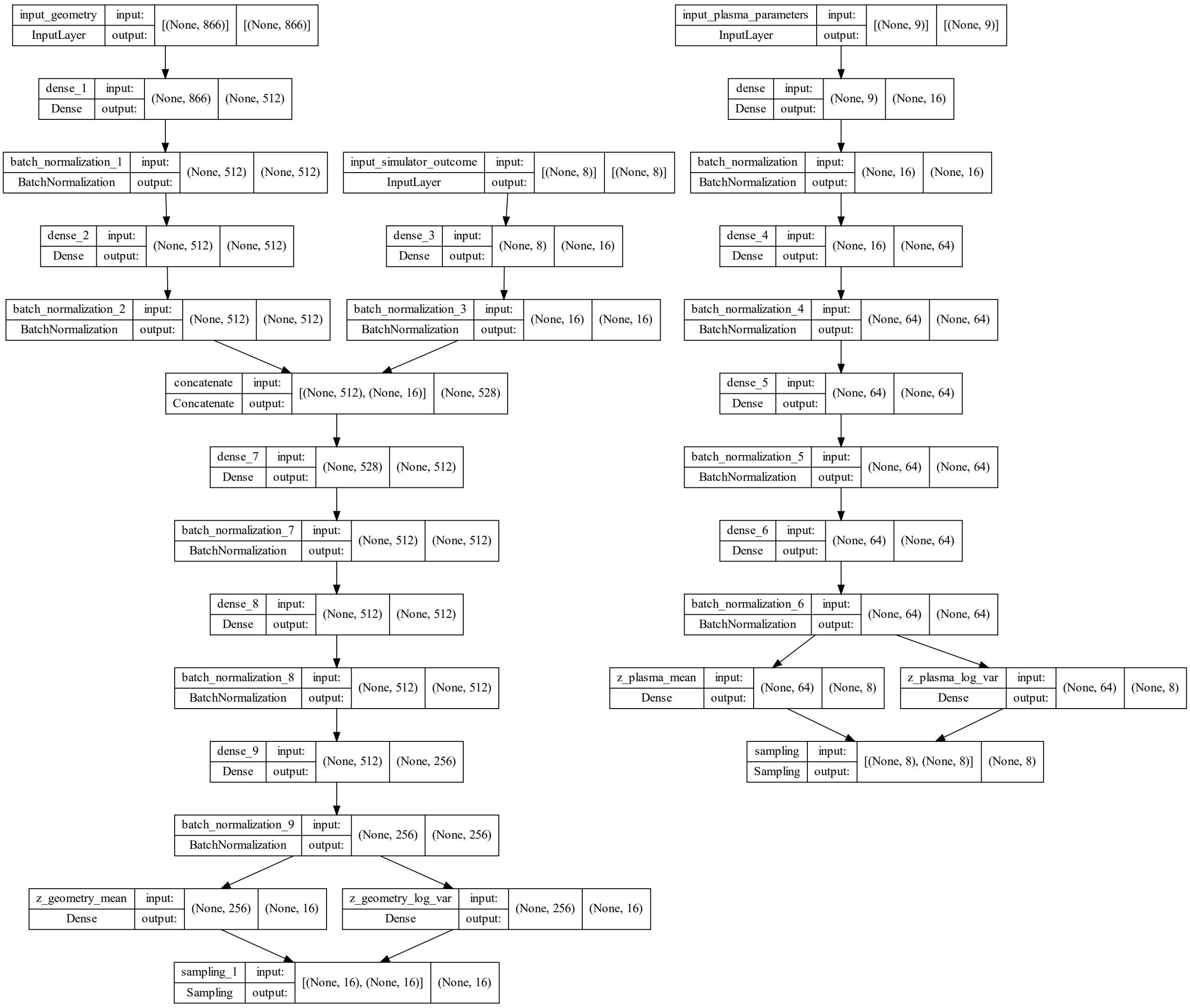


Figure C.3: The entire encoder side of Model 1.1 The input of this side is given by 3 input classes: geometry, simulator outcome and plasma parameters. The outcome of this side are the mean, ln($variance$) and a sample per latent space for all dimensions in that latent space.

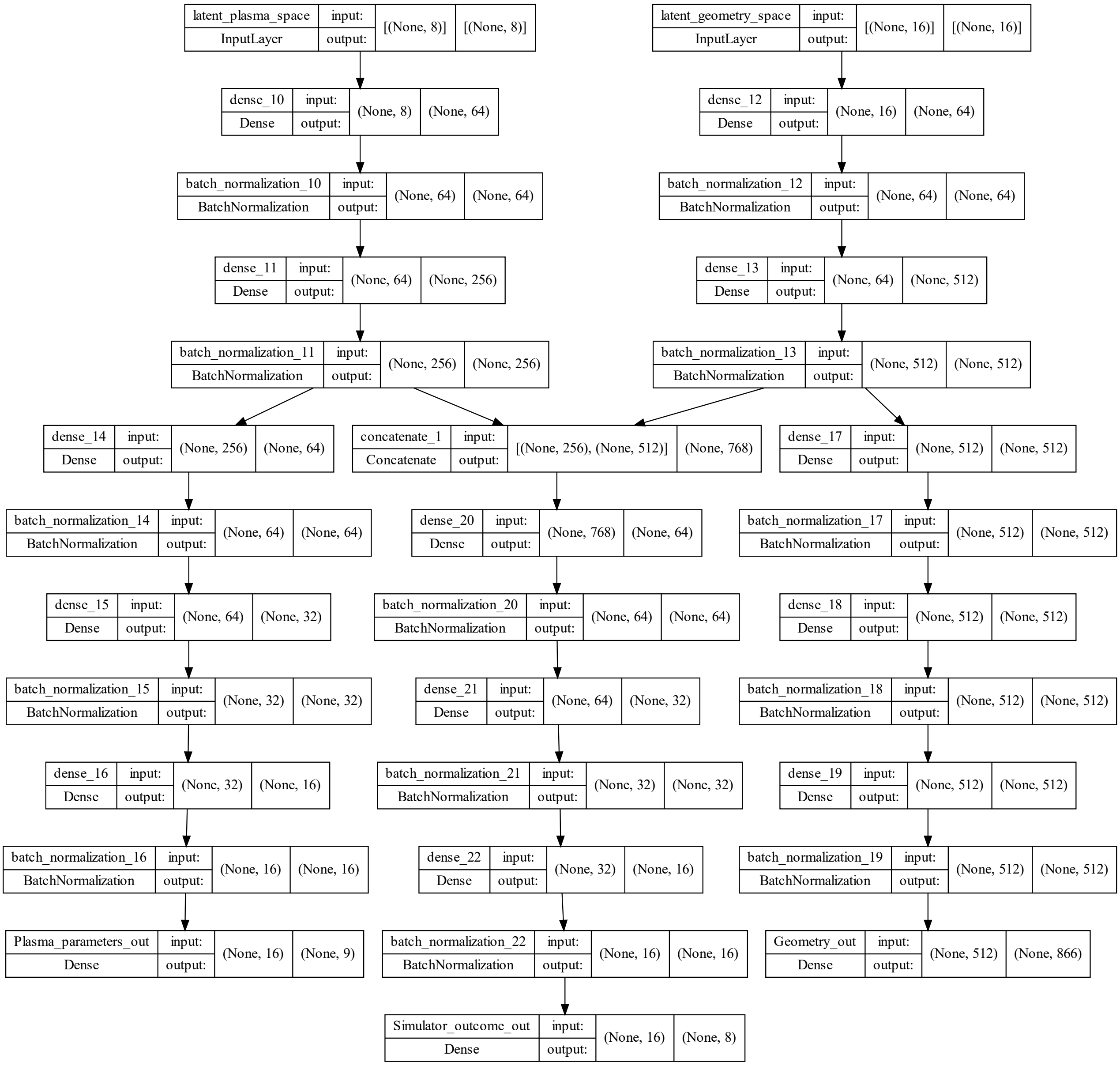
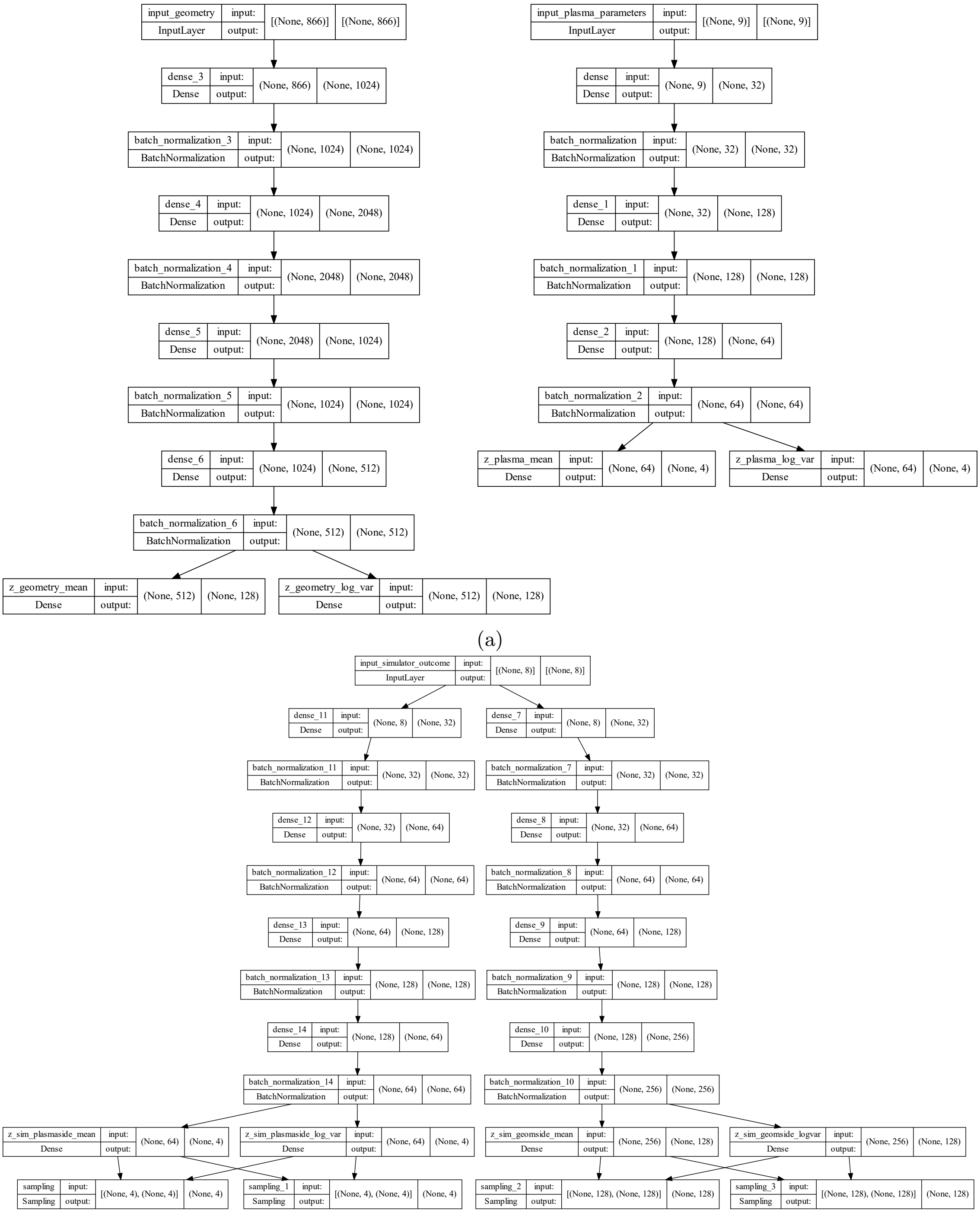

latent_plasma_space | InputLayer | input: | output: | [(None, 8)] | [(None, 8)]
latent_geometry_space | InputLayer | input: | output: | [(None, 16)] | [(None, 16)]
dense_10 | Dense | input: | output: | (None, 8) | (None, 64)
dense_12 | Dense | input: | output: | (None, 16) | (None, 64)
batch_normalization_10 | BatchNormalization | input: | output: | (None, 64) | (None, 64)
batch_normalization_12 | BatchNormalization | input: | output: | (None, 64) | (None, 64)
dense_11 | Dense | input: | output: | (None, 64) | (None, 256)
dense_13 | Dense | input: | output: | (None, 64) | (None, 512)
batch_normalization_11 | BatchNormalization | input: | output: | (None, 256) | (None, 256)
batch_normalization_13 | BatchNormalization | input: | output: | (None, 512) | (None, 512)
dense_14 | Dense | input: | output: | (None, 256) | (None, 64)
concatenate_1 | Concatenate | input: | output: | [(None, 256), (None, 512)] | (None, 768)
dense_17 | Dense | input: | output: | (None, 512) | (None, 512)
batch_normalization_14 | BatchNormalization | input: | output: | (None, 64) | (None, 64)
dense_20 | Dense | input: | output: | (None, 768) | (None, 64)
batch_normalization_17 | BatchNormalization | input: | output: | (None, 512) | (None, 512)
dense_15 | Dense | input: | output: | (None, 64) | (None, 32)
batch_normalization_20 | BatchNormalization | input: | output: | (None, 64) | (None, 64)
dense_18 | Dense | input: | output: | (None, 512) | (None, 512)
batch_normalization_15 | BatchNormalization | input: | output: | (None, 32) | (None, 32)
dense_21 | Dense | input: | output: | (None, 64) | (None, 32)
batch_normalization_18 | BatchNormalization | input: | output: | (None, 512) | (None, 512)
dense_16 | Dense | input: | output: | (None, 32) | (None, 16)
batch_normalization_21 | BatchNormalization | input: | output: | (None, 32) | (None, 32)
dense_19 | Dense | input: | output: | (None, 512) | (None, 512)
batch_normalization_16 | BatchNormalization | input: | output: | (None, 16) | (None, 16)
dense_22 | Dense | input: | output: | (None, 32) | (None, 16)
batch_normalization_19 | BatchNormalization | input: | output: | (None, 512) | (None, 512)
Plasma_parameters_out | Dense | input: | output: | (None, 16) | (None, 9)
batch_normalization_22 | BatchNormalization | input: | output: | (None, 16) | (None, 16)
Geometry_out | Dense | input: | output: | (None, 512) | (None, 866)
Simulator_outcome_out | Dense | input: | output: | (None, 16) | (None, 8)

Figure C.4: The full decoder side of Model 1.0. The decoder takes a sample from both latent spaces and uses these samples to reconstruct the input data. For the geometry and plasma data only one of the two samples is used while for the transport quantities the samples are combined.

C.3 Model 2

Model 2 uses the DIVA framework explained in Section 3.4. The transport quantities are encoded into both latent spaces using two encoders functioning independent from each other as can be seen in Figure C.5a. The transport quantities are reconstructed by one decoder functioning on a combination of a sample from both latent spaces, seen in Figure C.6. The latent spaces are shaped by two conditional priors based on the geometry and the plasma parameters, found in Figure C.5b. These conditions can be predicted using regression neural networks.


input_geometry InputLayer input: output: [(None, 866)] [(None, 866)]
dense_3 Dense input: output: (None, 866) (None, 1024)
batch_normalization_3 BatchNormalization input: output: (None, 1024) (None, 1024)
dense_4 Dense input: output: (None, 1024) (None, 2048)
batch_normalization_4 BatchNormalization input: output: (None, 2048) (None, 2048)
dense_5 Dense input: output: (None, 2048) (None, 1024)
batch_normalization_5 BatchNormalization input: output: (None, 1024) (None, 1024)
dense_6 Dense input: output: (None, 1024) (None, 512)
batch_normalization_6 BatchNormalization input: output: (None, 512) (None, 512)
z_geometry_mean Dense input: output: (None, 512) (None, 128)
z_geometry_log_var Dense input: output: (None, 512) (None, 128)
input_plasma_parameters InputLayer input: output: [(None, 9)] [(None, 9)]
dense Dense input: output: (None, 9) (None, 32)
batch_normalization BatchNormalization input: output: (None, 32) (None, 32)
dense_1 Dense input: output: (None, 32) (None, 128)
batch_normalization_1 BatchNormalization input: output: (None, 128) (None, 128)
dense_2 Dense input: output: (None, 128) (None, 64)
batch_normalization_2 BatchNormalization input: output: (None, 64) (None, 64)
z_plasma_mean Dense input: output: (None, 64) (None, 4)
z_plasma_log_var Dense input: output: (None, 64) (None, 4)
input_simulator_outcome InputLayer input: output: [(None, 8)] [(None, 8)]
dense_11 Dense input: output: (None, 8) (None, 32)
batch_normalization_11 BatchNormalization input: output: (None, 32) (None, 32)
dense_12 Dense input: output: (None, 32) (None, 64)
batch_normalization_12 BatchNormalization input: output: (None, 64) (None, 64)
dense_13 Dense input: output: (None, 64) (None, 128)
batch_normalization_13 BatchNormalization input: output: (None, 128) (None, 128)
dense_14 Dense input: output: (None, 128) (None, 64)
batch_normalization_14 BatchNormalization input: output: (None, 64) (None, 64)
z_sim_plasmaside_mean Dense input: output: (None, 64) (None, 4)
z_sim_plasmaside_log_var Dense input: output: (None, 64) (None, 4)
sampling Sampling input: output: [(None, 4), (None, 4)] (None, 4)
sampling_1 Sampling input: output: [(None, 4), (None, 4)] (None, 4)
dense_7 Dense input: output: (None, 8) (None, 32)
batch_normalization_7 BatchNormalization input: output: (None, 32) (None, 32)
dense_8 Dense input: output: (None, 32) (None, 64)
batch_normalization_8 BatchNormalization input: output: (None, 64) (None, 64)
dense_9 Dense input: output: (None, 64) (None, 128)
batch_normalization_9 BatchNormalization input: output: (None, 128) (None, 128)
dense_10 Dense input: output: (None, 128) (None, 256)
batch_normalization_10 BatchNormalization input: output: (None, 256) (None, 256)
z_sim_geomside_mean Dense input: output: (None, 256) (None, 128)
z_sim_geomside_logvar Dense input: output: (None, 256) (None, 128)
sampling_2 Sampling input: output: [(None, 128), (None, 128)] (None, 128)
sampling_3 Sampling input: output: [(None, 128), (None, 128)] (None, 128)

(a)

(b)

Figure C.5: The entire encoder side of Model 2. The input of this side is given by 3 input classes: geometry, simulator outcome and plasma parameters. Subfigure a is the encoder encoding the transport quantities into two separate latent spaces. Subfigurer b shows the two neural networks functioning as conditional priors for the geometry and the plasma parameters.

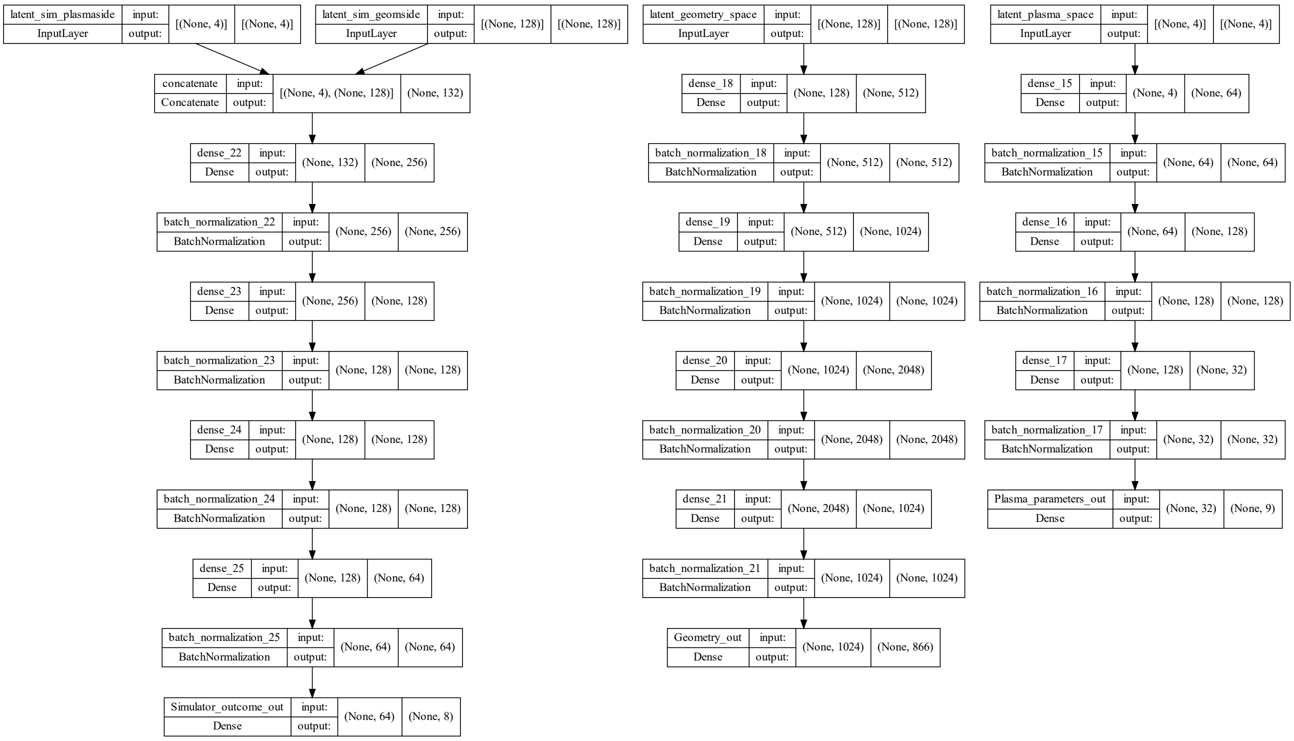

latent_sim_plasmaside | input: | [(None, 4)] | [(None, 4)]
InputLayer | output:
latent_sim_geomside | input: | [(None, 128)] | [(None, 128)]
InputLayer | output:
latent_geometry_space | input: | [(None, 128)] | [(None, 128)]
InputLayer | output:
latent_plasma_space | input: | [(None, 4)] | [(None, 4)]
InputLayer | output:
concatenate | input: | [(None, 4), (None, 128)] | (None, 132)
Concatenate | output:
dense_18 | input: | (None, 128) | (None, 512)
Dense | output:
dense_15 | input: | (None, 4) | (None, 64)
Dense | output:
dense_22 | input: | (None, 132) | (None, 256)
Dense | output:
batch_normalization_18 | input: | (None, 512) | (None, 512)
BatchNormalization | output:
batch_normalization_15 | input: | (None, 64) | (None, 64)
BatchNormalization | output:
batch_normalization_22 | input: | (None, 256) | (None, 256)
BatchNormalization | output:
dense_19 | input: | (None, 512) | (None, 1024)
Dense | output:
dense_16 | input: | (None, 64) | (None, 128)
Dense | output:
dense_23 | input: | (None, 256) | (None, 128)
Dense | output:
batch_normalization_19 | input: | (None, 1024) | (None, 1024)
BatchNormalization | output:
batch_normalization_16 | input: | (None, 128) | (None, 128)
BatchNormalization | output:
batch_normalization_23 | input: | (None, 128) | (None, 128)
BatchNormalization | output:
dense_20 | input: | (None, 1024) | (None, 2048)
Dense | output:
dense_17 | input: | (None, 128) | (None, 32)
Dense | output:
dense_24 | input: | (None, 128) | (None, 128)
Dense | output:
batch_normalization_20 | input: | (None, 2048) | (None, 2048)
BatchNormalization | output:
batch_normalization_17 | input: | (None, 32) | (None, 32)
BatchNormalization | output:
batch_normalization_24 | input: | (None, 128) | (None, 128)
BatchNormalization | output:
dense_21 | input: | (None, 2048) | (None, 1024)
Dense | output:
Plasma_parameters_out | input: | (None, 32) | (None, 9)
Dense | output:
dense_25 | input: | (None, 128) | (None, 64)
Dense | output:
batch_normalization_21 | input: | (None, 1024) | (None, 1024)
BatchNormalization | output:
batch_normalization_25 | input: | (None, 64) | (None, 64)
BatchNormalization | output:
Geometry_out | input: | (None, 1024) | (None, 866)
Dense | output:
Simulator_outcome_out | input: | (None, 64) | (None, 8)
Dense | output:

Figure C.6: The total decoder side of Model 2 including the auxiliary regression neural networks. The left column is the decoder for the transport quantities where two samples, one from each latent space, are combined to be reconstructed. The two right columns are two regression neural networks to provide predictions for the geometry and the plasma parameters for a given sample.

D Latent space exploration

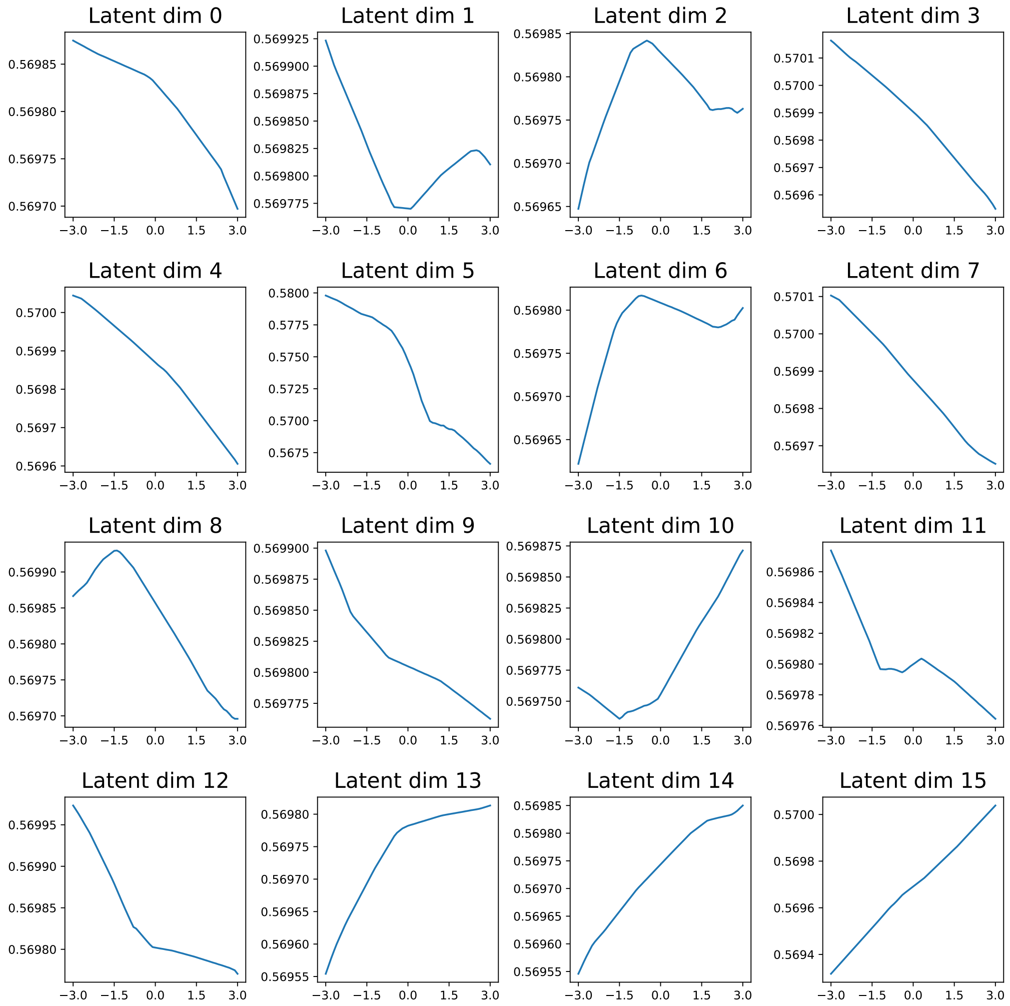


Figure D.1: The scaled value of the neoclassical particle flux density along one latent space dimension from -3 to 3 while keeping the value of the other dimensions fixed at 1. for the plasma latent space a random sample is picked from a Gaussian but within 1σ. The plasma sample used is the same as in Figure 7.6

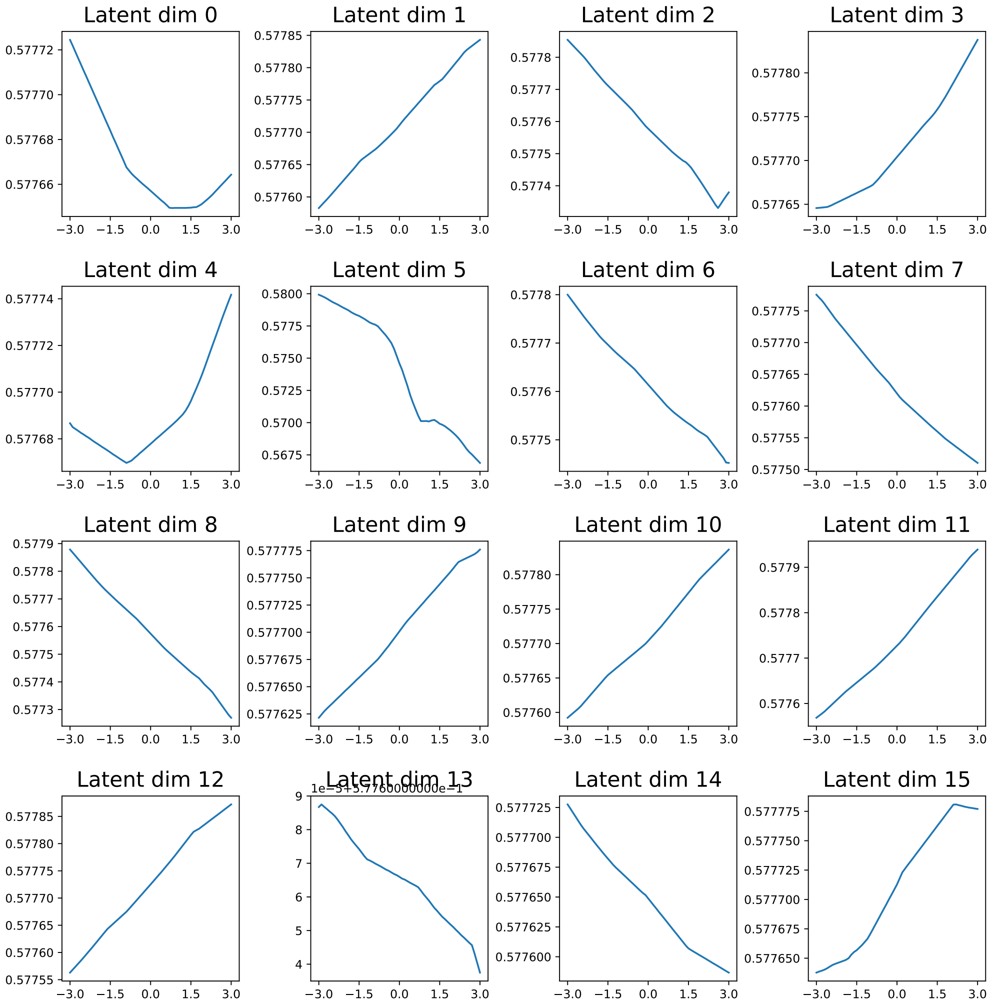


Figure D.2: The scaled value of the neoclassical particle flux density along one latent space dimension from -3 to 3 while keeping the value of the other dimensions fixed at -1. for the plasma latent space a random sample is picked from a Gaussian but within 1σ. The plasma sample used is the same as in Figure 7.6.