Applications of deep generative models to Tokamak Nuclear Fusion

Nieuwenhuizen, Daan M.

Award date:
2021

Eindhoven University of Technology

Faculty of Applied Physics

Applications of Deep Generative Models to Tokamak Nuclear Fusion

Master Thesis

D.M. Nieuwenhuizen

Supervisors:	prof. dr. ir. J.M.V.A. Koelman
	dr. V. Menkovski
	dr. J. Citrin
	ir. K.L. van de Plassche

February 15, 2021

Abstract

Integrated simulation for output profiles of a nuclear fusion reactor is generally too slow, when using first-principle-based modeling, to be useful for real-time plasma dynamics control purposes. Neural networks hold the promise of delivering surrogate models that are sufficiently fast to operate in real-time. Here we present the results of using a deep generative model as a neural network alternative for integrated simulation suites. First, by training a conditional variational auto-encoder on a simple synthetic dataset, the quality of generated data is investigated, yielding high-quality predictions with a relative error of only 1.38% on a testing set. We also examine the role that the latent space plays in this generation, and find it to effectively take the role of all relevant parameters that are the source of variation between the samples. Then, we introduce the more powerful regressive disentanglement variational auto-encoder (ReD-VAE), which is applied to a much larger and complex experimental dataset. By introducing an additional loss term, the quality of generated data is improved significantly with respect to a normal VAE. This leads to excellent performance in the prediction of known scaling laws, which are used to validate the correctness of model predictions. Other usages of ReD-VAE are also presented, including statistical inference of experimental parameters with good accuracy and improved plasma mode classification. A proof-of-principle is provided of using the latent space for unsupervised hidden parameter discovery. Plasma mode is expected to be the main hidden parameter (to the model), which can be reconstructed with 86% accuracy in an unsupervised fashion, showing that this model can potentially be used to gain further insights into what other hidden parameters influence tokamak performance.

Contents

1 Introduction

1.1 Motivation

In the ongoing energy transition, nuclear fusion energy holds a strong promise for the more distant future. Nuclear fusion has the potential to provide a clean source of vast amounts of energy, with practically no limits on the sustainability, since most of the resources can be gained from water [1].

However, at this point in time, fusion is not yet profitable. While the techniques have high potential, existing fusion reactors require more energy to run than the amount of energy that is output by the reactor, resulting in a net energy loss. Lots of research is currently being done to increase the efficiency of fusion reactors, including large international projects such as the ITER project [2].

In these projects, it is not feasible to base work solely on experimental investigations. The cost of an experiment is far too great, and therefore simulation software has been developed to provide complementary avenues of research. Traditionally, this takes the shape of integrated modeling software, where complex differential equations are numerically integrated. This process is computationally very expensive. In order to speed up these simulations, all sorts of approximations have to be made.

Another way to speed up simulations is by using the recent advancements that have been made in machine learning. It has already been shown that neural networks can yield tremendous speed-ups in multi-physics tokamak simulation through the development of surrogate networks [3], indicating the potential that these methods harbor. However, in these publications, only *discriminative* models have been used. A discriminative neural network models the probability distribution of observing some variable $\mathbf{x}$ given an input $\mathbf{y}$[1], i.e. $\mathbb{P}(\mathbf{x} \mid \mathbf{y})$.

We believe that a lot more can be learned about tokamak nuclear fusion, by using neural networks to create a *generative* model. A generative model is more powerful than a discriminative model, because it models the joint distribution of input and output, as well as some intermediate latent representation, $\mathbf{z}$. This results in a model for the joint distribution $\mathbb{P}(\mathbf{x}, \mathbf{y}, \mathbf{z})$. The latent representation is designed to contain all information that is necessary for a good description of an experimental output profile, and thus captures the role of hidden parameters. The generative model allows for several novel applications, among which:

- Fast simulation of fusion data, allowing for quick prediction of fusion performance for a wide range of parameter values.
- Convenient generation of realistic synthetic data for physics model testing and validation, and controlled parameter space sampling for the creation of surrogate model training datasets.
- Quantification of the uncertainty in results induced by inaccuracy of the base conditions.

[1]By physicists' standards, this choice of $\mathbf{x}$ and $\mathbf{y}$ might be counter-intuitive. It is, however, standard convention in the field of machine learning, where $\mathbf{y}$ generally represents a label or condition that corresponds to an input $\mathbf{x}$.

- Prediction of most likely tokamak input parameters based on some given output.
- Discovery of hidden variables using the latent space.

In this work, we first present a proof-of-principle, demonstrating the modeling capabilities of a simple variational auto-encoder (VAE) on a simulated dataset. By obtaining results in a clean, synthetic environment, a first look into the applications of such a model can be obtained. Subsequently, a more powerful model, the regressive disentanglement variational auto-encoder (ReD-VAE), is introduced. This model is used on experimental data from the JET tokamak, where we get more insight into the exciting possibilities of this generative model.

1.2 Research Questions

In the first part of this work, we implement the generative model using a conditional variational auto-encoder. The goal here is to show that the CVAE has the modeling power to capture the distribution of the data. This leads to the first research question:

Can a conditional variational auto-encoder be used to generate synthetic results similarly to conventional simulation methods?

Some sub-questions are relevant here:

- What is the role of the latent variables?
- How accurate are the generated predictions?

Once success is obtained in this first part, we also want to explore the new applications that a generative model brings to the table. After all, the goal of this project is to not only gain the same results in a faster way, but also to gain new insights into the physics of a nuclear fusion reactor. This leads to the second research question:

Which new applications are possible after training a generative model on a complex, experimental dataset?

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

- What generative model structure allows for the desired modeling capabilities:
  - The generation of realistic output profiles, where some of the input parameters can be unknown.
  - The estimation of the variance in output profiles, for a given set of input parameters.
- How can this model be optimized to perform well for data generation?
- Can the model reproduce known trends, either via comparison to literature or to data?
- Which information is contained in the latent space, and how can this information be used to gain insight into hidden parameters?

1.3 Contributions

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

- Introduction of the ReD-VAE model, a generative model specifically optimized for high-quality data generation of experimental profiles given some input parameters.
- Providing proof-of-principle methods for different applications of generative models in physical sciences, including:
  - Generation of samples for different experimental conditions, with corresponding variance, for the optimization of tokamak performance.
  - Discovery of hidden variables, relevant to the performance of the fusion reactor.
  - Prediction of most likely input parameters for a set of experimental results.
  - Improved state classification of plasma.

This work should be viewed mainly as a proof-of-principle, where some of the possible applications of these models are highlighted. A full in-depth analysis of these methods and the new insights that can be obtained is left for later work.

1.4 Overview

The rest of this report will start with a short introduction of the basics of nuclear fusion and the relevant tokamak parameters in section 2. Then, in section 3, the relevant machine learning background that is used is discussed, followed by an overview of literature related to the current work in section 4. Section 5 introduces the ReD-VAE model, followed by a description of the datasets in section 6. Sections 7 and 8 give experimental results for the simulated dataset and the experimental data respectively, as well as a dive into the potential applications of the ReD-VAE model. Section 9 discusses the strengths and weaknesses of the methods that have been applied, and the thesis will be concluded in section 10.

2 Nuclear Fusion Background

2.1 Basics

Two promising methods of nuclear fusion, are either using a deuterium-tritium reaction, or a reaction with two deuterium particles. Deuterium (D) and tritium (T) are both isotopes of hydrogen, with one proton and one, respectively two neutrons. During a typical fusion reaction, deuterium and tritium fuse to form helium and a neutron:

$$ {}^{2}_{1}D + {}^{3}_{1}T \rightarrow {}^{4}_{2}He + n^{0} \tag{2.1}$$

This exothermic reaction releases a large amount of energy $E = 17.59$ MeV [4]. The origin of this energy is that the resulting mass of the Helium particle and neutron is lower, compared to the incoming deuterium and tritium particles. As per Einstein's $E = mc^2$, the mass difference m is converted into energy with a multiplication factor of the speed of light, c, squared. The fusion power that is generated on a larger scale is given by equation (2.2), where n_A and n_B are the densities of the two components of the reaction [5].

$$P_{fus} = n_A n_B \langle \sigma v_{A,B} \rangle E \tag{2.2}$$

In this equation, $\langle \sigma v_{A,B} \rangle$ is the cross-section of the reaction, averaged over all energies for the particles of species A and B. This cross-section can be regarded as the probability for the reaction to happen.

For low energy particles, the fusion cross-section is negligible, as the energy in the particles is too low to overcome the Coulomb repulsion between the particles, preventing them from fusing. Therefore, particles with high energy (temperature) are required. As the gases are heated such that particles obtain these temperatures, they transition to the plasma state, breaking up atoms and molecules in the gas into electrons and ions. As the plasma is further heated, the electrons and ions gain more energy. This is beneficial, as a higher temperature increases the cross-section (likelihood) of a fusion reaction (up to a certain point where the cross-section decreases again) [6]. For this reason, aside from the particle density, the particle temperatures are very important inputs in determining the expected fusion power. This also means that for optimal fusion power, high particle densities need to be accompanied by high particle temperatures. To achieve this, multiple types of fusion reactors have been suggested. Of these, a tokamak reactor appears to have the most potential.

2.2 Tokamak Reactor

2.2.1 Magnetic Fields Design

In a tokamak, the plasma is contained in a toroidal reactor. A schematic overview of such a reactor is given in Figure 1.

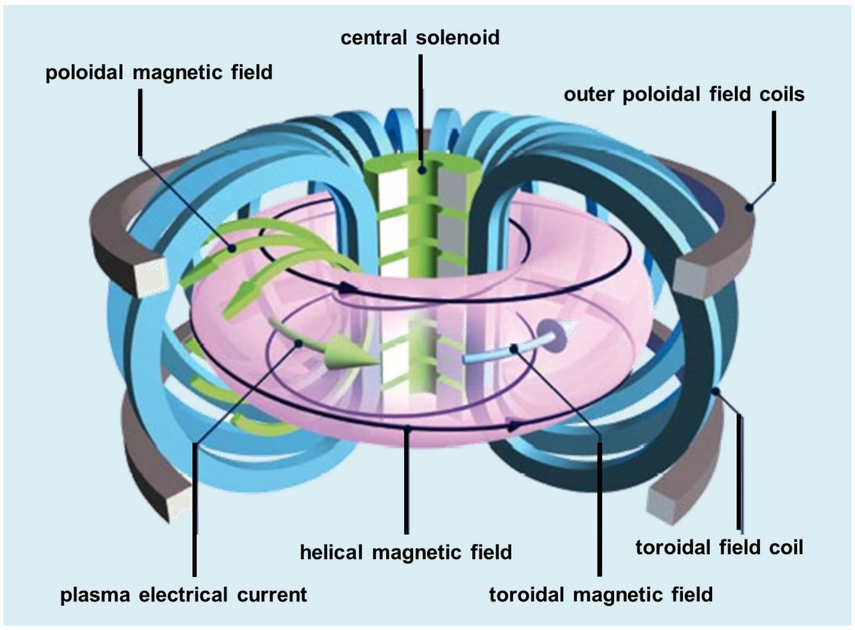


Figure 1 – Schematic overview of the magnetic field design of a tokamak reactor [7].

To maintain high temperatures, the plasma cannot be in contact with the reactor walls. To avoid excessive temperature loss, the plasma is confined by magnetic fields. The toroidal field coils induce a toroidal magnetic field B_t. The central solenoid induces an electrical current I_p in the plasma, as well as a poloidal magnetic field. The combination of these toroidal and poloidal fields makes it so that the charged particles follow helical trajectories around the torus.

In Figure 2 a schematic cross-section of the tokamak reactor is given. A few important concepts are highlighted in this image, which serves to illustrate the shaping of the magnetic surfaces. The most important magnetic surface here is the last closed flux surface (LCFS), or separatrix, which is the last surface where the flux lines form a closed loop. Let R_{max} and R_{min} be the maximum and minimum major tokamak radius of the LCFS and let Z_{max} and Z_{min} be the maximum and minimum height of the LCFS. The minor radius a of the plasma is then defined as

$$a = \frac{R_{max} - R_{min}}{2} \tag{2.3}$$

which leads to the definition of the elongation κ as

$$\kappa = \frac{Z_{max} - Z_{min}}{2a} \tag{2.4}$$

The geometric major radius is then the middle of the LCFS, given by:

$$R_{geo} = \frac{R_{max} + R_{min}}{2} \tag{2.5}$$

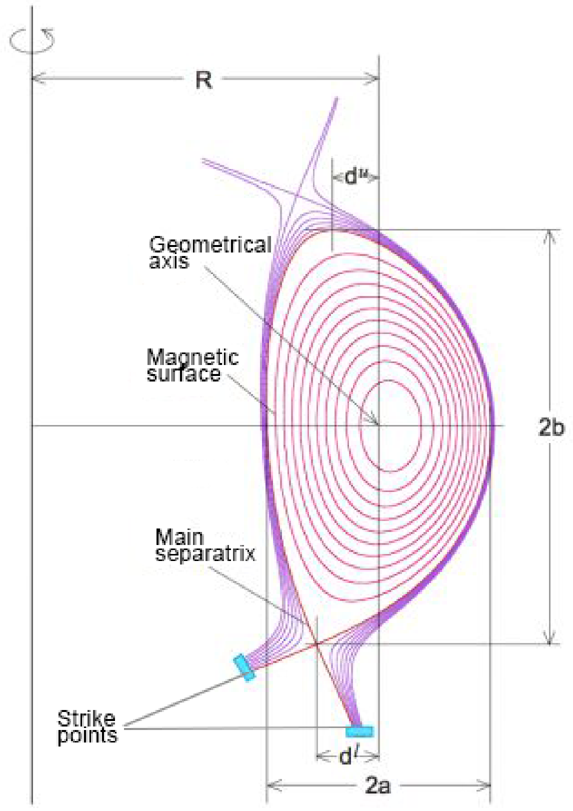


Figure 2 – Cross-section of the tokamak reactor, showing the plasma shaping parameters. Image adapted from [8].

Now, let R_{upper} and R_{lower} be the major radii of the highest and lowest points in the LCFS. The upper and lower triangularities δ_u and δ_l are then defined as follows:

$$\delta_u = \frac{R_{geo} - R_{upper}}{a} \tag{2.6}$$

$$\delta_l = \frac{R_{geo} - R_{lower}}{a} \tag{2.7}$$

The LCFS field lines collide with the divertor plates in the inner and outer strike point. These points are characterized by their radial coordinate R and height Z.

In this work, radial profiles for electron temperature, electron density, and ion temperature are provided. The radial profiles are given with respect to the normalized radius ρ, defined as in equation (2.8). Here, ψ_t is the toroidal magnetic flux associated with the radial point, and ψ_{LCFS} is the toroidal flux through the area of the separatrix.

$$\rho = \sqrt{\frac{\psi_t}{\psi_{LCFS}}} \tag{2.8}$$

This square-root expression can be considered a normalized radius because the magnetic flux generally increases with the squared radius.

In this work, only plasma profiles in the core region inside the LCFS are considered. Because plasma transport in this region is much faster along magnetic field lines than perpendicular to them, it is assumed that the plasma parameters exhibit both axisymmetry, as well as poloidal symmetry. Consequently, the profiles have only radial dependence. Therefore, a one-dimensional profile is enough to describe the entire reactor for a given point in time. The analysis of two-dimensional profiles outside the LCFS region, which includes the plasma-wall interaction, is out of the scope of this work.

The parameters that have been discussed can be adjusted in a control room, in order to contain as many particles in the plasma as possible while it's being heated. There are multiple methods of heating the plasma, which will be discussed next.

2.2.2 Plasma Heating

Three forms of plasma heating are relevant in this thesis.

Ohmic Heating Ohmic heating (P_{ohm}) is the heating due to the electrical current I_p in the plasma. In any electrical conductor, running a current through a material generates heat, due to the resistance of that material. This also holds in the case of plasmas. The Ohmic heating can be calculated by using the local Ohmic current density $j_{ohm}(\rho)$, as given in equation (2.9).

$$P_{ohm} = \int \eta(\rho) j_{ohm}^2(\rho) \, \mathrm{d}V \tag{2.9}$$

Here, $\eta(\rho)$ is the resistivity of the plasma and the integration is done over the entire volume V within the LCFS. To do this calculation, this volume is approximated as a cylinder with length $2\pi R_0$, where R_0 is the major radius of the torus. Since poloidal symmetry is assumed, the integral then reduces to equation (2.10).

$$P_{ohm} = 4\pi^2 R_0 \int_0^1 \eta(\rho) j_{ohm}^2(\rho) \rho \, \mathrm{d}\rho \tag{2.10}$$

Because of the discrete nature of the used data, the integral is in this work approximated by a summation:

$$P_{ohm} \approx 4\pi^2 R_0 \sum_i \eta(\rho_i) j_{ohm}^2(\rho_i) \rho_i \, \Delta\rho \tag{2.11}$$

Where the summation is over all radial points ρ_i within the LCFS, and $\Delta\rho$ is the distance between two of these points. The plasma resistivity $\eta(\rho)$ is approximated by the Spitzer resistivity [9], given by equation (2.12).

$$\eta = \frac{\sqrt{2}}{12\pi^{3/2}} \frac{Z_{eff} e^{1/2} m_e^{1/2} \ln \Lambda}{\epsilon_0^2 T_e^{3/2}} F(Z_{eff}) \tag{2.12}$$

In this equation, Z_{eff} is the effective charge, which is a radial profile in the plasma, e is the elementary electron charge, m_e is the electron mass, ϵ_0 is the vacuum permittivity and T_e is the electron temperature in electronvolts (eV), again a radial profile. Furthermore, we use

$$F(Z) \approx \frac{1 + 1.198Z + 0.222Z^2}{1 + 2.966Z + 0.753Z^2} \approx 1.96 \tag{2.13}$$

for $Z = 1$, and $\ln\Lambda$ is the Coulomb logarithm, given by equation (2.14) [10].

$$\ln\Lambda = 15.2 - \frac{1}{2}\ln(n_e \cdot 10^{-20}) + \ln(T_e \cdot 10^{-3}) \tag{2.14}$$

In this equation, n_e is the density of electrons in m^{-3}.

Because plasma resistivity drops as the plasma temperatures increase, as seen in (2.12), Ohmic heating is mainly effective as an initial heat source [11]. To further heat the plasma past a certain temperature, other heating methods will also be necessary.

Neutral Beam Injection In Neutral Beam Injection (NBI), beams of neutral particles are shot into the plasma [12]. The beams consist of high-energy atoms. Once inside the plasma, these atoms collide with the plasma particles, transferring their energy into the plasma, thus heating the plasma. The power that is supplied in this manner is indicated as P_{NBI}.

Ion Cyclotron Resonance Heating The final form of heating discussed here is Ion Cyclotron Resonance Heating (ICRH). Mainly the majority ions are heated, rather than the electrons [13]. The ions are heated by applying an electromagnetic wave to the plasma, at a resonance frequency of the majority ion species. The power that is supplied to the plasma in this method will be referred to as P_{ICRH}.

All these forms of heating contribute to the energy in the plasma. The *confinement time* is a measure for how well this energy is retained in the plasma, which will be discussed next.

2.2.3 Confinement Time

The confinement time is a measure of how long stored energy remains in a plasma when all heating sources are stopped. To estimate the confinement time, the thermal stored energy W is calculated. This consists of a volumetric integral over the density of charged particles, multiplied by their temperatures. Since poloidal symmetry is assumed, only the radial part of the integral remains. The thermal stored energy W is given in equation (2.15), where the integral is again approximated by a sum for discrete intervals.

$$W = \int \frac{3}{2}\, n_e e\; (T_e + T_i)\; \rho \, \mathrm{d}\rho \tag{2.15}$$

$$\approx \frac{3}{2} \sum_i n_e(\rho_i)\, e\; (T_e(\rho_i) + T_i(\rho_i))\, \rho_i \, \Delta\rho \tag{2.16}$$

Where T_i is the ion temperature in eV and e is the elementary charge. Applying more heating is an easy way of getting more energy in a plasma. This is however not a result that we are interested in, our interest rather lies in how much of this energy is actually retained in the plasma. In order to get an estimate for this, the thermal stored energy is often scaled with the input power, to obtain the confinement time τ.

$$\tau = \frac{W}{P_{ohm} + P_{NBI} + P_{ICRH}} \tag{2.17}$$

To optimize the performance of a tokamak reactor, we seek to maximize the confinement time. This is very much dependent on the *state* that the plasma is in. Generally, plasmas are in L-mode. Once a required input power threshold is passed, the plasma can undergo a sudden transition to H-mode, where confinement times are typically two to three times longer [14]. An H-mode plasma is characterized by the formation of an edge pedestal region in the temperature and density profiles. A pedestal is a sharp increase in the radial profile for a quantity. An example of a plasma with a pedestal region is given in Figure 3. Here, a sharp change in electron temperature in the edge region between $\rho = 0.9$ and $\rho = 1.0$ is observed. This sudden increase in the edge region leads to higher temperatures overall, and thus a higher confinement time.

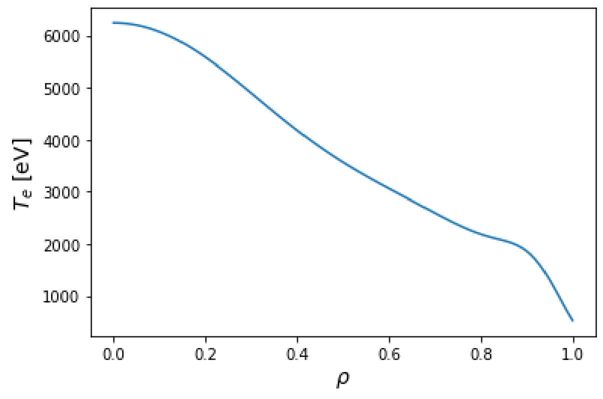


Figure 3 – Example of the electron temperature profile for an H-mode plasma. In the edge region, between $\rho = 0.9$ and $\rho = 1.0$, a pedestal is formed.

In order to obtain estimates of the expected confinement time for a fusion reactor, empirical scaling laws have been composed. In H-mode, a widely used scaling law is given by equation (2.18), known as the ITERH-98P(y,2) scaling law [15].

$$\tau_{98y2} = 0.0562 I_p^{0.93} B_t^{0.15} \overline{n_e}^{0.41} P^{-0.69} M^{0.19} R_0^{1.97} \kappa^{0.78} \epsilon^{0.58} \tag{2.18}$$

Here, $\overline{n_e}$ is the radially averaged electron density, P is the absorbed power and M is the hydrogen isotope mass in atomic mass units (2u for deuterium, 3u for tritium). The other parameters are specific to the tokamak, where κ is the elongation of the LCFS and ϵ is the inverse aspect ratio a/R_0. These values are approximated as constants, because this work only uses JET data where R_0 and ϵ are fixed, and κ has a limited range in practice. For the analysis presented in this work, the values $R_0 \approx 2.96$, $\kappa \approx 1.68$ and $\epsilon \approx 0.33$ are used [16].

In L-mode, a slightly different scaling law, known as ITERL-96P(th), is used [17]. This L-mode scaling law is given in Equation 2.19.

$$\tau_{96th} = 0.23 I_p^{0.96} B_t^{0.03} \overline{n}_e^{0.40} P^{-0.73} M^{0.20} R_0^{1.83} \kappa^{0.64} \epsilon^{-0.06} \tag{2.19}$$

These scaling laws can be used to validate findings in our models. In both scaling laws, I_p is given in MA, the average density in 10^{19} m^{-3} and the absorbed power in MW. In this work, the absorbed power is approximated as simply being $P_{NBI} + P_{ICRH}$. As the confinement time and expected fusion power are dependent on the particle densities, the amount of gas in the reactor has a substantial effect on tokamak performance. The amount of particles in the reactor is controlled by two variables: the gas input rate g_r, giving the number of particles that are injected per second, and the total amount of particles that are injected, g_d.

2.3 Tokamak Simulation

As mentioned earlier, simulation software is used to predict tokamak performance for different parameter values. Current simulation of tokamak fusion is carried out using JETTO [18]. This simulation suite, used to calculate the plasma evolution, uses Runge-Kutta and finite difference methods to solve ordinary and partial differential equations. This results in multi-physics simulation, with self-consistent determination of turbulence, collisional transport, radiation, heating and particle sources, and the evolution of one-dimensional temperature and density profiles. These simulations can take days to make a full plasma evaluation, depending on the desired complexity.

To increase the speed of calculations, quasi-linear gyrokinetic models are used, such as QuaLiKiz [19]. Furthermore, neural networks can be used as surrogate models that accelerate the bottleneck physics, speeding up the calculation considerably [3].

3 Machine Learning Background

3.1 Neural Networks

The models used in this work are mostly implemented using artificial neural networks. A short introduction to these models will be given here.

3.1.1 Artificial Neuron

An artificial neural network (NN) consists of many interconnected artificial neurons, also known as nodes. A neuron takes a number k of inputs x_i, where each input is weighted by a factor w_i. A bias b is then applied, and by using a non-linear activation function f, the output y of this neuron is determined [20]. This is summarized in equation (3.1), and shown schematically in Figure 4, for $k = 3$.

$$y = f\left(\sum_{i=0}^{k} w_i x_i + b\right) \tag{3.1}$$

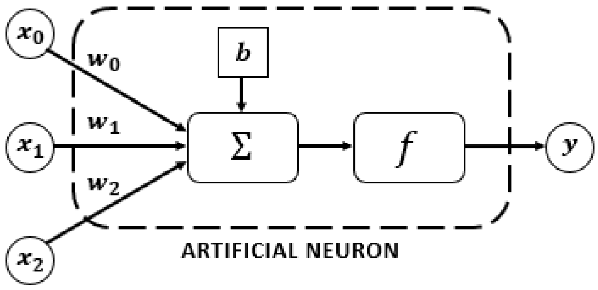


Figure 4 – Schematic overview of the components of an artificial neuron with three inputs x_i, that are weighted by independent weight parameters w_i. These weighted inputs are summed, along with a bias parameter. The final output y is determined by the activation function f.

An NN is created by stacking layers consisting of artificial neurons. By adjusting the behaviour and connectivity of a layer, different types of layers can be created that have different functionalities. The types that are most relevant in this work will be discussed in the next section. Because of the complex structures that can be created by combining layers, neural networks can model complex phenomena. To do this, the networks need to be trained, which means that the weights and biases of the neurons are optimized. This will be shortly discussed in section 3.1.3.

3.1.2 Layers

Dense Layer The most basic layer in a neural network is a densely connected layer, or just dense layer. In a dense layer, all neurons are connected to each neuron in the previous layer.

This can lead to a high number of trainable parameters. When there are n_1 nodes in the first layer and n_2 in the second layer, $n_1 n_2$ weight parameters are created. This makes the dense layer very powerful, but combining a lot of dense layers with many nodes can lead to long training times.

Convolutional Layer A convolutional layer is used when there is some positional relation between input data points, for example in a photograph, where the location of each of the pixels and the connection with neighboring pixels contain important information [21]. This information is then down-sampled into a more low-dimensional representation, by using filters with a specific size.

Deconvolutional Layer A deconvolutional layer, or transposed convolutional layer, essentially performs the inverse action from the convolutional layer [22]. These layers are used for upsampling, where the goal is to increase the dimensionality of some summary information.

Dropout Layer Dropout layers were created to prevent overfitting of neural networks, where the performance of a model on unseen testing data is substantially worse than on the supplied training data [23]. To prevent this, random nodes in the dropout layer are disabled during a training epoch. The number of nodes that are dropped out is dependent on the dropout-ratio. During inference, all nodes are used normally.

3.1.3 Neural Network Training

Loss Function To train a neural network, some goal has to be set, that can either be minimized or maximized, by changing the weights and bias parameters of the neurons. This goal is the loss function $\mathcal{L}$. Often, neural networks are trained in *supervised* fashion, meaning that they are optimized for the training input x with respect to some label y. *Unsupervised* training is another possibility, where the labels y are not used during training. *Semi-supervised* training is a mix of both methods, where for some samples the labels are used, while others are used for training without the label.

Optimization Neural networks are trained using gradient backpropagation. During training, the gradient of the loss is calculated for each of the trainable parameters of the model. Then using some form of gradient descent, the trainable parameters are updated such that the loss is optimized. For the models in this work, the Adam algorithm [24] is applied for optimization. A neural network is trained for a number of *epochs*, where each epoch consists of a pass through the training data that uses each sample once.

3.2 Generative Model

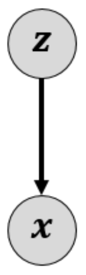
As mentioned in section 1, the aim of this project is to implement a generative model. Given some conditions $\mathbf{y}$ and output $\mathbf{x}$, a regressive model will model the probability distribution $\mathbb{P}(\mathbf{x} \mid \mathbf{y})$. A generative model describes how data is generated. To do that, a generative model will try to learn the joint probability $\mathbb{P}(\mathbf{x}, \mathbf{y})$ of the training set. This represents a powerful model, as this allows for several applications not feasible for discriminative models.

One such application is to use the joint probability as is. By sampling from this joint probability, data can be generated for a random combination of $\mathbf{x}$ and $\mathbf{y}$, that is realistic according to the training data. However, by statistical inference, the conditional probabilities can be calculated as well, not only $\mathbb{P}(\mathbf{x} \mid \mathbf{y})$, but also $\mathbb{P}(\mathbf{y} \mid \mathbf{x})$.

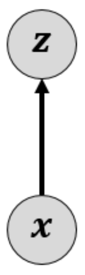
These distributions lead to many usages. In addition to this, because probability distributions are used, rather than point-wise predictions, estimates of the uncertainty in the predictions can be obtained. This is important because we want to know how certain a model is of its predictions. Otherwise, it's hard to draw decisive conclusions from an output. The uncertainty in a certain prediction can also be an indication of the stability of the output for a given set of conditions.

3.3 Latent Variable Model

The generative model that is implemented in this work, is a latent variable model. In a latent variable model, complex outputs $\mathbf{x}$ are assumed to be determined by a smaller number of variables, called *latent variables*. In the generative model, these latent variables play an important role. Graphical models for both the generative process as well as the inference process are given in Figure 5. In a graphical model, an arrow indicates a dependence. These simple graphs thus indicate that $\mathbf{x}$ and $\mathbf{z}$ are dependent on one another, the direction being dependent on the process that we are in.



(a) Generative



(b) Inference

Figure 5 – Graphical model when only the latent space is included.

The latent variables are a surrogate for every variable that we cannot (or haven't) measure accurately. Generally, there are also variables that we *do* know, or for which we want to examine the effect of changing its value. In this case, these conditions $\mathbf{y}$ can be added to the model. This will be discussed in Section 3.4.3.

3.4 Variational Auto-Encoder

In this work, a generative network is implemented by making use of a variational auto-encoder (VAE), that was first introduced in [25]. A VAE is a probabilistic version of a regular auto-encoder, which will be discussed first.

3.4.1 Auto-Encoder

An auto-encoder is a set of two connected neural networks, an *encoder* and a *decoder*. The encoder takes an input $\mathbf{x}$. In this work, the input will be a radial profile for tokamak quantities of interest, such as ion temperature T_i, or electron density n_e. The encoder will convert the input to a point $\mathbf{z}$ in a latent space.

In the next step, $\mathbf{z}$ serves as the input for the decoder. The decoder operation is inverse to that of the encoder and provides a reconstruction of the input $\hat{\mathbf{x}}$. During training, the goal is to minimize the difference between $\mathbf{x}$ and $\hat{\mathbf{x}}$. This requires optimization of both the encoder and the decoder.

A schematic overview of an auto-encoder is given in Figure 6.

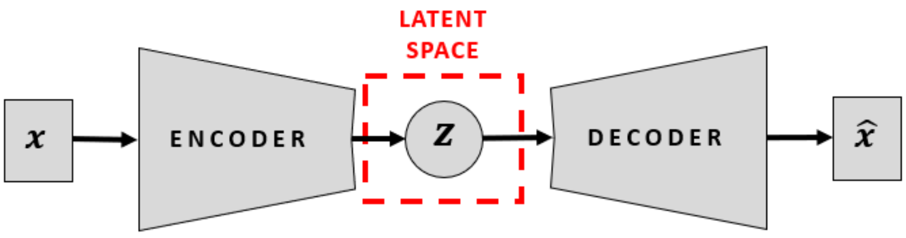


Figure 6 – Schematic overview of an auto-encoder. The encoder and decoder are implemented using neural networks.

3.4.2 VAE

A VAE is similar to an auto-encoder, but with a probabilistic touch. Instead of the encoder giving a point estimate $\mathbf{z}$ in the latent space, the encoder outputs two vectors $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$. These give an estimated mean and standard deviation for each of the latent variables. In doing this, we try to approximate the *true posterior* $p(\mathbf{z} \mid \mathbf{x})$ with an *approximate posterior* $q_\phi(\mathbf{z} \mid \mathbf{x})$. The approximate posterior is implemented as a neural network with trainable parameters ϕ. The decoder then takes a sample from this distribution as an input, and again gives a reconstruction of the original input.

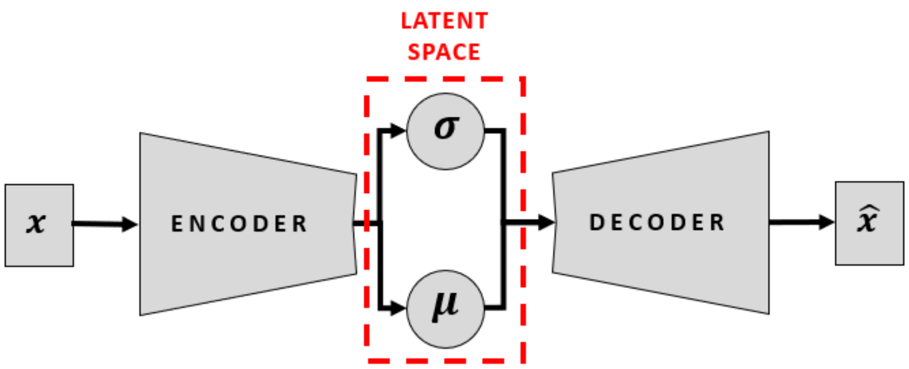


Figure 7 – Schematic overview of a *variational* auto-encoder. The encoder and decoder are implemented using neural networks.

The latent space is designed to follow a fixed *prior* distribution $p(\mathbf{z})$. Generally, this is chosen to be a multinomial standard Gaussian. The loss function restricts the posterior to be close to the prior, as discussed below. Because the approximate distribution of the latent space is then known, a sample can be taken from this multinomial Gaussian. If this sample is fed to the decoder, a new, unseen data-point will be created. This is where a VAE can take on a generative role. This is schematically depicted in Figure 8. Note that the encoder does not play a part in this process.

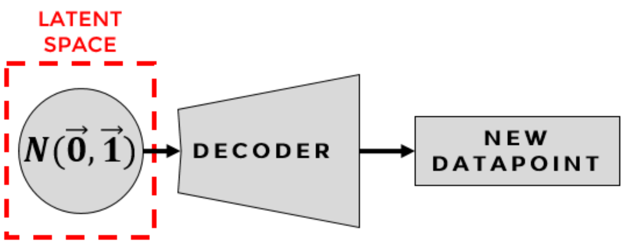


Figure 8 – Schematic overview of the generative process where a VAE is used to generate a new datapoint.

VAE Loss Function. The standard loss function of a VAE consists of two main terms, see equation (3.2). The first term is some distance measure d between the reconstruction and the original input. This should be minimized, in order to produce good reconstructions. The second term is used to restrict the latent space to the chosen prior distribution. Usually, this is the KL-divergence [26] between the fixed prior and the approximate posterior that is found by the model.

$$\mathcal{L}_{\mathrm{VAE}}(\mathbf{x}) = d(\mathbf{x}, \hat{\mathbf{x}}) + KL(q_\phi(\mathbf{z} \mid \mathbf{x}) \mid\mid p(\mathbf{z})) \tag{3.2}$$

These two terms counteract each other. To make optimal reconstructions, the highest amount of information in the latent space is of course desirable. However, inserting information into the latent space requires the approximate posterior to further deviate from the prior, resulting a larger KL-divergence. Careful balancing of these two losses is still a relevant research subject, as this has a large influence on model performance [27, 28].

When a standard Gaussian prior is used, as is often the case in VAE design, the KL-divergence in equation (3.2) can be written in a closed form. This closed form is given in equation (3.3), and the derivation can be found in Appendix C.

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

Here μ_q and σ_q are the mean and standard deviation of the approximate posterior in one dimension. For a multi-dimensional latent space, a summation is required over all dimensions.

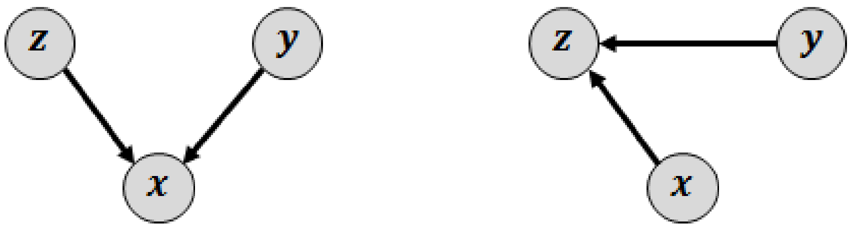
Reparameterization Trick A neural network cannot be trained directly through a stochastic node, as no gradient can be taken for a stochastic node. Usually, the latent distribution that the encoder outputs is a normal distribution with mean μ and standard deviation σ, i.e. $z \sim \mathcal{N}(\mu, \sigma^2)$. Directly taking a sample here would not allow taking the gradients that are needed for neural network training.

To solve this problem in VAE training, the reparameterization trick is implemented [25]. By reparametrization of z to $z = \mu + \sigma\epsilon$, where $\epsilon \sim \mathcal{N}(0, 1)$ is a Gaussian noise factor, deterministic nodes for μ and σ can be used, which can be optimized by using backpropagation.

3.4.3 Conditional VAE

A 'vanilla' VAE can generate new samples by sampling from the latent space, but if no restrictions are enforced on this data-point, it will be nothing more than a random sample. Although this could provide information about what might be high-probability behavior of the machine, little information is obtained about what specific settings for the tokamak reactor have as an effect on the data, and this limits the potential of the model.

To gain some more control over what the properties of the newly generated sample are, a *conditional* VAE (CVAE) can be implemented. In this implementation, the decoder is extended to allow for the input of conditions, aside from the input from the latent space. The latent space will then still contain relevant information regarding other sources of variation, that are not known as conditions, but the sample that is generated should be forced to have properties related to the given conditions. Graphically, this is given in Figure 9.



(a) Generative **(b)** Inference

Figure 9 – Graphical model for conditional VAE.

In such a model, new samples can be generated for some given condition $\mathbf{y}$ by supplying the decoder with both $\mathbf{y}$ and a sample from the prior of the latent space. The models used in this project are based on the concept of a CVAE. However, for the applications to experimental data in this work, a more powerful model is required, which will be described extensively in section 5.

4 Related Work

In literature, several publications can be found that bear some resemblance to the current project. Studies directly related to the current work are described in [29] and [30]. Here, two deep learning applications to JET tokamak data are presented. The first is the application of neural networks to form a surrogate model for plasma tomography, to speed up the processing of sensor data. For the second application, variational auto-encoders are used, but with a very different purpose than the applications in this work. The writers use a VAE for anomaly detection, to find patterns in the data leading to plasma disruptions. Because the VAE has trouble reconstructing these unusual patterns of the anomalies, a high reconstruction error is found between the reconstruction and the original data. This can be coupled to an anomaly score, that can indicate the disruptive patterns. These anomalous samples can then be further investigated.

Because VAEs have been shown to be effective for anomaly detection [31], this type of application is not uncommon in the physical sciences. VAEs were, for example, applied to find anomalous samples of particle collisions in CERN [32]. Such an anomaly would indicate a collision that shows behavior not modeled in the current standard model for particle physics, therefore indicating a potential value of studying this sample. Although the cited work uses similar modeling methods by using a VAE, the goals of the work presented in this thesis are very different from this type of anomaly detection. These articles make very little use of the generative capabilities of the model, which is what a large part of the focus will be on in this project.

For these generative capabilities, generative adversarial networks (GANs) are often used. This is another implementation to model a generative process [33]. In [34], a comparison between using GANs and VAEs for generative applications is presented, where the generative models are used to generate synthetic images of galaxies. The goal of this is to gain insight into dark energy and dark matter, by measuring the gravitational lensing induced by these galaxies. To do this, a conditional variational auto-encoder is used to generate new images, based on certain parameters of interest, such as brightness. This more closely relates to the work presented here, as the generation of synthetic data is also one of the main goals of this project.

There is also a lot of work being done using GANs, rather than VAEs, especially (but not exclusively) in the field of particle physics [35, 36, 37, 38]. However, we opted to not use these methods, for several reasons. Firstly, due to the inherent generator-discriminator adversarial structure of the GAN, they are hard to train [39, 40]. Secondly, the interpretability of the GAN is very low, as well as difficult disentanglement of the latent spaces [27, 41]. Finally, the GAN architecture lacks an explicit inference structure. Since these concepts are fundamentally valuable to this research, the VAE was deemed more promising.

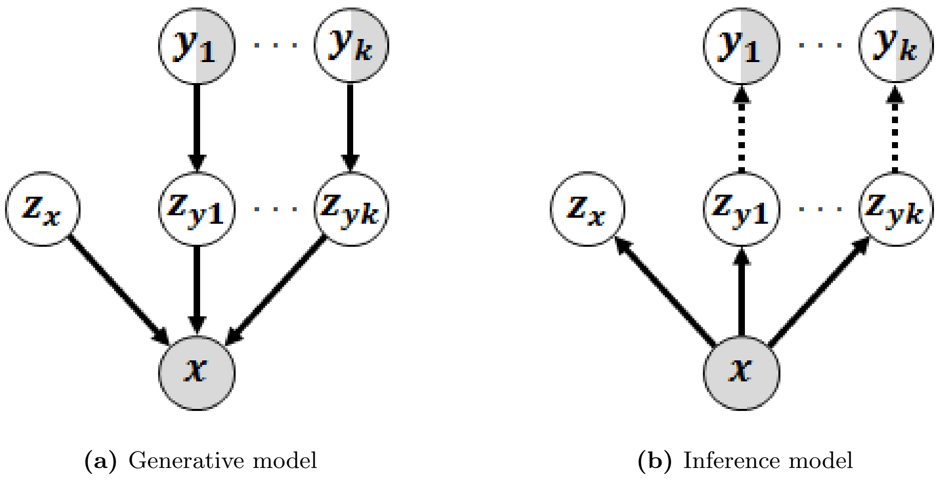
Another application in physical sciences is the usage of machine learning, and specifically neural networks, for statistical inference [42, 43]. Autoregressive models are a viable option here [44], but generative models can also be applied in this setting. VAE structures, or similar, have been adjusted to perform well in these inference tasks [45, 46, 47]. Although not the main focus of this work, we will touch upon statistical inference to determine system parameters, using the same VAE structure that is used for the other applications discussed in this thesis.

To the best of my knowledge, the work reported on here constitutes the first application of generative models aimed at generating tokamak fusion samples. However, there are other applications of neural networks in this domain. This generally comprises methods where neural networks are used to speed up a single part of the simulation process [48, 3]. This is in stark contrast to the methods that are presented in this work, where the generative model is used stand-alone, in a purely data-driven approach, that allows direct simulation of tokamak fusion output profiles.

5 Methodology

5.1 Model Structure

In [49], the Domain Invariant Variational Auto-encoder (DIVA) model is introduced, which uses three separate latent spaces, that all contain different information regarding the sample that is chosen. This model has been adjusted, to obtain a VAE with multiple separate latent spaces. For each of the k conditions y_i in the set of conditions $\mathbf{y}$, a low-dimensional ($d = 3$) latent space $\mathbf{z_{yi}}$ is constructed, while one high-dimensional ($d = 32$) latent space $\mathbf{z_x}$ is used to encode the remaining information. This space should thus not contain information that is linked to the given conditions, but rather to the unknown, hidden variables. The conditional information is disentangled into the $\mathbf{z_{yi}}$ spaces using auxiliary regression networks. Therefore, we call this model the Regressive Disentanglement VAE (ReD-VAE). A graphical abstraction of the model is given in Figure 10.



(a) Generative model **(b)** Inference model

Figure 10 – Graphical model for the ReD-VAE model with multiple separate latent spaces. The half-coloring of the conditions indicates that these are used in semi-supervised fashion. The dashed arrows indicate the auxiliary regression networks.

In the inference process, neural networks $q_{\phi_x}(\mathbf{z_x} \mid \mathbf{x})$ and $q_{\phi_y}(\mathbf{z_y} \mid \mathbf{x})$ approximate the posteriors for $\mathbf{z_x}$ and $\mathbf{z_y}$, with trainable parameters ϕ_x and ϕ_y respectively. The values for $\mathbf{z_x}$ and $\mathbf{z_y}$ are then sampled from these approximate distributions. In the actual implementation, there are separate networks for each condition y_i in $\mathbf{y}$, but for readability and conciseness, they are all summarized in one term here, just like all conditional latent spaces are summarized into one $\mathbf{z_y}$. Following the estimation of $\mathbf{z_y}$, a prediction is made of $\mathbf{y}$, using auxiliary regression neural networks $\hat{\mathbf{y}}_{\omega_y}(\mathbf{z_y})$, with trainable parameters ω_y. These regression neural networks force information regarding a condition y_i to be contained in the latent space for that condition.

During the generative process, the standard Gaussian priors $p(\mathbf{z_x})$ and $p(\mathbf{z_y})$ for each of the latent spaces can be sampled, or if a condition y_i is known, the conditional prior $p_{\theta_y}(\mathbf{z_y} \mid \mathbf{y})$ can be employed to obtain an estimate of $\mathbf{z_y}$. These conditional priors are implemented as neural networks with trainable parameters θ_y. Then, a prediction $\hat{\mathbf{x}}$ can be generated, using a decoder neural network $\hat{\mathbf{x}}_{\theta_x}(\mathbf{z_x}, \mathbf{z_y})$, with trainable parameters θ_x. As a final step, a Gaussian moving filter is applied to smooth out the decoder output. This Gaussian filter has a kernel with standard deviation $\sigma = 3$.

5.2 Loss and Training

The model is trained in a semi-supervised fashion, where both labeled (supervised) and unlabeled (unsupervised) data is used during training [50]. This has two main advantages:

- The size of the used datasets can be increased. As there is no restriction to using data for which all conditions are known, other samples where this information is missing can also be included in training data.
- Using unsupervised training provides more regularization on the z_y latent spaces. This regularization will force the z_y latent spaces to more closely follow a standard Gaussian prior, enabling us to sample from these spaces. This will be further explained later in this section.

The semi-supervised learning is implemented on a per-epoch basis. All samples where some condition y_i is missing are always used as unsupervised training data. Furthermore, in each epoch, a random selection of the supervised training data is also used in unsupervised fashion during that epoch. The size of this selection can be adjusted by a parameter r_{ss}, where r_{ss} is the ratio of the size unsupervised selection over the size of all supervised training data. A supervised ratio of $r_{ss} = 0.5$ was found to perform best, as further demonstrated in Section 8.2. Each epoch is then split into two phases, where first all supervised samples are used for training, followed by all unsupervised samples. For the next epoch, a new selection of unsupervised samples is made, and the cycle repeats.

5.2.1 Supervised Training

The loss function for supervised training is composed of five components and is given in Equation (5.1).

$$\begin{aligned}
\mathcal{L}_{sup}(\mathbf{x},\ \mathbf{y}) = {} & \alpha_1\ \mathbb{E}_{q_{\phi_x}(\mathbf{z_x}|\mathbf{x}), q_{\phi_y}(\mathbf{z_y}|\mathbf{x})} \left[\mathrm{mse}(\mathbf{x},\ \hat{\mathbf{x}}_{\theta_x}(\mathbf{z_x}, \mathbf{z_y})\right] && \text{(5.1a)}\\
& + \alpha_2\ \mathbb{E}_{q_{\phi_y}(\mathbf{z_y}|\mathbf{x})} \left[\mathrm{mse}(\mathbf{y},\ \hat{\mathbf{y}}_{\omega_y}(\mathbf{z_y})\right] && \text{(5.1b)}\\
& + \beta_1\ KL\left(q_{\phi_x}(\mathbf{z_x} \mid \mathbf{x}) \mid\mid p(\mathbf{z_x})\right) && \text{(5.1c)}\\
& + \beta_2\ KL\left(q_{\phi_y}(\mathbf{z_y} \mid \mathbf{x}) \mid\mid p_{\theta_y}(\mathbf{z_y} \mid \mathbf{y})\right) && \text{(5.1d)}\\
& + \gamma\ \mathbb{E}_{p(\mathbf{z_x}), p_{\theta_y}(\mathbf{z_y}|\mathbf{y})} \left[\mathrm{mse}(\mathbf{x},\ \hat{\mathbf{x}}_{\theta_x}(\mathbf{z_x}, \mathbf{z_y}))\right] && \text{(5.1e)}
\end{aligned}$$

All expectations in this loss function are approximated by Monte Carlo sampling. The term given by (5.1a) is the reconstruction loss, where the mean squared error is taken of the reconstruction that is output by the decoder with the original profiles $\mathbf{x}$. (5.1b) gives the regression loss for the auxiliary networks. Increasing the weight α_2 ensures that information in $\mathbf{y}$ is captured in the conditional $\mathbf{z_y}$ spaces. (5.1c) and (5.1d) give the regularizing KL-divergences. The first is the KL-divergence with the standard Gaussian prior for the non-conditional latent space $\mathbf{z_x}$, the second is the KL-divergence with a conditional prior for the conditional latent space $\mathbf{z_y}$. This second term is important in training for both the encoder for $\mathbf{z_y}$ and the conditional prior $p_{\theta_y}(\mathbf{z_y} \mid \mathbf{y})$.

The final term in the loss function (5.1e) is not usually present in VAE models, but was added in the ReD-VAE specifically for the generative goals that this model has. Since a lot of interest goes towards using the model to generate new profiles based on given conditions, this functionality is explicitly added to the loss function. Here, the expectation is taken over the standard Gaussian prior $p(\mathbf{z_x})$ for the non-conditional latent space $\mathbf{z_x}$, and the conditional prior $p_{\theta_y}(\mathbf{z_y} \mid \mathbf{y})$. This is in contrast with the reconstruction term in (5.1a), where the expectation is taken over the approximate posteriors that are output by the encoder networks. This new way of taking the expectation, only allows the model to use the information from $\mathbf{y}$ to generate a new profile, while $\mathbf{z_x}$ can only give the variation that is not due to the conditions $\mathbf{y}$, which is exactly the type of behavior that is desired from such a model. The effectiveness of adding this term will be further demonstrated in Section 8.1.

To make sure that the model sufficiently uses both latent spaces, while still providing good predictions and reconstructions, the five loss weight parameters α_1, α_2, β_1, β_2 and γ need to be carefully balanced. For example, if β_1 is too large, the model can opt to use only $\mathbf{z_y}$, or this can be detrimental to the reconstruction quality. The balancing of the loss parameters should be done very carefully and is critical in the performance of the model. For this project, hyperparameter searches were performed to try and approximate an optimal balance, for which details are given in Appendix A.2.5. In recent work, there have been suggestions to automate this process of parameter optimization, either via heuristics or through machine learning of the parameters [51, 52, 53]. Similar developments for this model could lead to potential improvements to ReD-VAE in the future.

5.2.2 Unsupervised Training

The loss function for unsupervised training is given in Equation (5.2).

$$\begin{aligned}
\mathcal{L}_{unsup}(\mathbf{x}) = \alpha_1 \, & \mathbb{E}_{q_{\phi_x}(\mathbf{z_x}|\mathbf{x}),q_{\phi_y}(\mathbf{z_y}|\mathbf{x})} \left[\text{mse}(\mathbf{x},\ \hat{\mathbf{x}}_{\theta_x}(\mathbf{z_x},\mathbf{z_y})\right] && (5.2\text{a}) \\
& + \beta_1 \, KL\left(q_{\phi_x}(\mathbf{z_x} \mid \mathbf{x}) \mid\mid p(\mathbf{z_x})\right) && (5.2\text{b}) \\
& + \beta_2 \, KL\left(q_{\phi_y}(\mathbf{z_y} \mid \mathbf{x}) \mid\mid p(\mathbf{z_y})\right) && (5.2\text{c})
\end{aligned}$$

Because the training labels are not available during unsupervised training, the auxiliary classifier networks can not be trained during this phase. Furthermore, the conditional prior distributions cannot be used. Instead of these conditional priors, the standard Gaussian prior is used during unsupervised training, specifically for the KL-divergence in (5.2c). This invokes a desired regularization on these latent spaces, as this indirectly also affects the conditional prior, in the subsequent supervised training phase. This regularization is needed to allow for sampling of the latent space, when the goal is to generate data where one or more conditions are unknown. The reason for this is that the conditional latent spaces are only forced to adhere to the standard Gaussian prior in this unsupervised step. During the supervised step, the KL-divergence is taken with the conditional prior, but since this conditional prior is implemented as a neural network without any further restrictions, this leaves the latent space with too much freedom. This will be further demonstrated in Section 8.2.

All neural networks are implemented in Tensorflow/Keras [54]. Further details on implementation and architectures of all models are given in Appendix A.

5.3 Model Evaluation

In order to evaluate the performance of the model, the quality of conditional generation is considered. To do this, the KL-divergence is estimated between the predicted distribution $p_{model}(\mathbf{x} \mid \mathbf{y})$ with the distribution that we expect based on the dataset, $p_{data}(\mathbf{x} \mid \mathbf{y})$.

To get an estimate of the model predicted distribution $p_{model}(\mathbf{x} \mid \mathbf{y_m})$ for a sample m with conditions $\mathbf{y_m}$, we first feed the conditions to the conditional encoders to obtain the distribution $p_{\theta_y}(\mathbf{z_y} \mid \mathbf{y_m})$. Then, a number of samples n_s is taken from this distribution. This is usually set at $n_s = 100$, as this amount leads to consistent results that can still be generated quickly. Now, since this is conditional generation rather than reconstruction, there is no estimate of what $\mathbf{z_x}$ is for these samples, but this variation in $\mathbf{z_x}$ will rather give a variation in the decoded samples that can be attributed to unknown variables. To obtain this variation, n_s samples are taken from the prior distribution $p(\mathbf{z_x}) = \mathcal{N}(\mathbf{0}, \mathbf{1})$. Together with the $\mathbf{z_y}$ samples these are fed into the decoder to get n_s predictions for $\mathbf{x}$. For these predictions, a normal distribution is assumed for each radial point, and then the mean μ_{model} and standard deviation σ_{model} for each radial point are used as parameters for this normal distribution. This gives the resulting $p_{model}(\mathbf{x} \mid \mathbf{y_m})$, for each of the radial points.

As an estimate of the distribution $p_{data}(\mathbf{x} \mid \mathbf{y_m})$ belonging to the conditions $\mathbf{y_m}$ is desired, using only the sample m is not sufficient, as more data is needed for these conditions to get an estimate of the mean and variance. As there generally is only one sample with a specific set of conditions in the dataset, we need to use samples that have conditions that are similar to those of the sample m. Another sample n is considered similar if for all conditions $y_{m,i}$ in $\mathbf{y_m}$, the value of $y_{n,i}$ is within a certain threshold t, multiplied by the range in training data for this condition, according to condition (5.3).

$$|y_{m,i} - y_{n,i}| < t \cdot (\max_j y_{j,i} - \min_j y_{j,i}) \qquad \forall\ y_{m,i} \in \mathbf{y_m} \tag{5.3}$$

Then, only when the amount of samples that are found to be similar exceeds a minimum value n_{min}, this sample m is considered in the model evaluation. This requirement is used to further guarantee the quality of the estimated distribution, as no good estimate can be made when only presented with very few samples. The data distribution $p_{data}(\mathbf{x} \mid \mathbf{y_m})$ is found by taking the mean μ_{data} and standard deviation σ_{data} of all similar samples, again assuming normality.

If the requirement for a minimal number of similar samples is met, the KL-divergence is calculated between the two distributions, according to equation (5.4).

$$KL(p_{model}(\mathbf{x} \mid \mathbf{y_m}) \mid\mid p_{data}(\mathbf{x} \mid \mathbf{y_m})) = \int p_{model}(\mathbf{x} \mid \mathbf{y_m}) \log\left(\frac{p_{model}(\mathbf{x} \mid \mathbf{y_m})}{p_{data}(\mathbf{x} \mid \mathbf{y_m})}\right) \mathrm{d}\mathbf{x} \tag{5.4}$$

Under the assumption of normality for the distributions, this reduces to equation (5.5). A full derivation of this result is given in Appendix C.

$$KL(p_{model}(\mathbf{x} \mid \mathbf{y_m}) \mid\mid p_{data}(\mathbf{x} \mid \mathbf{y_m})) = \frac{(\mu_{model} - \mu_{data})^2 + \sigma_{model}^2}{2\sigma_{data}^2} - \frac{1}{2} + \log\left(\frac{\sigma_{data}}{\sigma_{model}}\right) \tag{5.5}$$

The value for the KL-divergence, averaged over the radius and all samples m is then used, where we aim to minimize this value. During training, it was found that only using the KL-divergence favors models that provide standard deviations that are too small. To remedy this, the inverse KL-divergence, where the roles of p_{model} and p_{data} are interchanged, is also used. Minimizing the sum of the KL-divergence and the inverse KL-divergence, also known as the symmetrised KL-divergence, results in the desired optimization behaviour. The symmetrised KL-divergence is denoted as KL_{sum} in the rest of this work.

For the optimization procedures in this thesis, a value of $t = 0.03$ was used for the threshold, along with $n_{min} = 5$ as a minimum requirement. This value for n_{min} is too low to robustly determine whether the assumption of normality is valid. However, increasing n_{min} to values where such an analysis is possible, would lead to very few samples qualifying for the evaluation, without raising t to a much higher value. This is undesired, as this value for the threshold should be as low as possible to consider two samples similar. Even though it's not possible to confirm the assumption of normality here, the model evaluation metric should still give valuable information here, as it does give a quantifiable metric for how well the mean and standard deviations of the two distributions match.

6 Data

6.1 Simulation

The first dataset that is used is part of the dataset that is described extensively in [55]. This is a small dataset, consisting of 43 samples. A specific JET discharge (#92436) is simulated using a JETTO + QuaLiKiz [19] integrated model from time $t = t_0$ to $t = t_f$. Each simulation contains results for a large number of tokamak quantities, giving the radial profile from $\rho = 0$ to $\rho = 1$ at each timepoint. Both time and radius are discretized into 100 points. The samples are generated by Monte Carlo sampling of the boundary conditions. The boundary condition is a fixed value at $\rho = 0.85$, for each of the quantities of interest.

In this project, we only consider the ion temperature T_i, the electron temperature T_e and the electron density n_e, as these are the relevant quantities to compute the stored energy according to equation (2.15). Because of this, these are also the only boundary conditions that are used. For the first part, the interest lies in (semi-)steady-state solutions, so only the radial profiles for $t = t_f$ are extracted from the original datasets. This results in a set of 43 samples, where each sample contains a radial profile of 100 data points for T_i, T_e, and n_e, as well as three corresponding values for the boundary conditions. In preprocessing, the data is scaled using a min-max scaler, that maps the values linearly to the range [0, 1].

6.2 Experimental

The experimental dataset is extracted from a database of JET reactor data, on which Gaussian process regression was applied, as described in [55]. As experimental profiles, the electron temperature T_e, the electron density n_e, the ion temperature T_i, the ohmic current density j_{ohm} and the effective charge Z_{eff} are extracted, all having dimensionality 101.

The database contains 13109 samples. This number is filtered down to 11612 samples, by only choosing samples that adhere to all of the following conditions:

- Experimental profiles T_e, n_e and T_i are all available.
- Peak T_e is smaller than 20 keV.
- T_e and T_i profiles don't have more than 2 maxima or minima, not including endpoints.
- n_e profile doesn't have more than 4 maxima or minima, not including endpoints.
- Core T_e is greater than 500 eV.
- n_e profiles don't have a constant value for more than 4 consecutive radial points.
- Ohmic heating power is positive but smaller than 50 MW.

These requirements filter out all data that looks to be coming from broken sensors.

In Table 1, all experimental parameters that are extracted are given. These take the role of the conditions $\mathbf{y}$.

Name	Symbol	Amount Missing
Toroidal magnetic field	B_t	0
Plasma current	I_p	0
NBI input power	P_{NBI}	0
ICRH input power	P_{ICRH}	0
Main isotope mass	M	0
Upper triangularity	δ_u	0
Lower triangularity	δ_l	0
Total gas input rate	g_r	5312
Total gas dosage	g_d	5312
Inner strike point radial coordinate	R_i	2200
Inner strike point height coordinate	Z_i	2200
Outer strike point radial coordinate	R_o	2197
Outer strike point height coordinate	Z_o	2197

Table 1 – Control room parameters that have been extracted, along with their symbols used and the number of times this data is missing for a sample.

It is assumed that if either input power parameter is missing, this means that this form of input power was not applied during this shot, and the power is thus simply 0. This leads to the 0 for amount missing for both these control room parameters in Table 1.

For all samples, some more information was also extracted from the database, which is not used as conditioning in the model, but is needed for some of the results presented in this work. This includes the following information:

- Whether a sample is predicted to be in H-mode or L-mode, based on some heuristics.
- The phase of a sample (ramp-up, ramp-down or flat-top current).
- The Ohmic heating power, calculated from the applied plasma current.

In preprocessing, the conditions are scaled using a min-max scaler. For the experimental profiles, a robust scaler is used. This removes the median from the data and scales by the interquartile range, and is more robust to potential outliers that remain in the dataset after filtering.

The dataset was split into a training and testing set, where the testing set comprises 20% of the samples for which all conditions are available. This choice was made as full flexibility regarding the conditions is desired when testing, so it's important to have all data available. From the testing set, 100 samples are used as a validation dataset, for which the model hyperparameters have been optimized.

Most of the experiments are performed with a model that is trained on experimental profiles for T_e, n_e, and T_i. However, for some of the experiments, a model is used that also outputs predictions for j_{ohm} and Z_{eff}. This is trained on a subset of the data, with the following additional constraints:

- j_{ohm} and Z_{eff} profiles are both available.
- Peak j_{ohm} is smaller than $3 \cdot 10^5$ Am^{-2}.

This further reduces the size of the dataset to 10729 samples, where 5042 are missing gas input data, 1862 are missing inner strike point coordinates and 1859 are missing outer strike point coordinates.

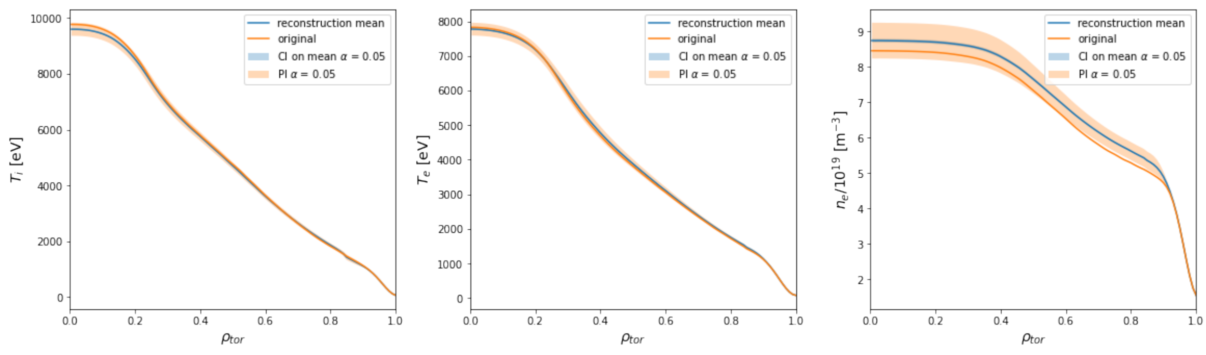
7 Experiments on Simulated Data

7.1 Model Without Conditions

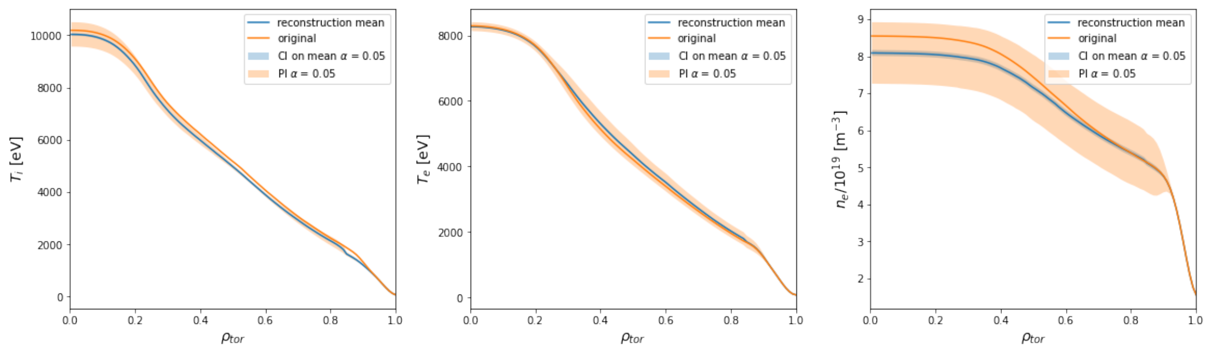
7.1.1 Reconstructions

First, a regular VAE is trained on the data, without any conditioning. To make sure the model remains interpretative, a low-dimensional latent space is used with dimensionality $d_l = 5$.

Reconstructions of unseen testing data are given in Figure 11. To estimate the variance induced by the latent space estimations not being fixed points, 100 reconstructions were made. The mean, 95%-confidence interval on the mean, and the 95%-prediction interval of the samples are all indicated in the figure.



(a) Closest reconstruction of testing data.



(b) Most separated reconstruction of testing data.

Figure 11 – VAE reconstruction of simulated testing data.

The best reconstruction in the testing data has an average relative error of 0.022. For the worst reconstruction, this value is 0.030, so the reconstruction quality is overall very good. When a reconstruction comes with a larger relative error, this might appear to indicate bad model quality, but this is not necessarily true. Of course, the goal of the VAE (a *generative* model) is not to reconstruct original data, but rather to generate new samples. If a perfect reconstruction is desired, it makes no sense to output a mean and standard deviation for the latent space, rather than a fixed point estimate, as this only introduces more variability. It is the influence of the KL-divergence term in the loss that causes a reconstruction to not always be as accurate as possible.

7.1.2 New Data Generation

By sampling from the latent space, new samples can be generated. Since no conditioning is used here, the variability in these new samples is representative of the variability in the training dataset. Some summary statistics of these samples are presented in Figure 12.

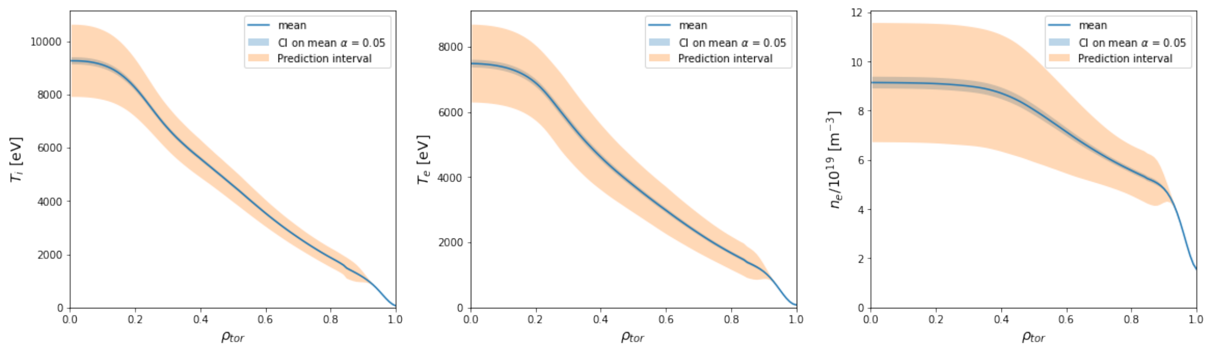


Figure 12 – Mean, confidence interval and prediction interval of newly generated samples using a VAE.

Since these samples are generated without any conditions applied to them, the prediction interval here is mostly just a representation of the variance in the original dataset. This correspondence is almost exact, as can be seen in Figure 13. Although this does not give any new information regarding the data, this does show that the VAE effectively captures the properties of the dataset. In this regard, this is a confirmation that the model is working as intended.

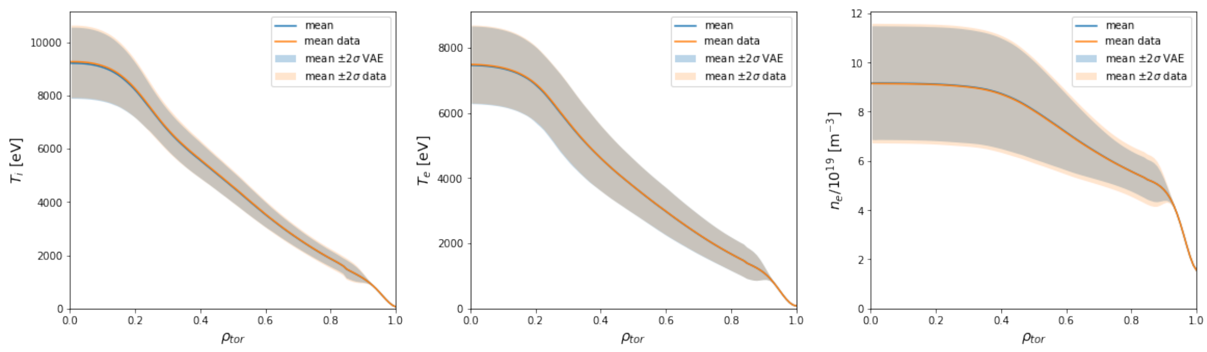


Figure 13 – Comparison of the mean and standard deviation of VAE-generated samples with the same summary statistics in the original dataset. A close to perfect correspondence is observed, indicating that the model correctly captures the properties of the dataset.

For more useful generated samples, some conditioning needs to be applied, which will be discussed in Section 7.2. However, the latent space can already be used for further investigation.

7.1.3 Investigation of Latent Space

To gain insight into the behavior of the latent space, the encoder can be used to place each sample in the latent space. Firstly, it can be confirmed whether or not the weight of the KL-divergence is large enough, by executing a Kolmogorov-Smirnoff test for normality for each of the latent dimensions. The results of these tests are given in Table 2. Based on a significance level $\alpha = 0.05$, there is no reason to assume that normality is violated. Therefore, this confirms that the influence of the KL-divergence is strong enough.

Table 2 – Kolmogorov-Smirnoff test results for normality. Based on these tests, there is no evidence to assume that the latent dimensions are not distributed normally.

Latent dimension	p-value
1	0.30
2	0.08
3	0.74
4	0.18
5	0.16

By taking two of the five latent dimensions, a scatter plot can be made of the latent predictions that the encoder makes for each sample[2]. This is shown in Figure 14, where in each of the subfigures another boundary condition is used for the coloring.

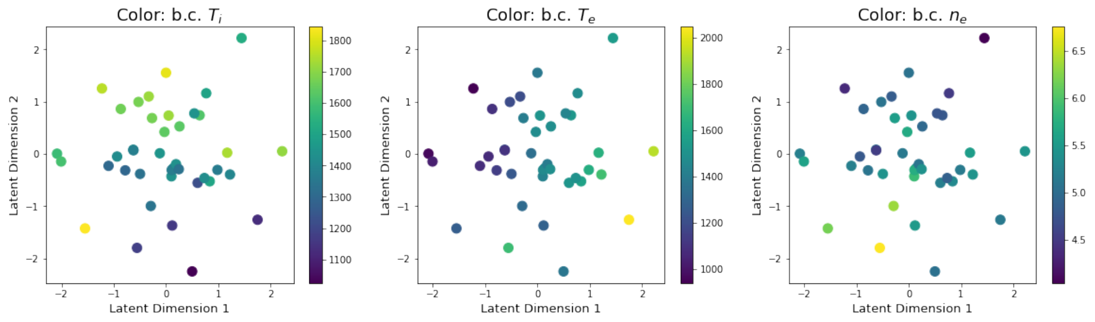


Figure 14 – Latent mean predictions of the encoder for the training data. By coloring the datapoints according to their boundary conditions, correlations between these quantities and the latent variables can be discovered.

This shows that, apart from a few exceptions, samples with similar conditions are mapped close

[2]The encoder actually outputs a distribution, i.e. a mean and standard deviation from which a sample can be taken. To keep results consistent, the mean is plotted here, rather than a sample from this distribution.

to each other in the latent space. Low T_e seems to correspond to a low value of z_1, while z_2 seems to positively correlate with T_i and negatively with n_e. To further quantify these relationships, a correlation matrix can be created. This correlation matrix is given in Figure 15.

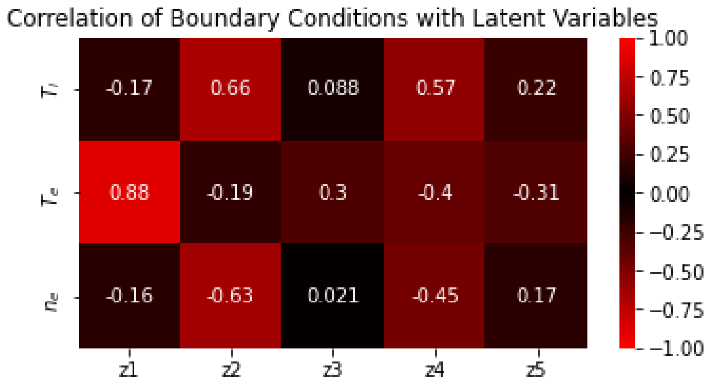


Figure 15 – Correlation matrix of the boundary conditions with the latent variables. The Pearson correlation coefficient is used. Especially the first two latent dimensions correlate strongly with the boundary conditions, indicating that these conditions are important for a description of the data.

This matrix confirms the hypotheses of a strong correlation between z_1 and T_e, and of a positive correlation between z_2 and T_i and a negative correlation between z_2 and n_e. These strong correlations indicate that these boundary conditions contain important information for the reconstruction of the experimental profiles. This was to be expected, as this dataset was created by simulating profiles while varying the boundary conditions. However, the fact that the model recognizes this importance, shows that this can be used as a tool to find out which parameters are potentially relevant or irrelevant, once it is applied to a more extensive dataset.

7.2 Conditional Model

To be able to have more influence on what type of samples is generated by the VAE, the conditional VAE can be used. The boundary conditions can be added to the model as distinct conditions. Of course, other parameters could also be added, but the choice for the boundary conditions was made here because those are the most important difference between different samples.

First, experiments will be presented where only one of the boundary conditions is added to the model (T_e, specifically). Thereafter, boundary conditions for each of the quantities of interest will be added, and a comparison of the results is given.

7.2.1 One Condition

Reconstructions By supplying the decoder with more direct information in the form of boundary conditions, the quality of reconstructions improves. For the vanilla VAE, the average mag-

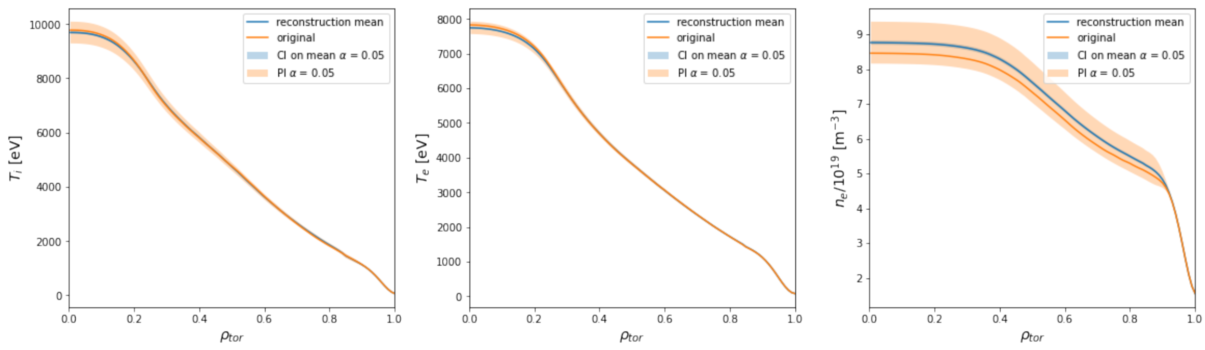


Figure 16 – CVAE reconstruction, using the boundary condition of T_e as a condition for the model. This increases accuracy of the reconstruction.

nitude of the relative error on the testing set was 0.0262. In this case, this value decreases to 0.0192.

A reconstruction using the CVAE is given in Figure 16. We observe that the smallest standard deviation is found in the reconstructions of T_e. This was also the relevant boundary condition that was given as input to the model, so by adding this condition the variability in that profile is decreased most.

Conditional Generation Using the conditioning, predicted output profiles for a single boundary condition can be made. In this experiment, the model is given the T_e boundary condition of a testing sample. Then, a prediction of the full profiles is made using the model. A result of this is given in Figure 17. This is for the same sample as the profiles in 16. Again, one hundred samples were taken from the latent prior to obtain an estimate for the variance.

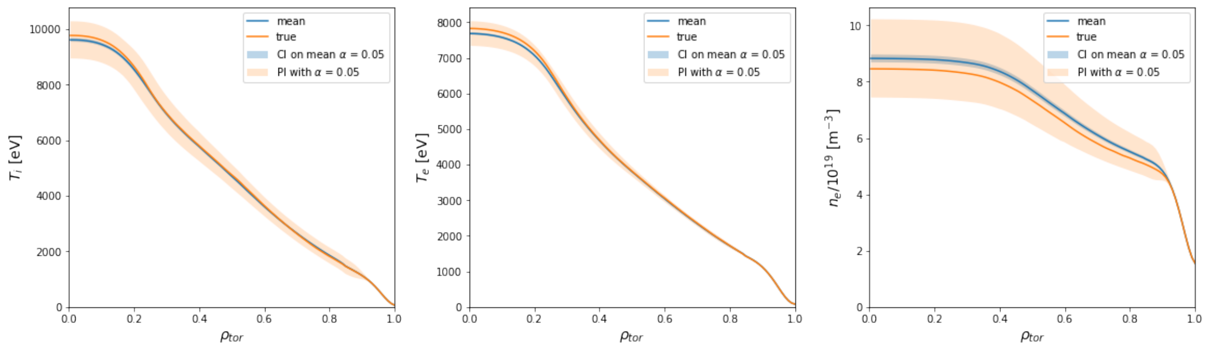


Figure 17 – Prediction of the profiles only giving the boundary condition of T_e to the model.

Because in this sample generation procedure no information is obtained from the encoder, but samples are taken directly from the latent prior, the standard deviation increases compared to the reconstructions given earlier, where latent information was used. Note also, that in this case, the profiles are not necessarily expected to match perfectly. Because the model only has limited information, the true profile is rather expected to be somewhere in the prediction interval with high probability.

Latent Space Adding conditions influences the latent space. Because the conditioning already gives the model information regarding the T_e boundary condition, this information should no longer be present in the latent space, due to the KL-divergence regulating the amount of information here. The correlation matrix of boundary conditions with the latent variables is given in Figure 18. This shows that, indeed, all correlation with T_e disappears from the latent space.

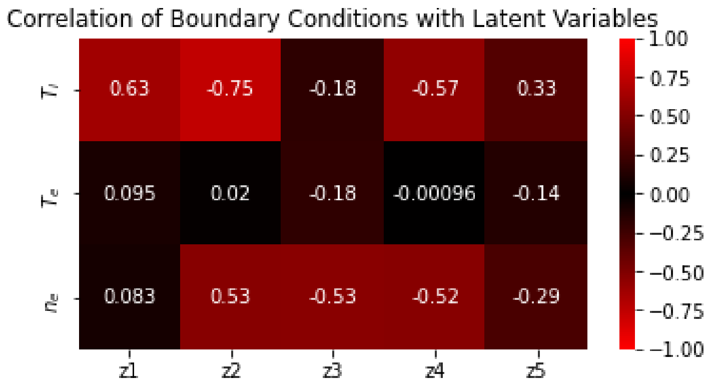


Figure 18 – Correlation matrix of the boundary conditions with the latent variables. The Pearson correlation coefficient is used. Because the T_e boundary condition was given to the model, this is no longer modeled in the latent space, and correlations with this boundary condition are much smaller.

7.2.2 Full Conditions

Now, the model architecture remains generally the same, but in addition to the boundary condition of T_e, the conditions for T_i and n_e are also passed to the CVAE. This has consequences for data generation and the latent space behavior.

Reconstructions Again, by giving more information to the model, reconstructions are improved. The same sample as given in Figure 16 was reconstructed using this model, the results are presented in Figure 19. Using all conditions leads to high-quality reconstructions.

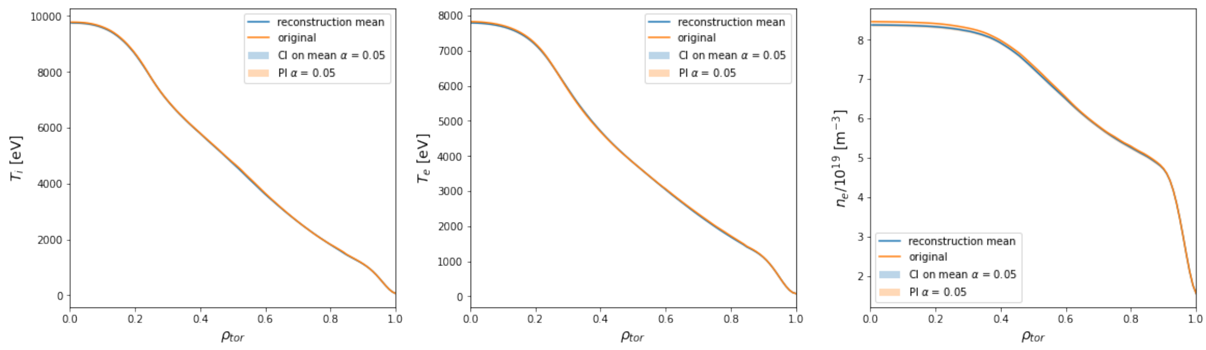


Figure 19 – Reconstruction of a testing sample. Because all boundary conditions were passed as input to the model, the variance of the predictions is very low. The reconstruction is almost exact.

Because all conditions were given, the model has a lot of information and there are few hidden factors. Therefore, the variance in the reconstructions decreases. A comparison of the reconstruction quality for all these models is given in Table 3.

Model	Mean relative error magnitude
Vanilla VAE	0.0262
CVAE on T_e	0.0192
Full CVAE	**0.0121**

Table 3 – Comparison of reconstruction quality for different amounts of conditioning. Increased information for the model in the form of conditions leads to better reconstructions.

Latent Space Collapse The simulation dataset on which these experiments are performed is very small. Furthermore, the different samples are created by sampling different boundary conditions, without changing other parameters. This means that by supplying the full set of boundary conditions, there is no longer any hidden information for the latent space to represent. Because of this and because of the regularization of the KL-divergence, the model opts to not use the latent space, and base predictions purely on the conditioning. This results in the encoder output being the same for all samples and dimensions: a standard Gaussian. By predicting a latent mean of 0 and standard deviation of 1, the model fully adheres to the standard normal prior, thus minimizing the KL-divergence. This phenomenon is known as the collapsing of the latent space.

Because the predicted mean and standard deviation are the same for all samples, the latent space effectively does not contain any information. These values are only barely used for reconstruction by the model: sampling from the latent space while keeping the conditions constant does not result in very different reconstructions, but just results in a small variance estimate. However, the model can still be used to make predictions of the profiles for given conditions.

Conditional Generation By using the full conditional model, data can be generated where all the boundary conditions are known. An example of such a generated data point is given in Figure 20. This was the sample in the testing set for which the correspondence between the data and the generated prediction was the worst.

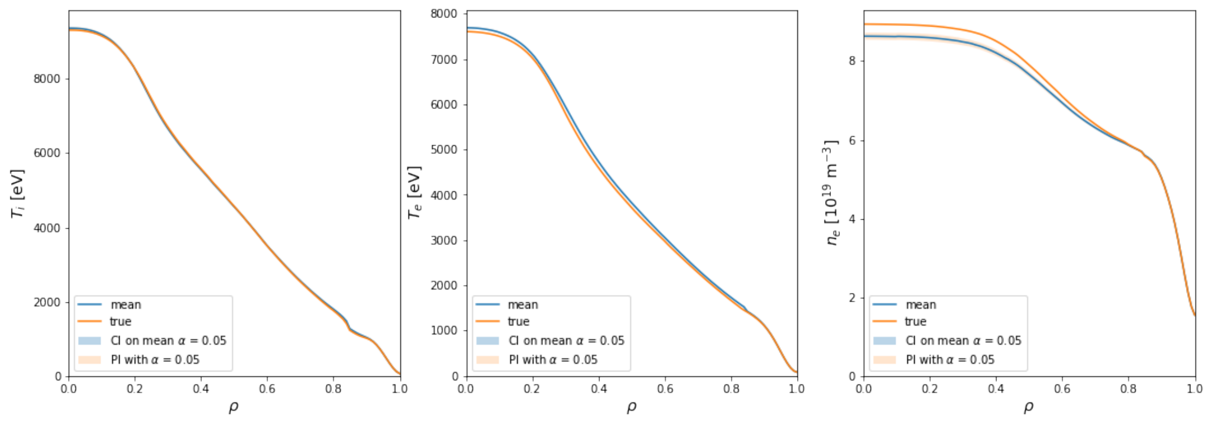


Figure 20 – Generated sample, compared to a sample from the testing set with the same conditions. This generation has the worst correspondence from all testing samples. The collapse of the latent space makes it so that the CI and PI have negligible sizes.

Even though this is the worst sample in the testing set with regards to correspondence, the model predictions are still very good. Over all test samples, the average magnitude of the relative error is only 1.38%, showing very good performance in predicting experimental outputs. It is important to note here that, just like in the generation experiment in section 7.2.1, the latent prior was sampled one hundred times to get an estimate of the confidence and prediction intervals. However, due to the latent space collapse, this sampling procedure has very little effect, leading to negligible intervals, as all generations are almost the same.

Now that it has been confirmed that reliable predictions can be made, the CVAE model can be used to make predictions for dependencies on the boundary conditions. Some of these can be used to compare with literature to confirm correctness. Such an example is given in Figure 21. A similar dependency is given in Figure 9 in [56].

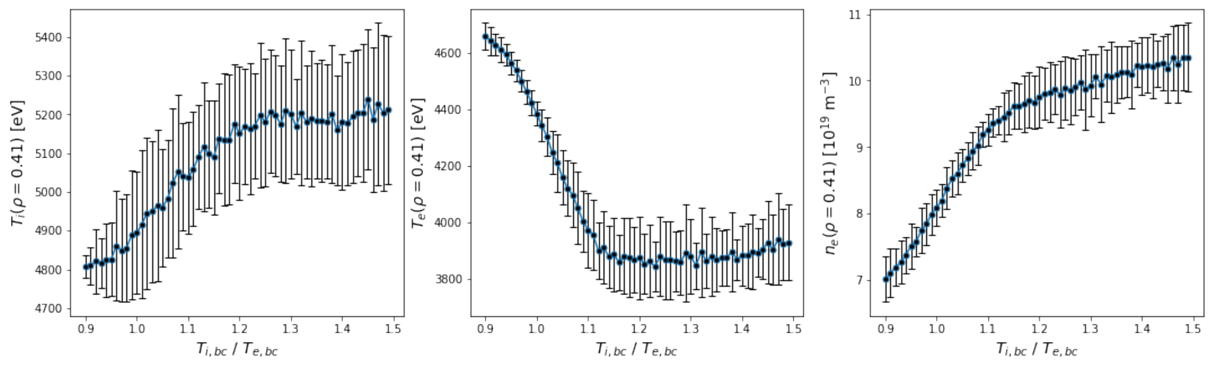


Figure 21 – Behavior of T_i, T_e and n_e at $\rho = 0.41$ for changing ratios of boundary condition temperatures, with corresponding standard deviations according to variation of the n_e boundary condition.

The profiles were generated for a boundary electron temperature $T_{e,bc} = 1100$ eV and varying boundary ion temperature in order to find results for different ratios. Because the electron density boundary condition is unknown, some variance is expected. This is found by taking 100 samples from a fitted normal distribution to the electron density boundary condition distribution in the data. Then for these 100 samples, the profiles are recreated, giving a mean and standard deviation.

Qualitatively, similar behavior is found as in [56], where more density peaking is observed with increasing $T_{i,bc}/T_{e,bc}$. This indicates that the model learns these implicit patterns in the data, even when supplied with only a small training dataset. A quantitative agreement between this result and the literature is not expected, as there are many differences between the applied settings.

Figure 22 shows another trend prediction. Here, the ratios of core temperature to boundary condition temperature are shown, plotted against the different boundary conditions for the temperatures. For three fixed ratios, a model prediction of the behavior is presented. Furthermore, the training data is given, as well as predictions for these specific training data points.

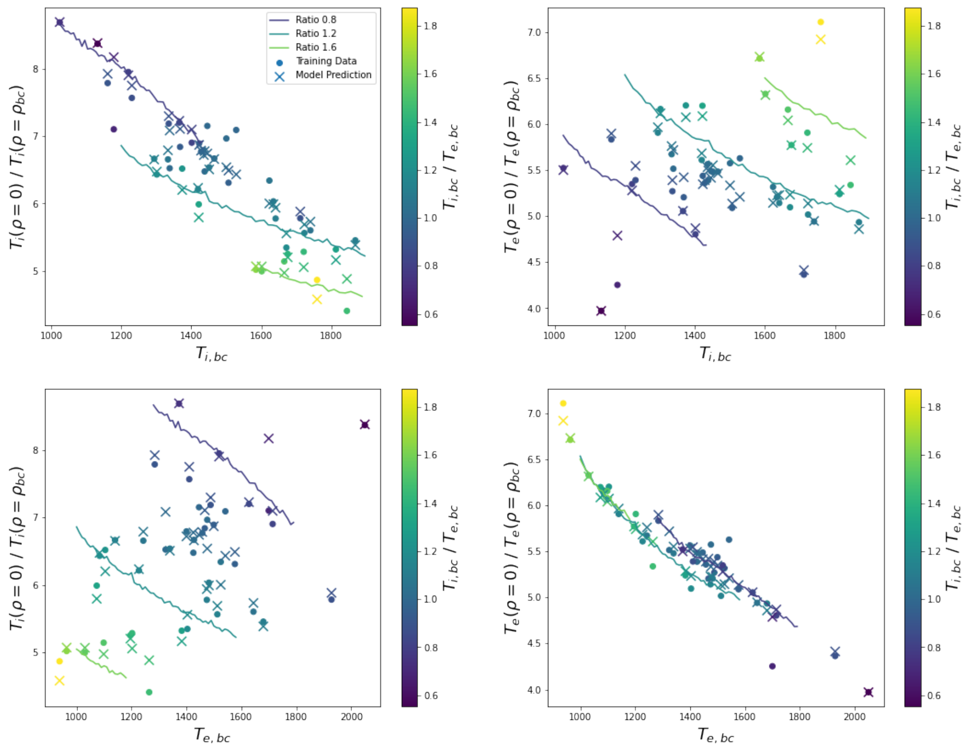


Figure 22 – Dependencies of peak electron and ion temperatures on the boundary conditions. For some fixed ratios of $T_{i,bc}/T_{e,bc}$, predictions are included to show the behavior for specific sets of conditions. Also, the training data, as well as predictions for these specific training data points are plotted. The correspondence between these two is generally good.

The training data and model predictions correspond for almost all samples. It's also seen that the profiles that are made for fixed boundary condition ratios, follow patterns that are visible in the data. For example, the figure shows that peak T_e temperature ratio is mostly dependent on the boundary temperature for T_e, and not so much on T_i, as the fixed-ratio lines all follow roughly the same curve. For T_i peaking, there does appear to be some dependency on the electron temperature.

By using these methods, these dependencies can be examined, perhaps leading to new insights. These insights will mainly be valuable when using experimental data, rather than the synthetic dataset on which this was trained. Therefore, the following section will discuss the experiments on the experimental JET data, using the more powerful ReD-VAE model.

Extrapolation Some extrapolation is possible using the CVAE, but when going far outside of the training data range, the model's predictive capabilities diminish. If conditions are fed that lie far outside of the training range, unrealistic profiles are created, with sharp gradients. Therefore, when making any conclusions based on data generated by these models, one has to take care that the training range allows for these conclusions.

8 Experiments on JET Data

In this section, the experiments will be discussed where the experimental JET tokamak data is used as training and testing data. For this, more modeling power is required than in the previous section, so the extensive ReD-VAE model described in section 5 is used here.

8.1 Conditional Generation Loss

First, the positive effect is demonstrated of the addition of a term for the conditional generation to the loss. As discussed in section 5.2.1, the following loss term is added to the overall loss:

$$\mathbb{E}_{p(\mathbf{z_x}),p_{\theta_y}(\mathbf{z_y}|\mathbf{y})}\left[\mathrm{mse}(\mathbf{x},\ \hat{\mathbf{x}}_{\theta_x}(\mathbf{z_x},\mathbf{z_y}))\right]$$

This term is weighted by a factor γ, and effectively minimizes the mean squared error between a set of profiles $\mathbf{x}$ and the prediction $\hat{\mathbf{x}}$ when the model is only supplied with the corresponding conditions $\mathbf{y}$.

In Figure 23, the value of the model evaluation metric is given for multiple values of the weighting factor γ. The mean and standard deviation in the Figure were determined by training 11 models for each value of γ. Once the value for γ passes a transition region around $\gamma = 50$, the model performs substantially better. In the transition region, a bifurcation is observed where a model either performs well or poorly, with nothing in between, leading to a large standard deviation. For larger values of γ, the gradient diminishes, but the effect in the region under $\gamma = 200$ is substantial. We therefore conclude that the addition of the conditional generation loss term has a positive effect on model performance.

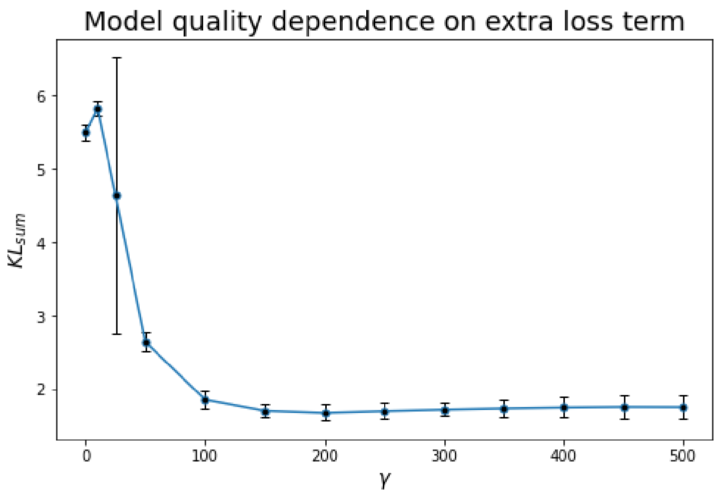
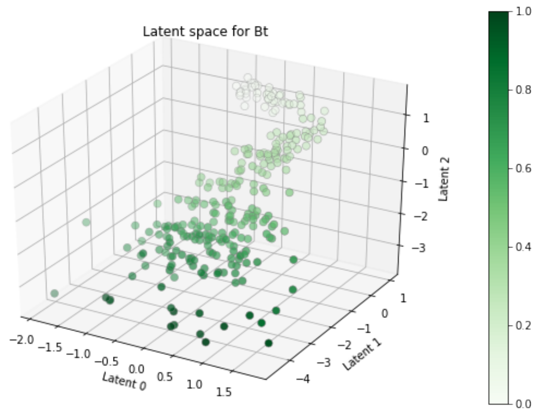


Figure 23 – Progression of the summed KL-divergence that is used as model evaluation metric, for different values of the conditional reconstruction loss weight γ. This shows that increasing γ past some threshold around $\gamma = 50$ drastically increases model performance.

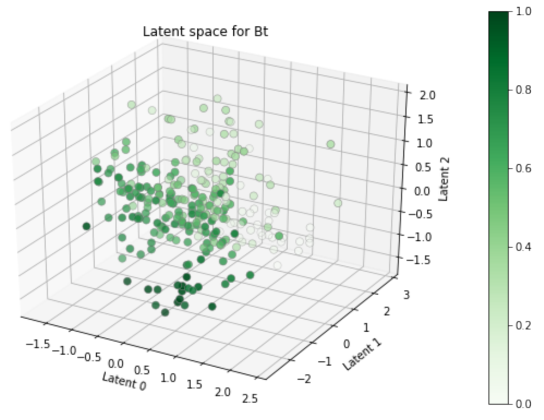
8.2 Semi-supervised Training

As the model is trained in a semi-supervised fashion, both supervised and unsupervised samples contribute to the training. The number of unsupervised samples during an epoch is determined by two factors: Firstly, the unsupervised dataset consists of all samples where data is missing for one or more of the conditions in $\mathbf{y}$. Secondly, from all remaining samples, a fraction r_{ss} is taken, that is also used in an unsupervised fashion.

Training with unsupervised samples has two advantages. The first advantage is that more data can be used during training, as there is no restriction on having all data regarding the conditions for a sample. This increases the number of available samples from 5,184 to 11,612 for our dataset. The second advantage is an increased regularization of the conditional latent spaces. This is demonstrated in Figure 24.



(a) Fully-supervised

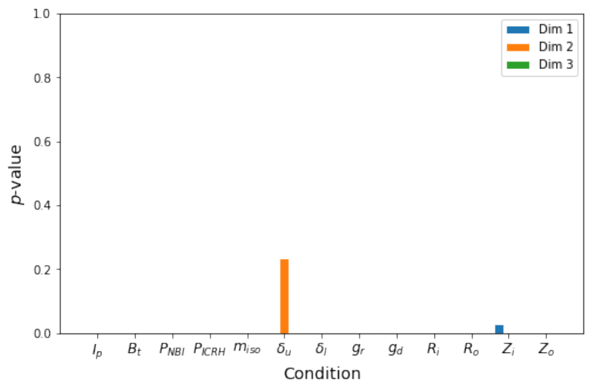


(b) Semi-supervised ($r_{ss} = 0.5$)

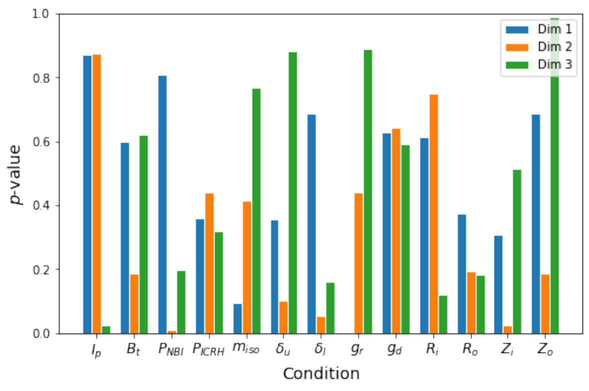
Figure 24 – Side-by-side comparison of latent spaces that were trained in fully-supervised or semi-supervised fashion. Coloring indicates the scaled value of the relevant condition, toroidal magnetic field. In the fully-supervised case, there is too little regularization of this latent space, which makes the encoders diverge from the standard Gaussian prior, which prohibits effective sampling of this latent space if the relevant condition is unknown.

In the fully-supervised case, the regularization of this space is lacking. The model is not constricted to follow the standard Gaussian prior, but only the conditional prior that is implemented using a neural network. As this provides the model with too much freedom, a very non-regular shape is found in the latent space, that diverges a lot from the standard Gaussian prior. This means that a large part of the latent space does not obtain a well-defined meaning, as these parts are not used by the encoder or decoder during training. Therefore, when generating data where this condition is non-fixed, taking a sample from the standard Gaussian prior does not yield valid results, as the posterior distribution does not match this prior. However, in the semi-supervised case, the latent space restriction is working much better. The latent space now follows a standard Gaussian sphere much more closely. Therefore, the prior can be used to sample from this latent space. Even though there is more regularization here, there is still a strong correlation between latent values and condition values, which are indicated by the coloring. This means that the auxiliary regression networks can still work correctly, despite the extra regularization.

To further quantify the differences between fully supervised and semi-supervised training, Kolmogorov-Smirnov tests were performed on the latent space. For this test, the latent space predictions of each dimension were compared with a standard Gaussian. Under the null hypothesis, the distribution of samples in the latent space resembles the standard normal distribution. Figure 25 shows the p-values for these tests.



(a) Fully supervised.



(b) Semi-supervised.

Figure 25 – p-values for Kolmogorov-Smirnov tests to test equality of the sample distribution in latent space and a standard normal distribution. For fully supervised training, p-values are close to zero for almost all dimensions.

For the model that was trained fully supervised, almost all p-values are very close to zero, indicating that the distribution in latent space is very much different from the standard normal distribution. The consequence of this is that samples from the standard Gaussian prior do not accurately represent the distribution in the latent space. For the semi-supervised setting, the p-values are much higher. Only for three of the 39 dimensions, the p-value is smaller than 0.05. With 39 independent tests, the p-value would be expected to be under 0.05 approximately two times, even if the null hypothesis of equality of distributions is true. We therefore conclude that, with semi-supervised training, the latent space distribution does resemble the standard normal prior, while it clearly does not for a model that was trained fully supervised.

The value of the r_{ss} parameter also has a substantial influence on model performance. In Figure 26 the effect of changing this ratio is presented.

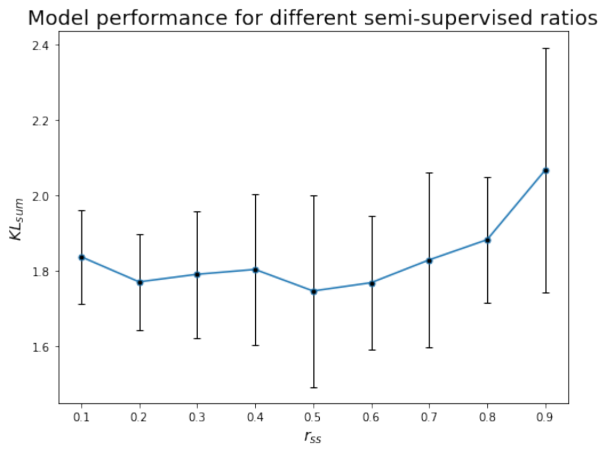


Figure 26 – Model performance for increasing values of r_{ss}. Very low of very high values decrease performance. A minimum is found for $r_{ss} = 0.5$.

The figure shows that neither high values, nor low values of r_{ss} lead to very good model performance. For the low values, this is due to the lack of regularization discussed earlier. For the high values, not enough supervised data is used to train the conditional priors. The model performs best for a ratio $r_{ss} = 0.5$. Therefore, this value was used during training to obtain the results that are presented in the rest of this document. This leads to a model with an average KL_{sum} metric of 1.82 for the entire testing set.

8.3 Conditional Generation

Now that the model hyperparameters have been set, the model can be used for conditional data generation. To do this, a set of conditions $\mathbf{y}$ is taken, which are used in the conditional priors. These provide a mean and standard deviation for each dimension in each conditional latent space, from which we take n_s samples. n_s samples are also taken from the standard Gaussian prior for the $\mathbf{z_x}$ latent space. Both of these are fed into the decoder to create n_s radial profiles for T_e, n_e, and T_i, from which the mean and standard deviation are determined. n_s profiles are taken to get an accurate estimate of the mean and standard deviations. $n_s = 100$ is used, as this gives stable results.

An example of such generation is given in Figure 27, where a sample $(\mathbf{x}, \mathbf{y})$ is used. In this figure, multiple things are presented. Firstly, T_e, n_e and T_i profiles $(\mathbf{x})$ of the original sample are given in blue. Secondly, the mean and standard deviation that are predicted by the VAE are shown in orange. Thirdly, the mean and standard deviation that are expected based on the dataset are given in green. These are constructed by finding samples in the dataset that have conditions similar to $\mathbf{y}$, where 'similar' is defined by condition (5.3). This also means that the evaluation metric discussed in Section 5.3 is effectively the sum of the normal and inverse KL-divergences between the orange and green distributions.

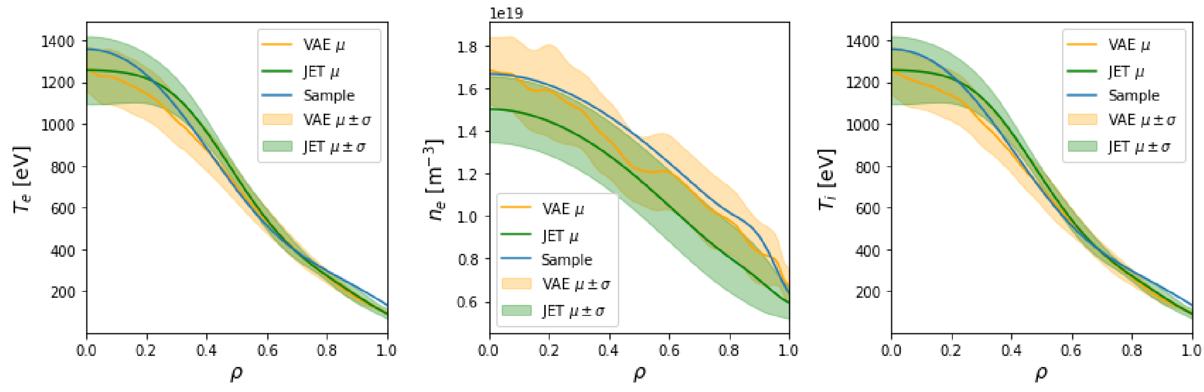


Figure 27 – Example of conditional generation using the generative model. The mean and standard deviation that are predicted by the VAE are shown, as well as those that are expected based on the dataset. The conditions for this sample are: $I_p = 1.05$ MA, $B_t = 1.070$ T, $P_{NBI} = 0$ W, $P_{ICRH} = 0$ W, $M = 1$ u, $\delta_u = 0.228$, $\delta_l = 0.276$, $g_r = 1.71 \cdot 10^{20}$ s^{-1}, $g_d = 1.21 \cdot 10^{23}$, $R_i = 2.39$ m, $R_o = 2.91$ m, $Z_i = -1.73$ m, $Z_o = -1.73$ m.

In the case of this sample, the predictions for T_e and T_i are very close to what's expected for the dataset. The predictions for n_e are, however, slightly off. In an ideal scenario, the original data point can be regarded as a sample from the predicted distribution. This means that the amount of samples within one standard deviation from the mean should be around 68%, and 95% should be within 2 standard deviations. For the testing dataset, a count was made of how many samples are within these boundaries. As small deviations from this are still acceptable, the requirement is set that a sample is counted as correct when more than 60% of the radial points are within the predicted distribution. This 60% might appear as a very loose restriction, but under this restriction, the T_e profile in Figure 27 is already rejected, even though the predictions for this profile match the desired distribution quite closely. Table 4 lists the result for this count, where the 'all' quantity indicates the amount of samples where the requirement is met for all quantities.

Table 4 – Frequencies at which a sample is within one or two predicted standard deviations from the predicted mean.

Quantity	Within $\mu \pm \sigma$	Within $\mu \pm 2\sigma$
T_e	38.0%	79.3%
n_e	40.4%	77.0%
T_i	36.4%	76.9%
all	14.9%	58.9%

The percentages are somewhat lower than in the ideal scenario. This could partly be due to the 60% restriction that was imposed being too strict, as for example the T_e profile in Figure 27 could also be acceptable, even though only around 55% of the sample is actually within one predicted standard deviation from the predicted mean. Therefore, this sample is actually too far off to be counted as a good sample, even though the prediction is very close. However, decreasing the requirement further below 60% would quickly mean that samples can be counted as being within one standard deviation, even though a majority of the sample isn't. Those cases should be avoided, so the threshold was set at 60%.

Still, even though the percentages in Table 4 are not perfect, general performance is very good. The predictions can be used to get an estimate of expected tokamak behavior for a given set of conditions, and the trends in tokamak performance as a function of certain conditions can be investigated, which will be demonstrated in the following section.

8.4 Trend Prediction

8.4.1 Experimental Output Profiles

By using the conditional prediction, a clear understanding can be obtained of how the conditions directly influence the experimental output profiles. To do this, one of the conditions is uniformly varied over a range of values. For each of the values, n_s samples are taken from the conditional prior output. For the other conditions, there are two options: in the first option, the condition retains a fixed value. The conditional prior neural networks are then employed to get a predicted $\mathbf{z_y}$ distribution, from which n_s samples are taken. The second option is to keep the condition non-fixed, or unknown. Instead of sampling the conditional prior, the normal prior for the latent space is sampled. As there is no information regarding the values of $\mathbf{z_x}$, the normal prior for this latent space is also sampled n_s times. The concatenation of all of these latent samples is then fed to the decoder. From this, an estimate for the mean and standard deviation of the experimental output profiles is obtained.

The result for varying I_p is given in Figure 28. Here, I_p is ranged from the minimum to the maximum value in the dataset, and all other conditions have a constant value, given in Table 5. These are conditions under which mostly H-mode samples are expected.

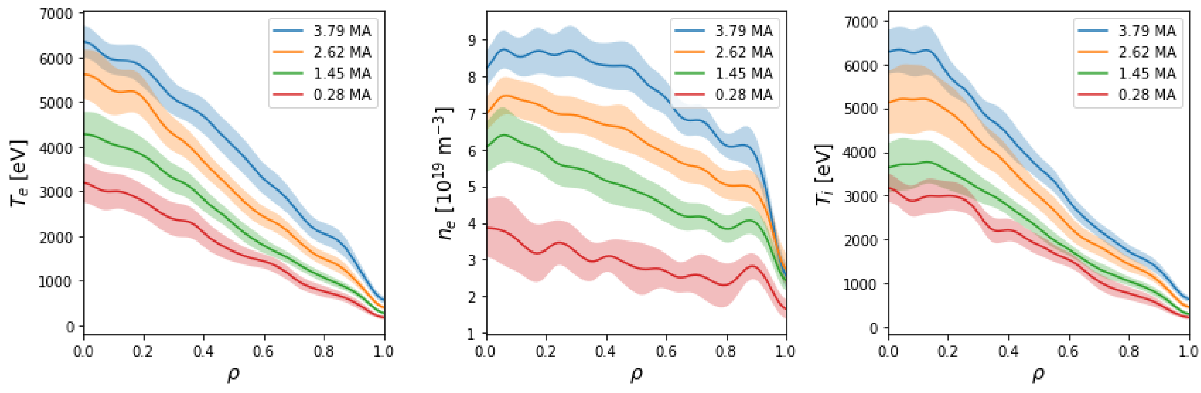


Figure 28 – Changes in output profiles for varying plasma current. All other conditions have fixed values. 100 samples were taken to get an estimate of the mean and standard deviation.

The plot shows that increasing plasma current leads to increased temperatures, as would be expected based on domain knowledge. An increased electron density is also observed for higher plasma currents. Although the density is expected to increase with plasma current, the increase in predicted densities can also be partly due to a correlation between the gas input and the plasma current, which leads to the model overestimating the effect of changing the plasma current. These types of correlations in the training data can reduce the reliability of model predictions.

Figure 28 can be compared to a setting where the other conditions are unknown, and the normal prior is sampled to obtain different values for the $\mathbf{z_y}$ latent variables. This result is displayed in Figure 29.

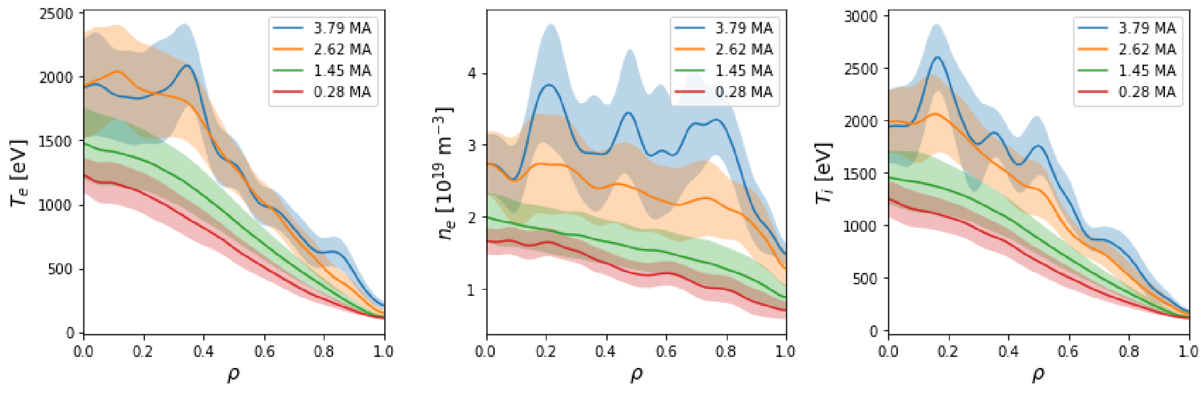


Figure 29 – Changes in output profiles for varying plasma current, when the other conditions are unknown, and the normal prior is used to get latent samples. This can lead to unrealistic combinations of conditions, giving irregularly shaped output profiles.

The thing that stands out most is the irregularly shaped prediction for $I_p = 3.79$ MA. This is due to the very high plasma current involved. When randomly sampling the other conditions, this leads to unrealistic combinations of conditions, as a high plasma current strongly correlates with a high toroidal magnetic field and input power. The model does not handle this unrealistic combination very well, which leads to spiky profiles and big standard deviations. This shows some of the limitations of ReD-VAE: extrapolating into areas far outside the hypercube of training conditions will lead to unrealistic reconstructions.

For the other profiles in Figure 29 some differences are observed, when compared to Figure 28. The profiles generally have lower values, and the pedestal is less clearly visible. The reason for this is that, when sampling randomly from the unknown conditions, L-mode profiles will generally be created. As most of the training data are L-mode samples, this is the 'default' behavior of the output. For the large plasma currents, we do see that a pedestal starts forming, but without the other fixed conditions, the energy in the tokamak will generally be much lower than in Figure 29.

The behavior that is observed corresponds to what is expected based on domain knowledge, as a first sign of correct model performance. However, interest also lies in the quality of the predictions of the performance of the machine, for which the stored energy and confinement time are considered.

8.4.2 Stored Energy

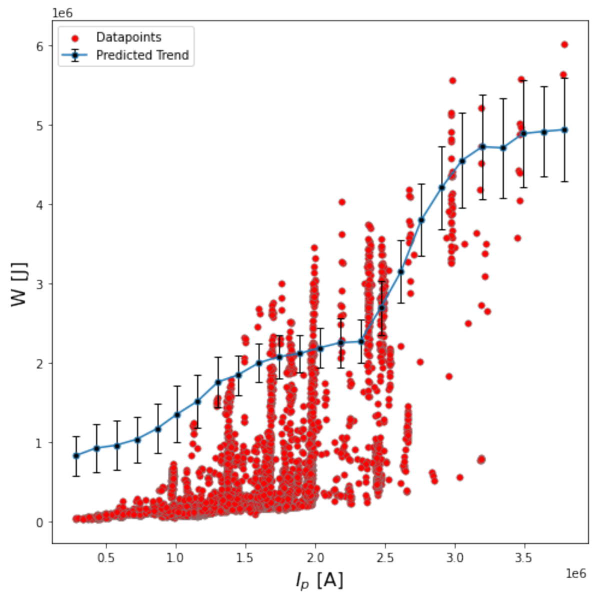
To predict trends for varying conditions, again, a choice can be made to either keep the other conditions fixed or variable. In the figures here, other conditions are fixed at values that stimulate H-mode plasmas. These fixed values are given in Table 5.

Condition	Value
B_t	3.19 T
I_p	3.45 MA
P_{NBI}	22.2 MW
P_{ICRH}	3.12 MW
M	2 u
δ_u	0.172
δ_l	0.334
g_r	$1.51 \cdot 10^{22}$
g_d	$1.17 \cdot 10^{23}$
R_i	2.42 m
Z_i	-1.58 m
R_o	2.77 m
Z_o	-1.67 m

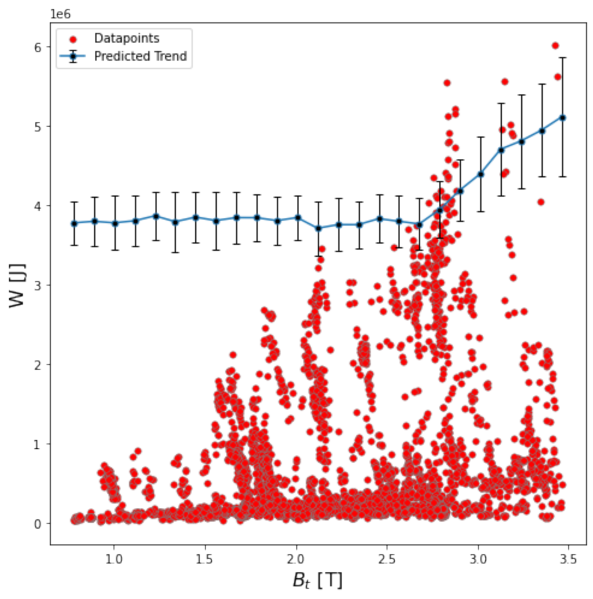
Table 5 – Control room parameters used for trend prediction. These values stimulate H-mode plasmas.

To find the dependence of stored energy on one of the conditions, that condition is uniformly varied from its minimum value in the dataset to the maximum value, while the others remain at the value given in Table 5. The conditional generation procedure is then used to obtain profiles for T_e, n_e and T_i, from which an estimate of the stored energy can be calculated, using equation (2.15). In Figure 30 these trends are given for the plasma current, toroidal magnetic field, NBI power, and gas input rate.

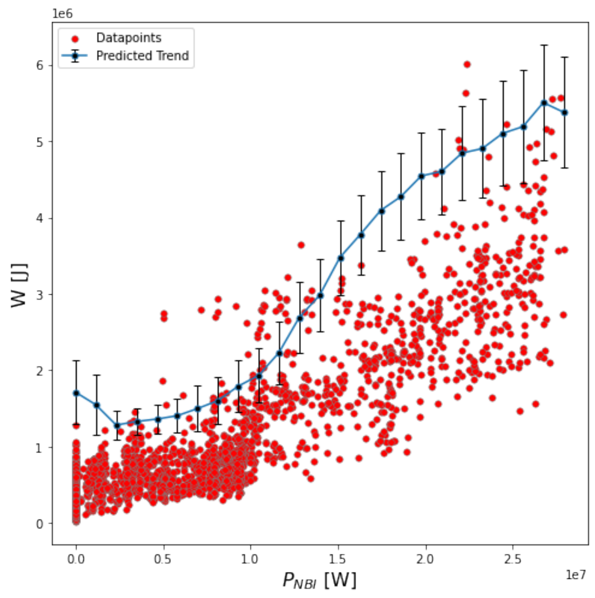
Figure 30a shows that for the most part, the predicted trend matches the data well. Note that it is not expected that the error-bars, indicating the standard deviation of the stored energy, match the distribution in the data, as all but one condition are fixed here. The trend is generally on the high end of the distribution in the data, which again is expected since fixed conditions are used that promote high stored energy. For low plasma current, the trend diverges from the one in the data. This is due to the data-points for these low plasma currents only being L-mode samples. The correlation of plasma current with the other condition also makes it such that the combination of conditions used for these points is not present in the data. In Figure 30c the trend predicted by the generative model again closely follows the trend that is seen in the data,



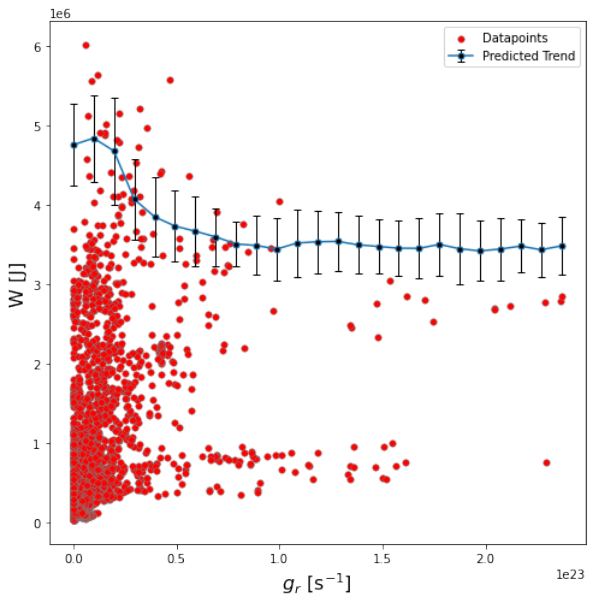
(a) Plasma current.



(b) Toroidal magnetic field.



(c) NBI Power



(d) Gas input rate

Figure 30 – Comparison of stored energy trend predicted by the generative model with the dataset, for four different conditions. Error-bars indicate the standard deviation of the stored energy. Good agreement between data and model predictions is observed.

indicating that the model adequately incorporates the effects of changing conditions.

In Figure 30b the trend diverges much further from the data. At first sight, this might hint towards a bad model, but this behavior is actually what is expected. According to the scaling laws discussed in section 2.2.3, there is very little dependence of stored energy on the magnetic fields expected. The reason for this apparent correlation in the data is due to the strong correlation of magnetic field with plasma current. The fact that the model diverges from the data (for the most part), indicates that it is able to disentangle the effects caused directly by the changing magnetic field, and the correlation with plasma current.

The final condition is shown in Figure 30d. For high gas rates, there is very little data, making it hard to confirm the trend with data. However, it is interesting to see that for this set of conditions, the model indicates that a low gas rate is expected to provide the highest stored energy. Such a result can be relevant when tokamak operators need to decide which gas rate to use for a given set of other conditions. Here lies one of the concrete applications of these trend predictions: for the parameters for which there is no good theoretical model to find the best stored energy, the generative model can be used to get an indication of what could work well, in a much faster and cheaper way than using other simulation software or trying multiple experiments. However, to really predict model performance, the confinement time is important, rather than just the stored energy. Therefore, this will be discussed next.

8.4.3 Confinement Time

Using a ReD-VAE instance that's trained to also output experimental profiles for j_{ohm} and Z_{eff}, predictions for the Ohmic heating can be made using equations (2.9) to (2.14). For all of the data points in the dataset, a prediction for the confinement time is made using this derivation, by sampling from the $\mathbf{z_x}$ space and using the conditional priors for the conditional latent spaces. A comparison between the predicted confinement time and the confinement time that was calculated from the data is given in Figure 31.

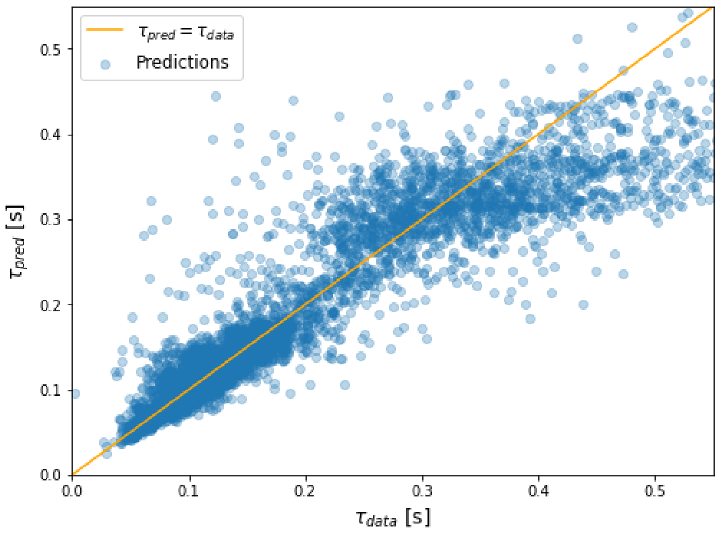


Figure 31 – Confinement time calculated from the predicted profiles against confinement time calculated from the profiles in the data. For low confinement times, the agreement is excellent, while for higher confinement times the predicted confinement time tends to underestimate those obtained from the data.

For low confinement times, excellent agreement is observed. For very high confinement times, the ReD-VAE generally under-estimates the confinement time. The average magnitude of the error is 0.037, the average relative error is 18%. This is good enough to use the predictions sensibly, especially for confinement times on the lower side.

The scaling laws for the confinement time can now be used to validate the model predictions. To do this, one condition is varied, the others are kept at a fixed value. The fixed values are again the values from Table 5. These values should give mostly H-mode samples, so the comparison is made to the scaling law given in equation (2.18). Some of the conditions are used in this scaling law, as well as the average electron density. As the electron density is an output of the ReD-VAE model, the average density can be calculated and used in the scaling law. The results are plotted as confinement time against $y_i^{\alpha_i}\overline{n_e}^{0.41}$, where y_i is the varied condition and α_i is the relevant exponent in equation (2.18). As all other factors in the scaling law are constant, this should result in a linear plot, and linear regression can be used to find the predicted scaling law. The results for I_p and B_t are given in Figures 32 and 33 respectively.

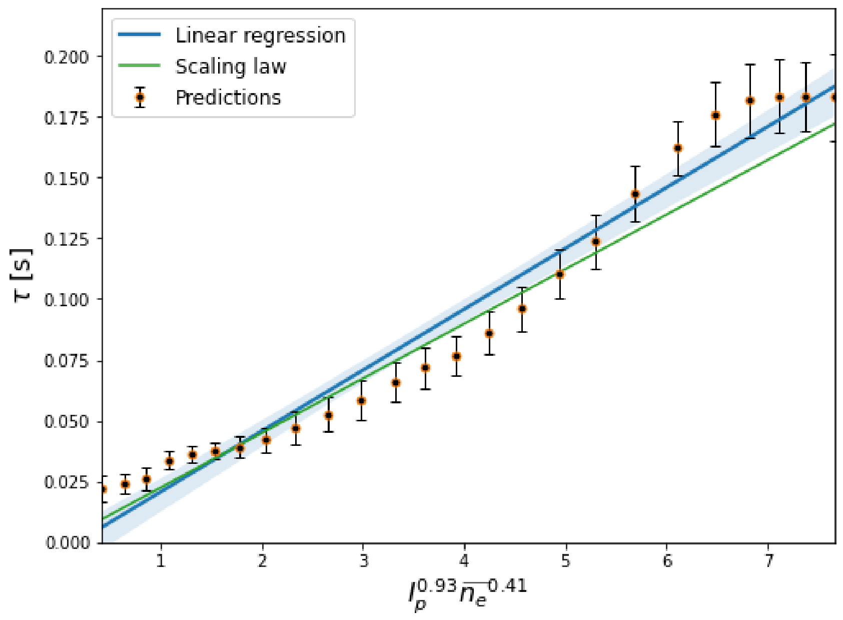


Figure 32 – Comparison of the scaling law for the plasma current by linear regression on ReD-VAE predictions (blue) with the ITERH-98P(y,2) scaling law (green). Excellent agreement between the scaling law and ReD-VAE predictions is observed.

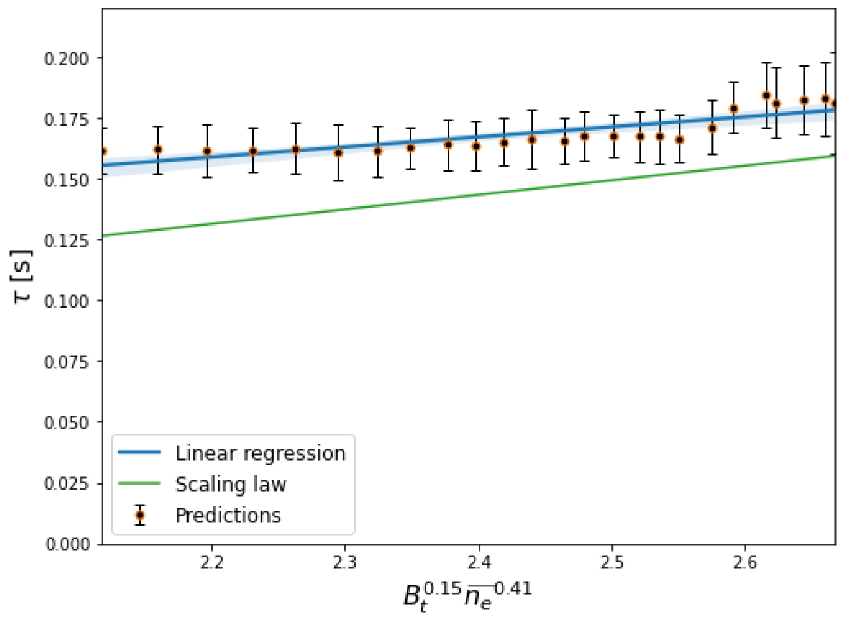


Figure 33 – Comparison of the scaling law for the toroidal magnetic field that by linear regression on ReD-VAE predictions (blue) with the ITERH-98P(y,2) scaling law (green). The model somewhat overestimates the expected scaling here, but the slopes are still very similar.

These figures both show a good agreement of predictions with the previously developed empirical scaling law. Especially for the plasma current, the predictions closely follow the ITERH-98P(y,2) scaling law. Here, based on equation (2.18), a slope of 0.0224 is expected. Using the linear regression model on ReD-VAE outputs, a slope of 0.0249 is found. This shows an incredible agreement between model predictions and empirical results from literature.

For the toroidal magnetic field, the model predictions are somewhat higher than the scaling law, but the values are still close and the slopes are similar. The scaling-law-based approximate slope expectation is 0.0597, and based on linear regression 0.0401 is found. However, because only a small range of values is covered on the horizontal scale in Figure 33, this difference in slopes does not lead to a big difference in predicted confinement times. Therefore, this is still an acceptable agreement, considering that some approximations were made in the calculation of the confinement time, as well as in the calculation of the scaling law, which is an experimental approximation in itself.

Now that it has been confirmed that sensible predictions for the confinement time, and how this changes when varying conditions, can be made using the ReD-VAE, the model can be used to get an estimate of how this behavior is for other conditions, for which there is no scaling law.

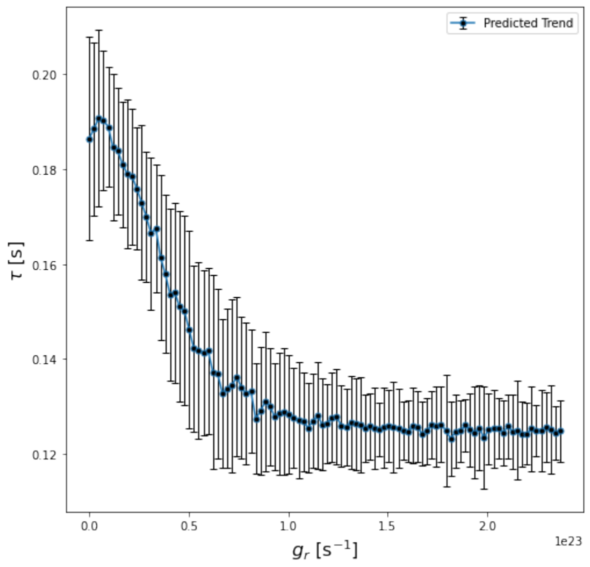
8.4.4 Gas Input Optimization

For the gas input parameters (rate and total dosage), there is no scaling law that can be used to predict the confinement time with respect to these parameters. As these parameters are used as conditional inputs, the ReD-VAE can be applied to optimize the gas input parameters.

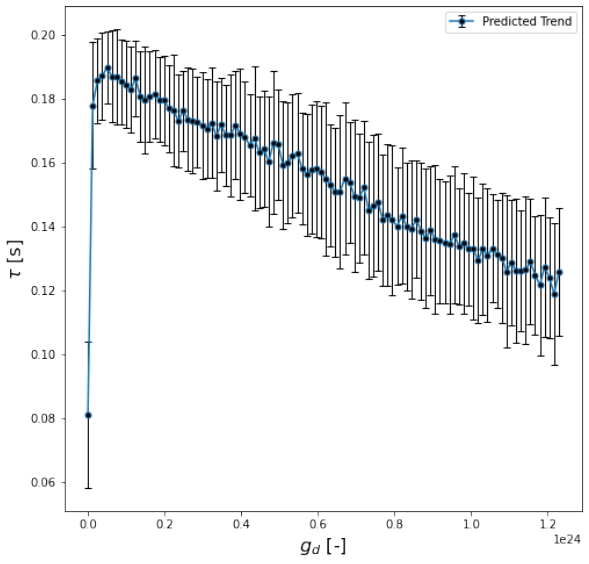
Suppose the scenario where all other conditions are known, but the gas parameters have to be optimized. For the known conditions, the values in Table 5 are used. One of the gas parameters is varied over 100 values in the range of the minimum to the maximum value in the dataset, and for each of the values 100 calculations of the expected confinement time are made. From this, the mean and standard deviation of the confinement time can be predicted. The results of this procedure are given in Figure 34.

After a short increase, the predicted confinement time generally decreases with increasing gas rate (Figure 34a). After this initial decrease, it appears to stabilize. For this second part, with high gas rates, the training data is very sparse. That could lead to the constant predictions. For the total gas dosage (Figure 34b), similar behavior is seen: after a maximum in confinement time for a low gas dosage, the confinement time decreases. This Figure also shows that for the minimum gas dosage, confinement time sharply decreases, most likely due to too little gas being in the reactor to form a stable plasma.

These two input parameters can also be varied simultaneously. This is presented in Figure 35. By variation of both parameters over 100 steps, a 100×100 grid of parameter combinations is created. For each point in the grid, the confinement time is predicted 100 times, and the mean is given in the plot.



(a) Gas input rate.



(b) Gas input dosage.

Figure 34 – Confinement time predictions for variation of a single gas input parameter.

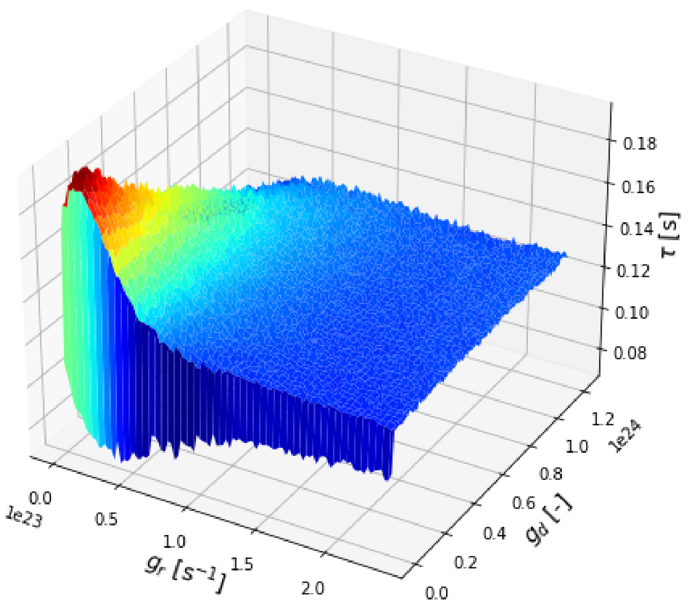


Figure 35 – Confinement time predictions for variation of both gas input parameters. Highest confinement time is expected when both gas rate and total dosage are low.

Similar results are obtained as in the one-dimensional cases. For low gas rate and gas dosage, maximum confinement time is expected. Again, when the gas dosage is decreased too much, the confinement decreases sharply.

These types of results can be used to optimize input parameters for which no empirical scaling laws have yet been developed, which could be one of the main applications of ReD-VAE. In the following sections, some additional applications are explored, that are not directly related to the generation of new data points.

8.5 Condition Regression

The ReD-VAE structure, as described in section 5, involves auxiliary regression networks. During training, these auxiliary regression networks predict the value of a condition y_i from the corresponding conditional latent space representation $\mathbf{z_{y_i}}$. This is what forces the conditional latent spaces to disentangle the conditional information. An additional benefit of this structure is that the auxiliary networks can be used during inference as well. Given a sample $\mathbf{x}$, the encoders can be used to obtain the latent representation in $\mathbf{z_{y_i}}$, and after this, the auxiliary network can predict the condition y_i. Such an application can be useful for missing data imputation, where the experimental profiles are available, but the value of the conditions is, for whatever reason, unknown.

Table 6 gives the results for this regression. In the table, the median absolute error is provided. Because of the difference in the order of magnitude between the conditions, the median absolute error is also provided relative to the maximum value of the condition. The magnitude of the relative error to the predicted condition could not be provided as the conditions often have a value of 0, leading to infinite relative errors.

Table 6 – Median regression error and median regression error relative to the maximum absolute value for each of the conditions.

Condition	Median Error	Relative to max
B_t	0.416 T	12.0%
I_p	0.236 MA	6.23%
P_{NBI}	1.68 MW	6.37%
P_{ICRH}	0.586 MW	7.36%
M	0.476 u	15.9%
δ_u	0.0388	6.71%
δ_l	0.0359	7.39%
g_r	$4.02 \cdot 10^{21}$ s^{-1}	1.76%
g_d	$5.09 \cdot 10^{22}$	4.86%
R_i	7.01 mm	0.282%
Z_i	52.9 mm	3.05%
R_o	56.3 mm	1.91%
Z_o	33.0 mm	1.90%

Since the errors are generally quite small in comparison to the order of magnitude of the relevant condition, using these regression networks can be a viable method for missing data imputation. However, the KL-regularization of the latent spaces actively counter-acts the quality of these predictions. An unbalanced model, with very little KL-divergence, can make much better predictions of the input parameters, at the cost of the quality of generated samples. The result for the unbalanced model is presented in appendix B. This shows the importance of balancing the loss weight parameters for specific applications.

8.6 Discovery of Hidden Variables

The $\mathbf{z_x}$ latent space that is not connected to the conditions should contain all information that is relevant for data reconstruction and generation that is not contained in the conditional $\mathbf{z_y}$ spaces. This means that any hidden variable, not included as a condition, should turn up in the $\mathbf{z_x}$ space.

To further investigate these hidden variables, predictions are made using the encoders for the $\mathbf{z_x}$ representation for all training data. On this latent representation, which is 32 dimensional, a k-means clustering algorithm [57] is performed, with two clusters, to find the two main groups of samples in the dataset. Our expectation is that the two main groups are the H- and L-mode samples, so a comparison is made between the clusters and the H-mode label from the dataset, where a value 1 represents H-mode and 0 is L-mode. In Figure 36 the results are given, where the 32 dimensional $\mathbf{z_x}$ space is plotted in 2 dimensions by using a t-SNE reduction algorithm [58].

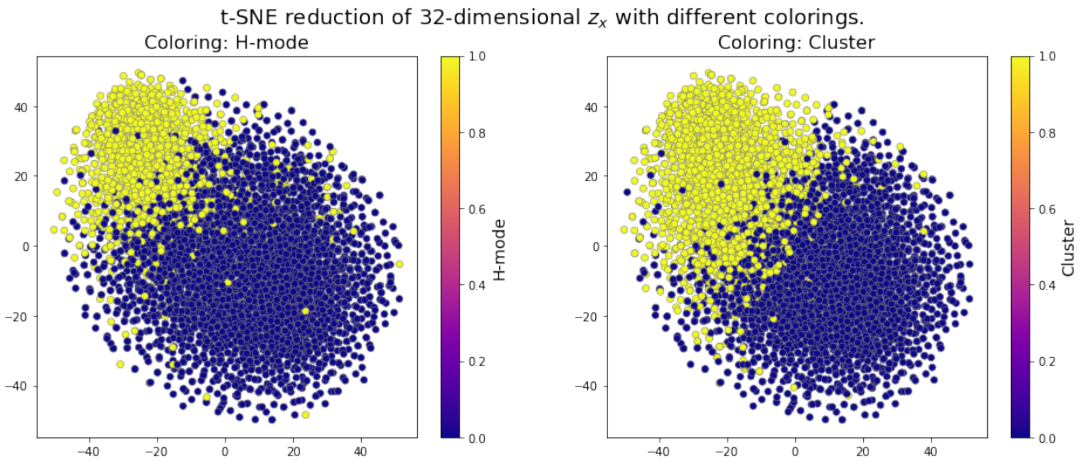


Figure 36 – Comparison of clusters in t-SNE reduction of $\mathbf{z_x}$ latent space with the H/L-mode samples. This shows that the clustering algorithm largely finds the H/L groups in the samples, with a match of 86.3%.

Figure 36 shows that the clustering algorithm finds the two groups of H- and L-mode samples for a large part. Due to the inherent randomness of the latent space, there is some variance in this, but after 100 experiments it was found that the average accuracy here is 86.1% with a standard deviation of 0.65. This is a high accuracy, especially when taking into account that the original labels are not fully correct, as will be discussed in section 8.7. Correct labels could further increase this accuracy. To further investigate the hidden variable, a comparison of the difference in clusters is necessary. Figure 37 shows the difference in cluster center for each of the latent dimensions. This indicates dimension 20 as the most important dimension for these clusters.

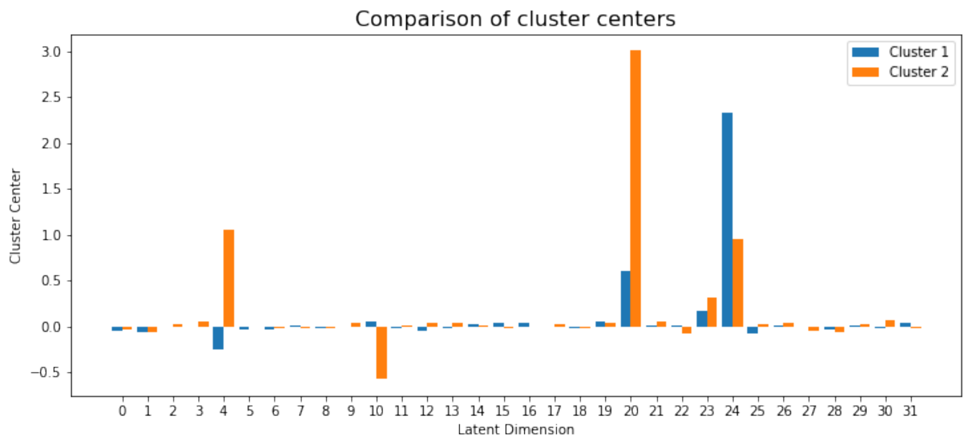


Figure 37 – Centers of the two clusters for each of the latent dimension. The biggest difference is observed for dimension 20.

To gain some insight into dimension 20, a plot can be made of the consequences of varying this latent variable. To this end, the latent representation of a random sample is taken. Then, while all other latent variables are kept fixed, variable 20 is uniformly varied over a range of values. For each value, the decoder is used to generate the T_e, n_e and T_i profiles. The result is provided in Figure 38, where the legend indicates the value that is used for latent dimension 20.

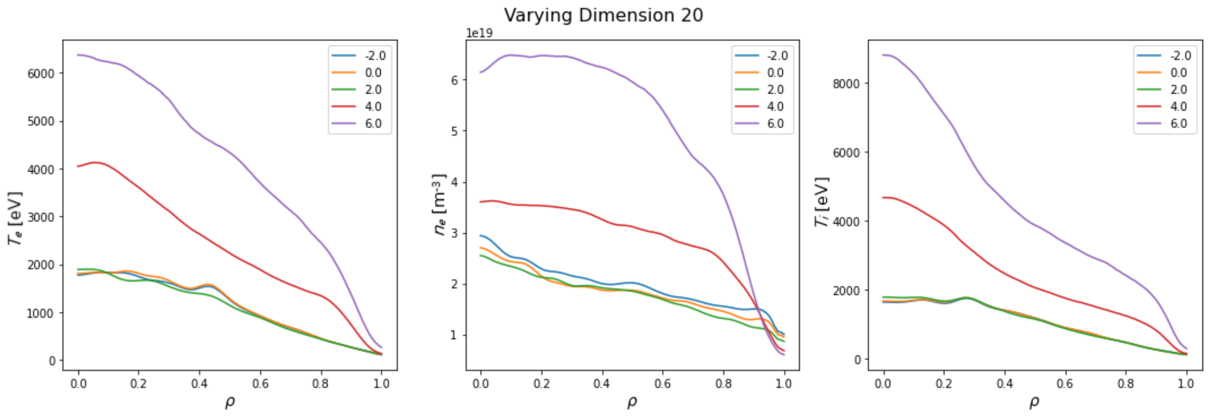


Figure 38 – Effect of varying latent $\mathbf{z_x}$ variable 20 on the profiles in T_e, n_e and T_i. The legend gives the value of the latent variable. This shows that when the latent value is increased past a certain point, a sudden jump is made and a pedestal region forms in the plasma boundary.

When increasing the latent variable, we see that first there is very little difference for the first steps: this corresponds to the plasma remaining in L-mode. Then, when the value is increased past 2, there is a sudden jump, with an increase in temperatures and density, and a pedestal region starts to form in the core region between $\rho = 0.85$ and $\rho = 1$. These are clear signs corresponding to an H-mode plasma. This is further confirmed when making a scatter plot of the representation in latent space for latent dimension 20, and coloring with the H-mode label, as shown in Figure 39. Here, a clear boundary around $z_{x,20} = 2$ is seen, where for higher values almost only H-mode plasmas are encountered.

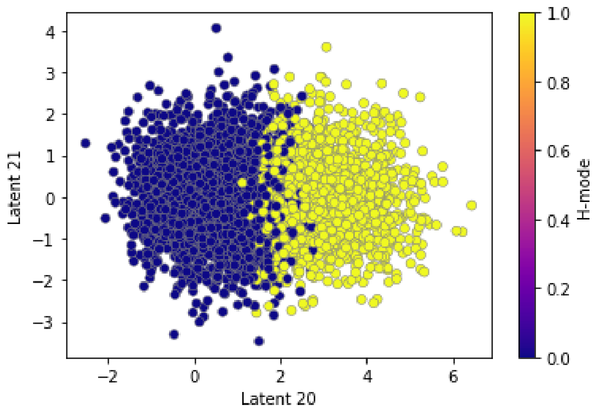


Figure 39 – Scatter plot for the latent representation in dimensions 20 and 21. For $z_{x,20} \geq 2$, almost all samples are H-mode plasmas.

As no information regarding H/L-mode was provided to the model, this can be regarded as a hidden variable. This result serves as a proof-of-principle, that shows that the generative model can be applied to find these hidden variables, by clustering on the non-conditional latent space, and then investigating the effects of changing important latent variables.

8.7 Mode Classification

Instead of doing unsupervised parameter discovery for H- and L-mode, another option is to train a classification network in a supervised manner, as the labels for these samples are available in the dataset. For this, a simple, sequential classification network was constructed, as given in Table 7. The network was trained for 200 epochs, for which the history is given in Figure 40

Table 7 – Architecture of the H/L-mode classification network

Layer Number	Layer Type	Parameters
1	Dense	Nodes = 64; Activation = ReLU
2	Dropout	Rate = 0.5
3	Dense	Nodes = 8; Activation = ReLU
4	Dense	Nodes = 2; Activation = Sigmoid

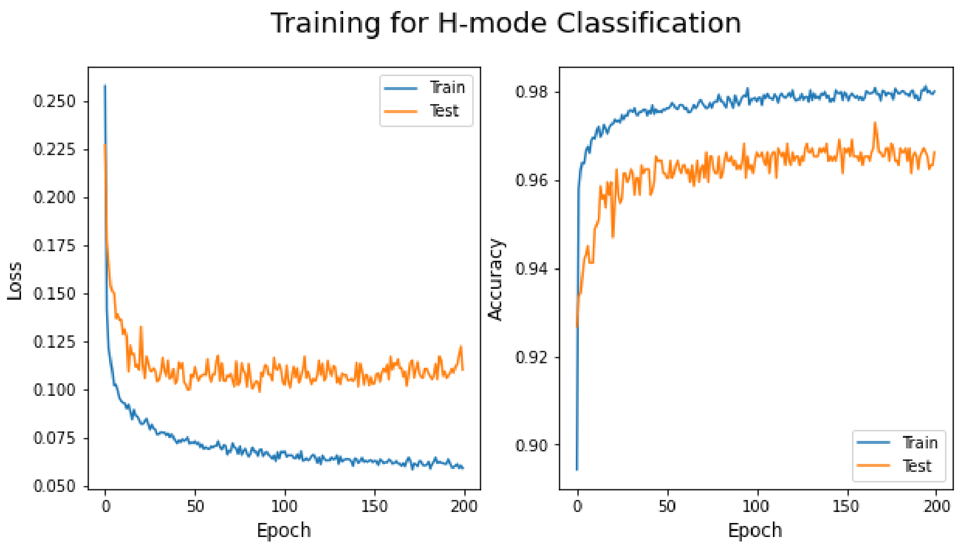


Figure 40 – Training history for the H/L-classification network.

After training, an accuracy of around 96% is obtained. However, the reason that some of the testing samples appear to be 'misclassified' is because they were mislabeled in the first place. An example of such a sample is given in Figure 41.

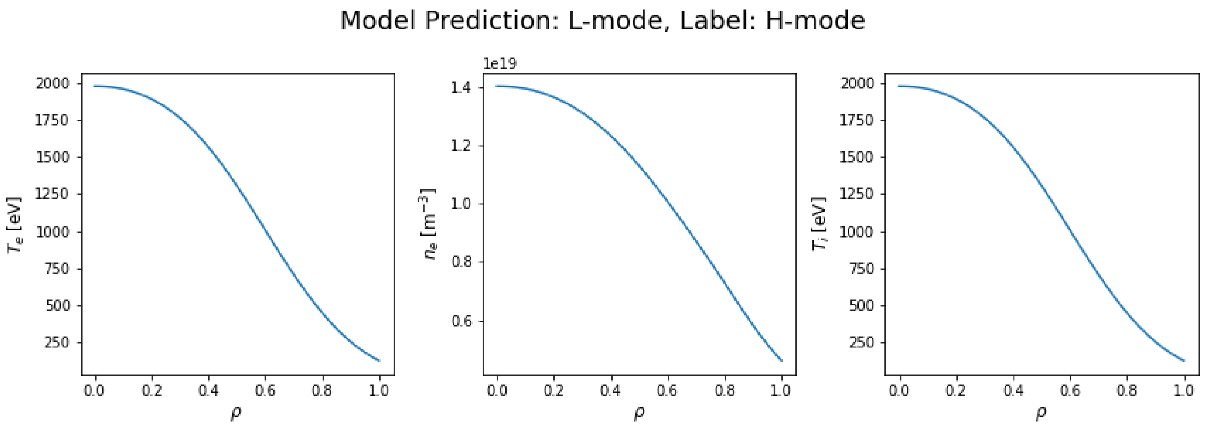


Figure 41 – Example where the classification prediction deviates from the label. The sample was labeled to be H-mode, but after re-examination appears to be in L-mode, as predicted by the classification network.

In this figure, a sample is given that was labeled as H-mode. However, with relatively low core temperatures and the absence of a pedestal in the outer region, this sample actually appears to be an L-mode plasma, which is the label that is predicted for this sample by the classification model. This type of 'misclassification', where the label was assigned wrongly, happened in around half of the misclassified samples. This shows the final application for the generative model: the model can be used to improve current labeling heuristics, as it can be worthwhile to check the correctness of the label when the network predicts a different label.

9 Discussion

9.1 Simulated Data

In the first part of this work, the application of a VAE to a clean, simulated dataset is presented. It was shown that in this simple setting, the VAE works exactly as expected. The latent space variables identify relevant information that describes the differences between the samples, i.e. the boundary conditions. This is demonstrated by high correlation values between latent variables and boundary conditions, with a Pearson correlation coefficient ρ up to 0.88, indicating a very strong relation. It was also shown that sampling the prior distribution yields samples following the distribution from the data. This shows that the VAE is suitable to model the probability distribution of tokamak data.

Then, by adding conditions in a CVAE model, the latent space becomes more relaxed. If one of the conditions is added, this information disappears from the latent space, as it's no longer necessary information, and minimizing the KL-divergence requires minimal information in the latent space. Having one of the conditions fixed decreases the predicted standard deviation, which is of course also the expectation. However, the limited size of the dataset makes it very hard to validate the magnitude of this decreased standard deviation.

With full boundary conditions as additional information, full latent space collapse happens. In the setting of this simulated dataset, the three boundary conditions give full information regarding the sample and no variation is expected. Therefore, the latent space should not contain any information, and the posterior for the latent space simply becomes the prior. This also means that the decoder largely ignores the latent space, and instead only uses the conditional information. This leads to negligible variance in the predictions for full conditions, which is exactly what should be happening, as boundary conditions are the only source of variation in this dataset.

Furthermore, observations were made that with full knowledge of the conditions, accurate predictions can be made of the expected profiles. With a relative error magnitude of only 1.38% on average on unseen testing data, these predictions are very reliable. These predictions can then be used to gain some insight into the behavior of the output profiles for changing boundary conditions. Qualitatively, this behavior is similar to what was found in other literature, but quantitative comparisons are hard to make, because of the limited size of this dataset and therefore the limited amount of generalization that can be expected. Another limitation of the model is in the extrapolation of the conditions. Once far outside of the training range, the predictions become unreliable.

Despite these limitations, the main goal of the first part has been achieved, and the first research question has been answered. We have shown that predictions are of high quality, standard deviations are modeled correctly, and that the latent space contains all necessary information but not more than that. This paves the way towards the application of VAE models in a more rough, experimental setting, which is what is discussed next.

9.2 Experimental Data

The second set of experiments was performed on an experimental dataset. Because this set has a lot more variance, both due to known and unknown factors, as well as measurement noise, a model with more capabilities is required here. Therefore, we have introduced the ReD-VAE model, that uses regressive networks to disentangle conditional factors into separate latent spaces, while retaining one high-dimensional latent space to account for all of the variance due to unknown factors. The generative capabilities of this model are improved with respect to a normal VAE or the DIVA [49] model from which it was adapted by adding a conditional generation loss term. Using the KL_{sum} evaluation metric, it was irrefutably shown that this addition has a positive effect on the generative qualities of the ReD-VAE. Similarly, the positive effect of semi-supervised training was demonstrated for this model.

Using the model optimized with these parameters, conditional predictions can be made. Here, the variance from the resulting posterior predictions, as well as from the $\mathbf{z_x}$ latent prior, should provide an estimate of the variance that is expected for a given set of experimental conditions. As there is information that is unknown to the model, the mean of the conditional prediction is not expected to correspond exactly with the testing sample that it's compared to. Rather, the testing sample should be a sample of the distribution that is predicted by the ReD-VAE model. Under this assumption, around 68% of the samples should be within one standard deviation from the predicted mean, with 95% being in two standard deviations. The frequencies of this that were found, however, are only around 40% and 78%, respectively. This could be due to the counting being too strict, as we showed that there are samples that still appear quite good, but are for some part outside of the standard deviation, and thus will not be counted. It is hard to quantify the quality of the generation with an intuitive measure, as our main quantification, the summed KL-divergence, is far from intuitive. However, the quality of the generated samples is good enough that insights can be obtained into how the experimental profiles change, when varying the conditions. The ReD-VAE model thus provides a new, powerful method to infer experimental results, based on machine conditions, including the impact of hidden variables. Within the boundaries of existing data-sets (as extrapolation capabilities are limited), the method facilitates convenient data-driven experimental design.

In principle, this can also be done by multi-physics modeling, like JETTO and other physics-based models, as long as all physics is included and accurate. However, this is only the case for a sub-section of the plasma. For the core region, within the pedestal, modeling is accurate but can take a few days. This can be accelerated by using machine learning for surrogate models, but the ReD-VAE predictions are still faster than that. Additionally, ReD-VAE modeling can provide insight on aspects that are less well modeled by theory, such as the pedestal region and the impact of gas input and strike point parameters. Integrated modeling for this can take weeks for a single case, with still many assumptions made. Nonetheless, theory-based modeling can extrapolate better to new regimes, hence a combination of the two approaches would be beneficial.

When predicting the stored energy W, the model generally seems to follow a curve similar to the distribution of the data. For one of the conditions, the toroidal magnetic field B_t, a divergence from this data trend was observed. As the stored energy is expected to be almost constant for varying B_t, this divergence shows that the model is able to largely disentangle the effects of changing magnetic field and plasma current. The apparent distribution in the data is suspected to mostly originate from the correlation of B_t with I_p, and the fact that the model diverges from this, by instead predicting a constant trend, shows this disentanglement. However, we have also seen that the model can overestimate the increase of electron density for increasing plasma current, due to a correlation between the plasma current and gas input parameters in the training data. Adjusting biases in the training data to account for correlations could improve the disentanglement of these effects.

The main comparison with literature, through empirical scaling laws, and the main testament to the workings of the ReD-VAE structure, is made by training a model that also predicts j_{ohm} and Z_{eff}. By having these extra outputs, predictions for the confinement time can be made, which can be compared with known scaling laws for the confinement time. The agreement between predictions and theory is remarkably good. When varying the plasma current, almost exact correspondence with the expectations from the scaling law was found. Variation of the magnetic field strength also yields good results, but the correspondence is somewhat farther off. Overall, this shows that the ReD-VAE is able to pick up on confinement time trends in the data. This model was trained in a completely 'physics-unaware' method, as no knowledge of background physics was used during training. The fact that it still identifies correct trends for the confinement time, shows the power of these generative models. This also suggests that sensible predictions can be made for other variables, for which the theory might not be available. This was shown in a use case for the gas input parameters. For a given set of other, fixed conditions, the ReD-VAE predicts optimal confinement time for low values of both gas input rate and dosage. The investigation of this use case demonstrates one of the most useful applications of a generative model, where input parameters can quickly be optimized to improve the efficiency of a fusion reactor, where theoretic considerations are insufficient.

After investigating the generative properties of the ReD-VAE model, some of the additional usages that this generative model supplies are discussed. First of all, the application of using the auxiliary regression networks to perform inference on the conditions was demonstrated. By supplying the auxiliary networks with the output from the encoders, an estimate for the conditions that were used to create the sample can be obtained. The quality of these predictions is different for each condition, but for most of them, a good estimate can be given. However, the KL-divergence regularization on the latent spaces makes this regression much harder. A very unbalanced model, with little KL divergence, can produce much better regression results, as is shown in Table 8 in Appendix B. Here, very accurate predictions can be made, as there is much more freedom in the latent space to separate samples. However, this makes the generative properties of the model much worse, as shown in Table 9. As most of the focus in this thesis is on generative properties, optimization was performed with regard to the KL_{sum} metric. However, if a model is desired that is particularly good at conditional inference, another optimization method could be better. Careful balancing of all qualities of the model is one of the most difficult parts of using this model, and this is one of the areas where there might be improvements possible, as will be discussed later.

The focus was then shifted from the conditional $\mathbf{z_y}$ spaces to the non-conditional $\mathbf{z_x}$ space. This space should contain the relevant information that is not directly related to the conditions that are used. The expectation is that two main clusters are found in this space, with H- and L-mode plasmas separated, as this causes the largest difference between the samples. The plasma mode is considered a hidden parameter, as this was not included in the model training data. By using a k-means clustering algorithm, these two groups can be found in an unsupervised way with an accuracy of around 86%. This shows that the model can be applied to find hidden parameters, by analysis of the $\mathbf{z_x}$ space. By finding the latent dimension in which the clusters differ most, and reconstructing profiles for different values of this specific latent parameter, the effect of the H-mode transition can be clearly reconstructed, with increased experimental output values, and pedestal regions in the profiles. Again, by using a differently balanced model, a higher accuracy could be obtained, at the cost of other performances. An accuracy of up to 93% has been observed.

Finally, instead of unsupervised clustering, a classification model can be trained on the $\mathbf{z_x}$ latent space to predict whether a sample will be H- or L-mode. A high accuracy of 96% was obtained here. For some of the samples where the model prediction is different from the label, it turned out that the label is wrong. By combining the classification model with the current labeling heuristics, an improved labeling method can be formed, where in case of disagreement between the two labels, human experts could come in to determine the correct label.

All in all, a lot of possible applications were demonstrated using the ReD-VAE model, mainly in the sense of a proof-of-principle. Further work will be required to gain in-depth knowledge of the specific applications, where the loss parameter balancing plays a big role. If one specific application is desired from the generative model, it should be balanced in such a way as to optimize this specific application. This optimization procedure could be improved, as brute-force hyperparameter searches are very time-consuming. In recent work, suggestions are made to automate the optimization of loss parameters for regular VAEs [51, 52, 53]. These methods were not directly applicable for the ReD-VAE structure, but a similar approach could be very helpful in speeding up one of the most tedious parts of the process. This could help make the model generalize more easily, as now, for example, when adding the outputs of j_{ohm} and Z_{eff}, some re-optimization was needed for the model. If this could happen automatically, it would be easier to apply in different situations.

Some benefits could also be obtained by downsampling the experimental data in preprocessing. The experimental profiles used here have a resolution of 101 points. A lower resolution can still be sufficient to convey the relevant information of these profiles. Downsampling in preprocessing could potentially lead to faster training and better optimization.

When, in future work, the ReD-VAE structure is further used, some attention should also be paid to the dataset on which it is trained. The dataset used in this work contains only stable plasmas. However, it is very possible that for certain sets of conditions, the plasma becomes unstable and actually cannot be sustained. The dataset strongly biases the model, in the sense that in the context of this ReD-VAE model, unstable plasmas do not exist, as those were never shown, and therefore a prediction for such a plasma will never be made. This could lead to unrealistic predictions. If the model is trained on a dataset that contains all experimental findings, rather than only successful ones, the predictions would be more reliable. This could also enable research possibilities investigating what causes these disrupted plasmas.

It was also observed that, for conditional generation, the model performs best in parts of the condition-space that are densely populated. Training appears to work best if the model is supplied with as many samples as possible, so large datasets are preferable for all future work. Large datasets could further improve results, as of course variances can only be modeled to the extent that they are represented in the data. Also, strong correlations between different conditions can make it difficult to disentangle the effects of these conditions. Extending datasets with more samples to reduce these correlations can help with this.

When making predictions for conditions far outside the hypercube of training conditions, non-reliable predictions are made. This is one of the weaknesses of the model, where large extrapolation gives unreasonable results. This could be improved in future work by making the model physics-aware for specific applications. By integrating first principles into the model, more consistent results are expected both within and outside of the training data ranges.

10 Outlook and Conclusion

Looking back at the first research question and its sub-questions, these were extensively answered in the first part of the thesis. It was shown that for such a simple dataset, a CVAE has adequate modeling power to generate realistic synthetic results, corresponding well to what is expected from a conventional simulation suite. For a full conditional model, very accurate predictions can be made, and when not all conditions are supplied, the latent space takes on the role of all unknown variables.

The secondary research question was answered by the results obtained with the powerful ReD-VAE model. The disentangled latent spaces give a lot of flexibility regarding the input to the model. By training in a semi-supervised fashion, with the additional loss terms from the ReD-VAE model, the model can be optimized well for the conditional generation of data, which can be evaluated using the KL_{sum} evaluation metric. This metric, while somewhat non-intuitive, leads to a model that can be used to quickly generate reliable data. By comparing this with the original dataset, and empirical scaling laws, the model was shown to be useful for a prediction of how reactor performance can be affected by changing input parameters, for which there might not be any theoretical alternatives. Furthermore, by using the information in the latent space, a proof-of-principle was presented of discovering hidden parameters, with H/L-mode as an example. The latent space information can also be used in other ways, like sample classification, or inference of the conditions.

Although the ReD-VAE can be very useful for multiple applications, there are some downsides. With limited extrapolation possibilities, this modeling method is not reliable for predictions far outside the range of training data. Furthermore, a biased dataset can lead to biased predictions, as this model will always predict a stable plasma, which is not realistic. In follow-up work, more extensive datasets could supply valuable additional information. Another improvement could be in automatically learning the loss weight parameters, as this is the most inflexible part of this model. Lastly, incorporating first principle physics into the model architecture could improve extrapolation capabilities. By improving these points, or optimizing the model for specific applications, deeper research into using ReD-VAE for a specific application is expected to yield valuable results.

We conclude that ReD-VAE shows tremendous potential in several applications for tokamak research. If future work automates the process of loss weight optimization, the ReD-VAE would not only be very powerful, but also very flexible. This structure can then be applied in not only tokamak fusion simulation, but for all sorts of experimental research, where output profiles are generated corresponding to some set of input conditions. This makes the results in this work valuable for a wide range of fields of research.

References

[1] D. Fasel and M. Q. Tran. Availability of lithium in the context of future d–t fusion reactors. *Fusion engineering and design*, 75:1163–1168, 2005.

[2] N. Holtkamp et al. An overview of the iter project. *Fusion Engineering and Design*, 82(5-14):427–434, 2007.

[3] K. L. van de Plassche, J. Citrin, C. Bourdelle, Y. Camenen, F. J. Casson, V. I. Dagnelie, F. Felici, A. Ho, S. Van Mulders, and JET Contributors. Fast modeling of turbulent transport in fusion plasmas using neural networks. *Physics of Plasmas*, 27(2):022310, 2020.

[4] JET Team et al. Fusion energy production from a deuterium-tritium plasma in the jet tokamak. *Nuclear Fusion*, 32(2):187, 1992.

[5] J. D. Lawson. Some criteria for a power producing thermonuclear reactor. *Proceedings of the physical society. Section B*, 70(1):6, 1957.

[6] T. Tanabe. *Tritium: Fuel of fusion reactors*. Springer, 2017.

[7] K. Haupt. Fusion machines: Searching for the perfect shape, Jun 2018. `https://www.iter.org/newsline/-/3037`, ITER.

[8] Triangularity, Jun 2014. `http://fusionwiki.ciemat.es/wiki/Triangularity`, FusionWiki.

[9] L. Spitzer. *Physics of fully ionized gases*. Courier Corporation, 2006.

[10] J. Wesson and D. J. Campbell. *Tokamaks*, volume 149. Oxford university press, 2011.

[11] D. Bruno, M. Capitelli, C. Catalfamo, and A. Laricchiuta. Cutoff criteria of electronic partition functions and transport properties of atomic hydrogen thermal plasmas. *Physics of Plasmas*, 15:112306–112306, 11 2008.

[12] G. Duesing, H. Altmann, H. Falter, A. Goede, R. Haange, R. S. Hemsworth, P. Kupschus, D. Stork, and E. Thompson. Neutral beam injection system. *Fusion Technology*, 11(1):163–202, 1987.

[13] D. F. H. Start, J. Jacquinot, V. Bergeaud, V. P. Bhatnagar, G. A. Cottrell, S. Clement, L. G. Eriksson, A. Fasoli, A. Gondhalekar, C. Gormezano, et al. Dt fusion with ion cyclotron resonance heating in the jet tokamak. *Physical review letters*, 80(21):4681, 1998.

[14] M. Keilhacker. H-mode confinement in tokamaks. *Plasma Physics and Controlled Fusion*, 29(10A):1401, 1987.

[15] E. J. Doyle, W. A. Houlberg, Y. Kamada, V. Mukhovatov, T. H. Osborne, A. Polevoi, G. Bateman, J. W. Connor, J. G. Cordey, T. Fujita, et al. Plasma confinement and transport. *Nuclear Fusion*, 47(6):S18, 2007.

[16] The jet project. eur-jet-r7, 1975.

[17] S. M. Kaye, M. Greenwald, U. Stroth, O. Kardaun, A. Kus, D. Schissel, J. DeBoo, G. Bracco, K. Thomsen, J. G. Cordey, et al. Iter l mode confinement database. *Nuclear Fusion*, 37(9):1303, 1997.

[18] G. Cenacchi and A. Taroni. Jetto: a free boundary plasma transport code. Technical report, ENEA, 1988.

[19] C. Bourdelle, X. Garbet, F. Imbeaux, A. Casati, N. Dubuit, R. Guirlet, and T. Parisot. A new gyrokinetic quasilinear transport model applied to particle transport in tokamak plasmas. *Physics of Plasmas*, 14(11):112501, 2007.

[20] J. Zou, Y. Han, and S. So. Overview of artificial neural networks. In *Artificial Neural Networks*, pages 14–22. Springer, 2008.

[21] K. O'Shea and R. Nash. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*, 2015.

[22] V. Dumoulin and F. Visin. A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*, 2016.

[23] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[24] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[25] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[26] J. Shlens. Notes on kullback-leibler divergence and likelihood. *arXiv preprint arXiv:1404.2000*, 2014.

[27] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. Beta-vae: Learning basic visual concepts with a constrained variational framework. 2016.

[28] S. Zhao, J. Song, and S. Ermon. Infovae: Information maximizing variational autoencoders. *arXiv preprint arXiv:1706.02262*, 2017.

[29] D. R. Ferreira, P. J. Carvalho, and H. Fernandes. Deep learning for plasma tomography and disruption prediction from bolometer data. *IEEE Transactions on Plasma Science*, 48(1):36–45, 2019.

[30] D. R. Ferreira, P. J. Carvalho, C. Sozzi, P. J. Lomas, and JET Contributors. Deep learning for the analysis of disruption precursors based on plasma tomography. *Fusion Science and Technology*, 76(8):901–911, 2020.

[31] J. An and S. Cho. Variational autoencoder based anomaly detection using reconstruction probability. *Special Lecture on IE*, 2(1):1–18, 2015.

[32] O. Cerri, T. Q. Nguyen, M. Pierini, M. Spiropulu, and J. R. Vlimant. Variational autoencoders for new physics mining at the large hadron collider. *Journal of High Energy Physics*, 2019(5):36, 2019.

[33] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27:2672–2680, 2014.

[34] S. Ravanbakhsh, F. Lanusse, R. Mandelbaum, J. Schneider, and B. Poczos. Enabling dark energy science with deep generative models of galaxy images. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[35] M. Paganini, L. de Oliveira, and B. Nachman. Accelerating science with generative adversarial networks: an application to 3d particle showers in multilayer calorimeters. *Physical review letters*, 120(4):042003, 2018.

[36] L. de Oliveira, M. Paganini, and B. Nachman. Learning particle physics by example: location-aware generative adversarial networks for physics synthesis. *Computing and Software for Big Science*, 1(1):4, 2017.

[37] P. Musella and F. Pandolfi. Fast and accurate simulation of particle detectors using generative adversarial networks. *Computing and Software for Big Science*, 2(1):8, 2018.

[38] B. Kim, V. C. Azevedo, N. Thuerey, T. Kim, M. Gross, and B. Solenthaler. Deep fluids: A generative network for parameterized fluid simulations. In *Computer Graphics Forum*, volume 38, pages 59–70. Wiley Online Library, 2019.

[39] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath. Generative adversarial networks: An overview. *IEEE Signal Processing Magazine*, 35(1):53–65, 2018.

[40] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[41] W. Lee, D. Kim, S. Hong, and H. Lee. High-fidelity synthesis with disentangled representation. *arXiv preprint arXiv:2001.04296*, 2020.

[42] K. Cranmer, J. Brehmer, and G. Louppe. The frontier of simulation-based inference. *Proceedings of the National Academy of Sciences*, 2020.

[43] P. De Castro and T. Dorigo. Inferno: inference-aware neural optimisation. *Computer Physics Communications*, 244:170–179, 2019.

[44] B. Uria, M. A. Côté, K. Gregor, I. Murray, and H. Larochelle. Neural autoregressive distribution estimation. *The Journal of Machine Learning Research*, 17(1):7184–7220, 2016.

[45] M. J. Vowels, N. C. Camgoz, and R. Bowden. Targeted vae: Structured inference and targeted learning for causal parameter estimation. *arXiv preprint arXiv:2009.13472*, 2020.

[46] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling. Improved variational inference with inverse autoregressive flow. *Advances in neural information processing systems*, 29:4743–4751, 2016.

[47] L. Dinh, D. Krueger, and Y. Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

[48] O. Meneghini, S. P. Smith, P. B. Snyder, G. M. Staebler, J. Candy, E. Belli, L. Lao, M. Kostuk, T. Luce, T. Luda, et al. Self-consistent core-pedestal transport simulations with neural network accelerated models. *Nuclear Fusion*, 57(8):086034, 2017.

[49] M. Ilse, J. M. Tomczak, C. Louizos, and M. Welling. Diva: Domain invariant variational autoencoders. In *Medical Imaging with Deep Learning*, pages 322–348. PMLR, 2020.

[50] D. P. Kingma, S. Mohamed, D. Jimenez Rezende, and M. Welling. Semi-supervised learning with deep generative models. *Advances in neural information processing systems*, 27:3581–3589, 2014.

[51] A. Asperti and M. Trentin. Balancing reconstruction error and kullback-leibler divergence in variational autoencoders. *arXiv preprint arXiv:2002.07514*, 2020.

[52] B. Dai and D. Wipf. Diagnosing and enhancing vae models. *arXiv preprint arXiv:1903.05789*, 2019.

[53] S. Lin, S. Roberts, N. Trigoni, and R. Clark. Balancing reconstruction quality and regularisation in elbo for vaes. *arXiv preprint arXiv:1909.03765*, 2019.

[54] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, pages 265–283, 2016.

[55] A. Ho, J. Citrin, F. Auriemma, C. Bourdelle, F. J. Casson, H. Kim, P. Manas, G. Szepesi, H. Weisen, and JET Contributors. Application of gaussian process regression to plasma turbulent transport model validation via integrated modelling. *Nuclear Fusion*, 59(5):056007, 2019.

[56] O. Linder, J. Citrin, G. M. D. Hogeweij, C. Angioni, C. Bourdelle, F. J. Casson, E. Fable, A. Ho, F. Koechl, M. Sertoli, et al. Flux-driven integrated modelling of main ion pressure and trace tungsten transport in asdex upgrade. *Nuclear Fusion*, 59(1):016003, 2018.

[57] J. A. Hartigan and M. A. Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the royal statistical society. series c (applied statistics)*, 28(1):100–108, 1979.

[58] L. van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

A Models

A.1 Simulated Data

For the simulated data, a simple model suffices. The hidden structures of the encoder and decoder that were used are the same, with two dense layers with 256 nodes and ReLU activation function, followed by a dropout layer with a 50% drop rate.

For the encoder, this hidden block feeds into two output layers, each with five nodes, as five is the dimensionality of the latent space that was used. One of these layers then represents the mean in this latent space and the second represents the standard deviation[3]. The input to the encoder is a concatenation of the three experimental profiles with the used conditions in the case of a CVAE, or just the experimental profiles in the case of a vanilla VAE.

The decoder takes latent samples as input, and the hidden block feeds to a single dense output layer with 300 nodes, that give predictions for each of the radial points for each of the profiles.

The model was trained for 20,000 epochs, using Adam optimization [24]. The high amount of epochs helped to remove some spikes in the output.

A.2 Experimental Data

The ReD-VAE model for the experimental dataset is more complex, as was described in section 5. This model consists of four different sub-models:

- The conditional priors $p_{\theta_y}(\mathbf{z_y} \mid \mathbf{y})$.
- The approximate posteriors $q_{\phi_x}(\mathbf{z_x} \mid \mathbf{x})$ and $q_{\phi_y}(\mathbf{z_y} \mid \mathbf{x})$.
- The decoder $\hat{\mathbf{x}}_{\theta_x}(\mathbf{z_x}, \mathbf{z_y})$.
- The auxiliary regression network $\hat{\mathbf{y}}_{\omega_y}(\mathbf{z_y})$.

The architectures for all of these sub-models will be discussed separately in the following subsections. The model has a latent dimension $d_{zy} = 3$ for all of the conditional latent spaces, and $d_{zx} = 32$ for the non-conditional latent space. It was trained for 5000 epochs, using Adam optimization [24], with each epoch consisting of two phases, as described in section 5. The full training procedure takes around six hours on the Marconi100 accelerated computing cluster from CINECA with an IBM POWER9 AC922 CPU at 3.1 GHz and an NVIDIA Volta V100 GPU. Evaluation of a trained model can be performed in around thirty seconds locally, on an Intel i7-10750H processor at 2.6 GHz.

[3]The actual output of the standard deviation layers in all VAEs discussed in this thesis is the logarithm of the variance, as this is more numerically stable than directly outputting standard deviation during training.

A.2.1 Conditional Priors

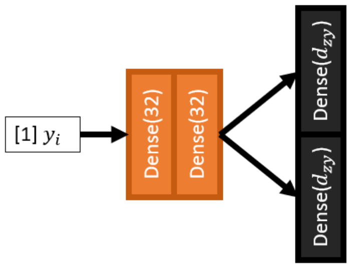


Figure 42 – Network architecture of a single conditional prior network.

A very shallow neural network suffices for the conditional priors. As shown in Figure 42, there are only 2 hidden dense layers in the model, each with 32 nodes and ReLU activation function. These lead to two output layers, one for the latent mean and one for the latent standard deviation. Each of those layers thus has a number of nodes equal to the dimensionality of the $\mathbf{z_y}$ space, d_{zy}, and a linear output function. One such network is made for each condition, and all of these networks work in parallel.

A.2.2 Approximate Posteriors

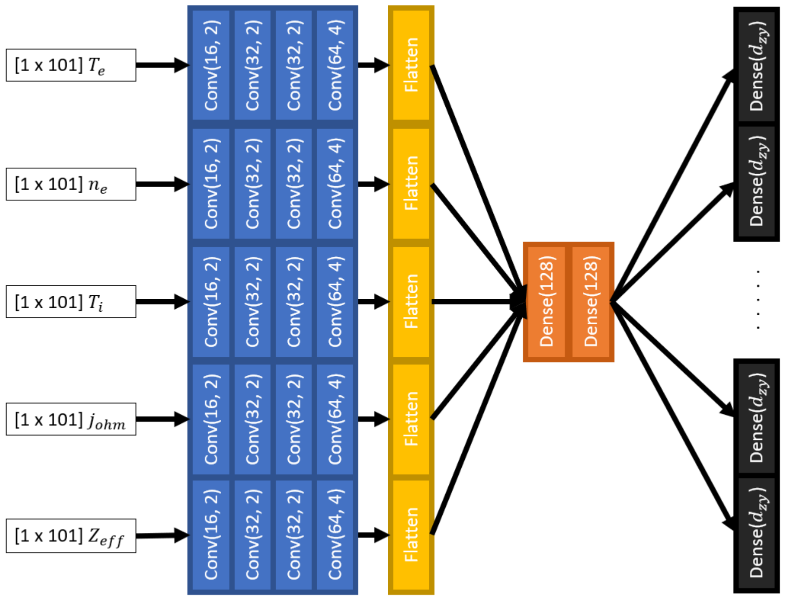


Figure 43 – Architecture for the approximate posterior network $q_{\phi_y}(\mathbf{z_y} \mid \mathbf{x})$.

The approximate posterior networks feature a more complex architecture, as shown in Figure 43. For each of the input quantities, there is a convolutional block consisting of four one-dimensional convolutional layers followed by a flattening layers. The convolutional layers have 16, 32, 32, and 64 filters, with kernel sizes 2, 2, 2, and 4, and with 1, 1, 1, and 4 strides. The convolutional layers have ReLU activation functions. These are concatenated into a dense block consisting of two layers with 128 nodes and ReLU activation. Finally, these feed into two output layers for each of the latent spaces, one for the means and one for the standard deviation.

The encoder for the non-conditional latent space has the same architecture, but with only two output layers for the non-conditional latent space.

A.2.3 Decoder

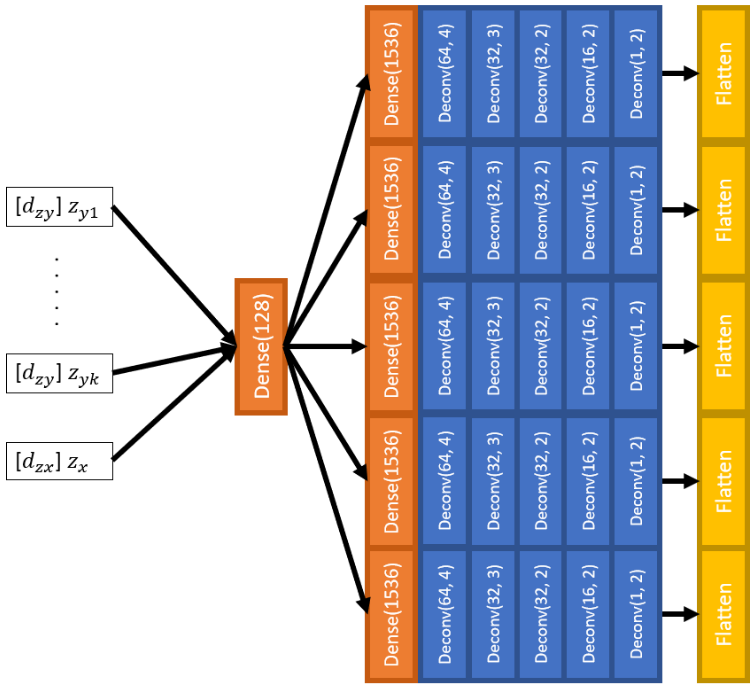


Figure 44 – Architecture for the decoder network.

The decoder architecture is for a large part a mirror image of the encoder architecture, see Figure 44. First, the information from all latent spaces is combined in one dense layer with 128 nodes. This is split into 5 parallel deconvolutional blocks, that are each preceded by a dense layer with 1536 nodes. All dense layers have ReLU activation functions. These layers are followed by five one-dimensional deconvolutional layers, with 64, 32, 32, 16, and 1 filters, and kernel sizes equal to 4, 3, 2, 2, and 1. The first layer has a stride of 4, for the remaining layers this is 1. The last deconvolutional layer has a linear activation function, the preceding four layers have a leaky ReLU activation. Furthermore, these convolutional layers have L2 regularization on kernel, bias, and activation.

A.2.4 Regression Network

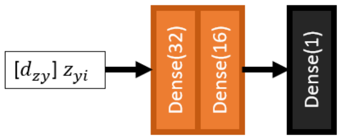


Figure 45 – Architecture for the auxiliary regression networks.

The regression networks again work in parallel, with one network for each of the conditions. Two hidden dense layers with ReLU activation and 32 and 16 nodes feed into one output node, with a linear activation.

A.2.5 Hyperparameter Optimization

In order to find a good model, a delicate balance is required between the different model loss parameters α_1, α_2, β_1, β_2 and γ. Hyperparameter optimization was run in a grid of parameters to find the best model. This optimization was performed in regard to the summed KL-divergence model evaluation metric. For the loss parameters, the following values were used:

$$\begin{aligned}\alpha_1 &\in \{10, 100, 500\}\\ \alpha_2 &\in \{10, 100, 500\}\\ \beta_1 &\in \{0.1\beta_2, 0.5\beta_2, 0.9\beta_2\}\\ \beta_2 &\in \{10, 50, 100\}\\ \gamma &\in \{100, 500\}\end{aligned}$$

The choice to scale β_1 relatively to β_2 is because we always want β_1 to be lower than β_2. This is because β_1 regulates the KL-divergence in the $\mathbf{z_x}$ space. It should always be preferable for the model to encode information into $\mathbf{z_x}$, unless it's beneficial for the regression networks in order to predict $\mathbf{y}$. Therefore, the KL-divergence term for the $\mathbf{z_x}$ space should be lower than the term for $\mathbf{z_y}$. For γ, only high values were considered, as it was earlier shown that high values consistently give better model performance.

This hyperparameter optimization yielded the best performance for the following set of loss parameters: $(\alpha_1, \alpha_2, \beta_1, \beta_2, \gamma) = (100, 100, 10, 100, 500)$. After this, the semi-supervision ratio was optimized, by training a model for the following values:

$$r_{ss} \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$$

This yielded the best performance for $r_{ss} = 0.5$.

For the model that also outputs Z_{eff} and j_{ohm}, a slightly different optimization is required. As five experimental output profiles are generated here, rather than three, the parameters that are related to profile generation in the loss function (i.e. α_1 and γ) are multiplied by a factor 5/3. This resulted in much better performance for the prediction of confinement time.

B Results for an Unbalanced Model

This section presents some alternative results with an unbalanced model, where the relative weight of the KL regularization that is applied in the ReD-VAE loss is 1000 times weaker than in the model that was used in the rest of this work. This leads to better performance for some tasks, but much worse performance in other applications.

Condition predictions with this model are better, as the latent space has more freedom. This makes regression on this space easier. Results are presented Table 8.

Table 8 – Median regression error and median regression error relative to the maximum absolute value for each of the conditions, for a model with very little KL regularization.

Condition	Median Error	Relative to max
B_t	0.162 T	2.97%
I_p	0.112 MA	4.67%
P_{NBI}	0.131 MW	0.495%
P_{ICRH}	0.0139 MW	0.175%
M	0.0143	0.477%
δ_u	0.0213	9.69%
δ_l	0.0151	3.11%
g_r	$1.489 \cdot 10^{21}$	0.650%
g_d	$2.32 \cdot 10^{22}$	2.22%
R_i	8.44 mm	0.340%
Z_i	16.3 mm	0.941%
R_o	25.2 mm	0.855%
Z_o	10.6 mm	0.611%

Generative properties of this model are, however, much worse than the original model, as shown in Table 9.

Table 9 – Frequencies for how many times a sample is within one or two predicted standard deviations from the predicted mean, for a model with very little KL regularization.

Quantity	Within $\mu \pm \sigma$	Within $\mu \pm 2\sigma$
T_e	1.06%	12.8%
n_e	1.64%	16.2%
T_i	1.35%	13.7%
all	0.00%	1.64%

C Derivation of KL-Divergence for Normal Distributions

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

By definition, the KL-divergence is:

$$KL(p(x) \mid\mid q(x)) = \int p(x) \log\left(\frac{p(x)}{q(x)}\right) \,\mathrm{d}x \tag{C.1}$$

And by definition of the normal distribution:

$$p(x) = \frac{1}{\sigma_p\sqrt{2\pi}} e^{-\frac{(x-\mu_p)^2}{2\sigma_p^2}} \tag{C.2}$$

For $q(x)$ a similar form can be used. Substituting these in the logarithm term yields:

$$\begin{aligned}
\log\left(\frac{p(x)}{q(x)}\right) &= \log\left(\frac{\frac{1}{\sigma_p\sqrt{2\pi}} e^{-\frac{(x-\mu_p)^2}{2\sigma_p^2}}}{\frac{1}{\sigma_q\sqrt{2\pi}} e^{-\frac{(x-\mu_q)^2}{2\sigma_q^2}}}\right) && \text{(C.3)}\\
&= \log\left(\frac{\sigma_q}{\sigma_p}\right) + \log\left(e^{-\frac{(x-\mu_p)^2}{2\sigma_p^2}}\right) - \log\left(e^{-\frac{(x-\mu_q)^2}{2\sigma_q^2}}\right) && \text{(C.4)}\\
&= \log\left(\frac{\sigma_q}{\sigma_p}\right) - \frac{(x-\mu_p)^2}{2\sigma_p^2} + \frac{(x-\mu_q)^2}{2\sigma_q^2} && \text{(C.5)}
\end{aligned}$$

Using this we can split up the integral from (C.1) and treat all parts individually.

$$\begin{aligned}
KL(p(x) \mid\mid q(x)) = &\int p(x) \log\left(\frac{\sigma_q}{\sigma_p}\right) \,\mathrm{d}x && \text{(C.6a)}\\
&- \int p(x) \frac{(x-\mu_p)^2}{2\sigma_p^2} \,\mathrm{d}x && \text{(C.6b)}\\
&+ \int p(x) \frac{(x-\mu_q)^2}{2\sigma_q^2} \,\mathrm{d}x && \text{(C.6c)}
\end{aligned}$$

For the first term (C.6a):

$$\int p(x) \log\left(\frac{\sigma_q}{\sigma_p}\right) \,\mathrm{d}x = \log\left(\frac{\sigma_q}{\sigma_p}\right) \int p(x) \,\mathrm{d}x = \log\left(\frac{\sigma_q}{\sigma_p}\right) \tag{C.7}$$

Since for the integration of a probability density function we have:

$$\int p(x)dx = 1 \tag{C.8}$$

By definition of the variance:

$$\sigma_p^2 = \int p(x)(x-\mu_p)^2 \, \mathrm{d}x \tag{C.9}$$

we have for the term in (C.6b):

$$-\int p(x)\frac{(x-\mu_p)^2}{2\sigma_p^2} \, \mathrm{d}x = -\frac{1}{2\sigma_p^2}\int p(x)(x-\mu_p)^2 \, \mathrm{d}x = -\frac{\sigma_p^2}{2\sigma_p^2} = -\frac{1}{2} \tag{C.10}$$

The final term in (C.6c) requires some careful rewriting:

$$\begin{aligned}
\int p(x)\frac{(x-\mu_q)^2}{2\sigma_q^2} \, \mathrm{d}x &= \int p(x)\frac{((x-\mu_p)+(\mu_p-\mu_q))^2}{2\sigma_q^2} \, \mathrm{d}x && \text{(C.11)}\\
&= \int p(x)\frac{(x-\mu_p)^2 - 2(x-\mu_p)(\mu_p-\mu_q) + (\mu_p-\mu_q)^2}{2\sigma_q^2} \, \mathrm{d}x && \text{(C.12)}\\
&= \frac{1}{2\sigma_q^2}\int p(x)(x-\mu_p)^2 \, \mathrm{d}x \\
&\quad - \frac{2(\mu_p-\mu_q)}{2\sigma_q^2}\int p(x)(x-\mu_p) \, \mathrm{d}x \\
&\quad + \frac{(\mu_p-\mu_q)^2}{2\sigma_q^2}\int p(x) \, \mathrm{d}x && \text{(C.13)}\\
&= \frac{\sigma_p^2}{2\sigma_q^2} + \frac{(\mu_p-\mu_q)^2}{2\sigma_q^2} = \frac{(\mu_p-\mu_q)^2+\sigma_p^2}{2\sigma_q^2} && \text{(C.14)}
\end{aligned}$$

Where, in order to obtain (C.14), (C.8) and (C.9) were used, as well as the definition of the mean in:

$$\int p(x)(x-\mu_p) \, \mathrm{d}x = \int p(x)x \, \mathrm{d}x - \mu_p\int p(x) \, \mathrm{d}x = \mu_p - \mu_p = 0 \tag{C.15}$$

By substituting the results from (C.7), (C.10) and (C.14) into (C.6) we obtain the final result for the KL-divergence:

$$KL(p(x) \mid\mid q(x)) = \frac{(\mu_p-\mu_q)^2+\sigma_p^2}{2\sigma_q^2} - \frac{1}{2} + \log\left(\frac{\sigma_q}{\sigma_p}\right) \tag{C.16}$$