Applying Machine Learning Techniques to Simulated and Experimental Data from the Wendelstein 7-X

Nathan Belmore

Dec 10th, 2022
Version: Revision 3

University of Greifswald

Department of Physics
with support from the Max Planck Institue for Plasma Physics

Master's in Physics

Applying Machine Learning Techniques to Simulated and Experimental Data from the Wendelstein 7-X

Nathan Belmore

1. Reviewer Prof. Thomas Sunn Pedersen
Department of Physics
University of Greifswald

2. Reviewer Prof. Ralf Schneider
Department of Physics
University of Greifswald

Supervisors Daniel Böeckenhoff

Dec 10th, 2022

Nathan Belmore

Applying Machine Learning Techniques to Simulated and Experimental Data from the Wendelstein 7-X

Master's in Physics, Dec 10th, 2022

Reviewers: Prof. Thomas Sunn Pedersen and Prof. Ralf Schneider

Supervisors: Daniel Böeckenhoff

University of Greifswald

Department of Physics

with support from the Max Planck Institue for Plasma Physics

Street address

Postal Code and City

Abstract

The Wendelstein 7-X (W7-X) plasma experiment is the most advanced stellarator of the HELIAS type. W7-X aims to demonstrate the feasibility of steady-state operation of a plasma experiment and the potential viability of a fusion reactor. W7-X is a five-fold device with a unique magnetic field geometry created using non-planar and planar superconducting coils. It also uses an island-divertor concept for managing heat and particle exhaust at the plasma-wall interfaces. The graphite divertor targets used in W7-X are designed to withstand a heat load of up to 10 MW/m^2, but exceeding this limit can damage the divertor structures and prevent sustained operation of the device. In order to prevent thermal overload, the magnetic field can be adjusted using trim and control coils. This thesis presents an approach for inferring the edge rotational transform, a key parameter that determines the position of the magnetic islands and heat load pattern, from infrared camera data. Using an inceptnet convolutional neural network, the 520x130 input resolution is a good compromise between computational cost and network performance. When evaluating ι, the rotational transform, this network is able to achieve an $rmse$ of $4.13 \cdot 10^{-3}$ and a training time of just less than a day on a single GPU, which is an order of magnitude better than prior work with similar data.

Contents

1 Introduction

1.1 Introduction

Wendelstein 7-X (W7-X) is the most advanced stellarator of the HELIAS type. Its aim is to demonstrate the feasibility of steady-state operation of a plasma experiment and fusion reactor. It is a five-fold device with a quasi-isodynamic magnetic field created by 50 non-planar and 20 planar superconducting coils. Like its predecessor, Wendelstein 7-AS, it uses the island-divertor concept for heat and particle exhaust. Due to the five-fold symmetry, 10 identical divertor units are arranged so that the divertor targets intersect the magnetic islands present in the edge magnetic field of W7-X and serve as a heat resistent plasma-wall interface. The graphite divertor targets are designed to withstand a heat load of up to $10\frac{MW}{m^2}$. Exceeding the heat load limit could result in severe damage to the divertor structure and prevent sustained operation of the device.

The dynamics of the particles in the plasma is determined by the magnetic field geometry. The quasi-isodynamic magnetic field geometry in W7-X is characterized by an edge rotational transform $\bar{\iota}$, that is the ratio of polodial turns per toroidal turn of a magnetic field line, which is close to 1 and results in five edge magnetic islands in the so-called standard magnetic field configuration. As the magnetic islands are used to create a plasma-wall interface in the island divertor, the heat and particle deposition pattern strongly depends on the position of the magnetic islands and therefore on the edge rotational transform.

In order to avoid thermal overload of individual divertor targets, the magnetic field can be corrected using a set of trim and control magnetic coils. Real time control of the head load deposition pattern could be used to prevent the thermal overload. In order to provide real-time control control of the state of the plasmas variety of different plasma parameters would need to be provided in real-time, such as $\bar{\iota}$. However, $\bar{\iota}$ cannot be measured directly and simulating $\bar{\iota}$ is computationally expensive and cannot be done in real time. Inferring $\bar{\iota}$ from the heat load pattern is valuable to any such control scheme. In this thesis an approach to inferring $\bar{\iota}$ from the heat load pattern provided by the infrared camera is presented.

Traditionally, to provide an estimate for $\bar{\iota}$ that accounts for the plasmas contribution to the magnetic field, one would need to simulate using a framework like VMEC. The current computational costs makes real time simulation impossible and therefore traditional approaches are not suitable for real time control. In this thesis a convolutional neural network (CNN) is instead used to infer $\bar{\iota}$ from the heat load pattern. Unlike simulations, trained neural nets can be run in real-time. The CNN is trained on a dataset of simulated $\bar{\iota}$ based on real heat load patterns. The CNN is then used to infer $\bar{\iota}$ from the heat load pattern provided by the infrared camera.

One of the questions being addressed by this thesis will be the impact of scaling the input image resolution has on the accuracy of $\bar{\iota}$. To resolve this question, the network is repeatedly trained on a dataset with different input image resolutions and the error in the regression on $\bar{\iota}$ is measured. The distributions of the errors are then compared to help validate the results.

2 Principles of magnetic confinement

2.1 Non-axisymmetric magnetic fields

Understanding why it is possible to extract properties of the plasma from a heat load image requires an understand the underlying physics. In a fusion device or plasma experiment, the motion of the particles and thus the deposition of particle and heat load on the plasma facing components is inextricably linked to the magnetic field topology. Therefore, a short overview over magnetic confinement based on [13] will be given in this section.
For confinement of a plasma, a magnetic field is required, which is related to the plasma current **J** by Ampere's law

$$\nabla \times \mathbf{B} = \mu_0 \mathbf{J}. \tag{2.1}$$

The magnetic force created by the plasma current balances the pressure force of the plasma and enables confinement. The amount of plasma current needed for confinement can be derived from the MHD equation of motion which results in

$$\mathbf{J} \times \mathbf{B} = \nabla p \tag{2.2}$$

for the steady state case without flows. As a consequence, **J** and **B** lie in surfaces of constant pressure, i.e.

$$\mathbf{B} \cdot \nabla p = \mathbf{J} \cdot \nabla p = 0. \tag{2.3}$$

These surfaces have toroidal topology in a magnetically confined plasma. Magnetic coordinates are used to describe these surfaces of constant pressure in a coordinate system where one coordinate is constant on these surfaces and the magnetic field lines are straight lines. Introducing the poloidal and toroidal angles ϑ and φ, a representation of the magnetic field as a composition of the toroidal and poloidal component is given by

$$\mathbf{B} = \nabla\psi \times \nabla\theta + \nabla\varphi \times \nabla\chi \tag{2.4}$$

with $\theta = \vartheta + \lambda$.
According to 2.3, ψ and χ are constant on surfaces of constant pressure and can be chosen to vanish on the innermost surface of constant pressure, that is just a line and described as the magnetic axis. By calculating the surface integral, it can be easily shown that the magnetic flux through a poloidal cross section of constant φ between the magnetic axis and a surface of constant ψ is equal to $2\pi\psi$, while the poloidal magnetic flux through a surface of constant ϕ between the magnetic axis and a flux surface ψ is equal to $2\pi\chi$. These surfaces of constant toroidal and poloidal flux are called flux surfaces.
χ can be interpreted as the derivative of ψ, which is called the rotational transform

$$\iota = \frac{d\chi}{d\psi} \tag{2.5}$$

and indicates the number of poloidal turns per toroidal turn of a field line, because θ and φ vary in proportion along a field line:

$$\frac{d\phi}{d\varphi} = \frac{\mathbf{B}\cdot\nabla\phi}{\mathbf{B}\cdot\nabla\varphi} = \frac{(\nabla\varphi\times\nabla\chi)\cdot\nabla\phi}{(\nabla\psi\times\nabla\theta)\cdot\nabla\varphi} = \iota \tag{2.6}$$

Introducing $\alpha = \theta - \iota\varphi$ leads to the so called Clebsch representation of the magnetic field

$$\mathbf{B} = \nabla\psi\times\nabla\alpha, \tag{2.7}$$

where $\mathbf{B}\cdot\nabla\alpha = 0$ and α is constant along the magnetic field. Consequently, the magnetic field lines are straight in the (θ,φ)-plane, which was one of the requirements for the magnetic coordinates. Thus, the magnetic field lines can be described by the two coordinates ψ and α. Because $\frac{d\theta}{d\varphi} = \iota$ along a field line , the poloidal angle of a field line changes by $2\pi\iota$ after one toroidal turn. If ι is a rational number, i.e. $\iota = \frac{n}{m}$, the field line returns to where it started, while it traces out the whole flux surface for irrational ι.

Although the magnetic field in stellarators is mainly generated by the magnetic field coils, the plasma current **J** that is needed to balance the plasma pressure modifies the magnetic field and consists of parallel and perpendicular components

$$\mathbf{J} = \mathbf{J}_{\parallel} + \mathbf{J}_{\perp}. \tag{2.8}$$

The perpendicular component is required to produce the magnetic force given in 2.2 and the parallel component is needed to satisfy 2.3. Thus, the current is given by

$$\mathbf{J} = (u(\psi,\theta,\varphi)p'(\psi) + \frac{\langle \mathbf{J}_\parallel \mathbf{B}\rangle}{\langle \mathbf{B}^2\rangle})\mathbf{B} + \frac{\mathbf{B}\times\nabla p}{B^2}, \tag{2.9}$$

where $u(\psi,\theta,\varphi)$ satisfies the magnetic differential equation $\mathbf{B}\cdot\nabla u = -(\mathbf{B}\times\nabla\psi)\cdot\nabla(\frac{1}{B^2})$. The first and second term in 2.9 describe the parallel current, where the first one is the so-called Pfirsch-Schlüter current and the second term the Ohmic current. The third term describes the perpendicular diamagnetic current. A more detailed derivation of the plasma currents is given in [8].
While the existence of MHD equilibria [22] can be proven e.g. using the variational principle [13], the pressure profiles or field lines do not necessarily have to be continuous or the plasma current free from singularities. A Fourier expansion of the parallel current in equation 2.9 reveals the possible singularities in the current on the rational surfaces, namely a surface current and a divergent Pfirsch-Schlüter current. While a detailed derivation of these singularities can be found in [13], we do not focus on the discussion of singularities in the current here as they can be avoided by giving up on the requirement of nested flux surfaces and allowing for magnetic islands at rational surfaces.
The representation used for the magnetic field here,

$$\mathbf{B} = \nabla\times\mathbf{A} = \nabla\psi\times\nabla\theta + \nabla\varphi\times\nabla\chi, \tag{2.10}$$

is a general repsresentation of the magnetic field and does not necessarily require the existance of magnetic surfaces. They exist if χ can be written as a representation of ψ, since then $\mathbf{B}\cdot\nabla\psi = 0$, but are not generally required. According to equation 2.10, the field lines are Hamiltonian

$$\frac{d\psi}{d\varphi} = \frac{\mathbf{B}\cdot\nabla\psi}{\mathbf{B}\cdot\nabla\varphi} = -\frac{\partial\chi}{\partial\theta} \tag{2.11}$$

$$\frac{d\theta}{d\varphi} = \frac{\mathbf{B}\cdot\nabla\theta}{\mathbf{B}\cdot\nabla\varphi} = \frac{\partial\chi}{\partial\psi} \tag{2.12}$$

and are generally chaotic. The Hamiltonian χ can be Fourier decomposed, leading to

$$\chi(\psi,\theta,\varphi) = \chi_0(\psi) + \sum_{m,n\neq 0}\chi_{m,n}(\psi)e^{i(m\theta-n\varphi)} := \chi_0(\psi) + f(\psi,\theta,\varphi). \tag{2.13}$$

If ψ and $\alpha = \theta - \iota(\psi_0)\varphi$ are used as canonical coordinates, , the Hamiltonian can be replaced by

$$H(\psi, \alpha, \varphi) = \chi(\psi, \theta, \varphi) - \iota(\psi_0)(\psi - \psi_0) = \chi_0(\psi) + f(\psi, \alpha) - \iota(\psi_0)(\psi - \psi_0), \quad (2.14)$$

which is equal to

$$H(\psi, \alpha, \varphi) = \frac{\chi_0''(\psi_0)(\psi - \psi_0)^2}{2} + f(\psi_0, \alpha). \quad (2.15)$$

This Hamiltonian can be integrated and described the magnetic islands around the resonant surface. The shape of the islands depends on the potential $f(\psi_0, \alpha)$. if only the first term is kept in the Fourier expansion, the potential becomes sinusoidal and the system is equivalent to an ordinary pendulum with the island separatrix corresponding to $2\chi_{mn}$ and

$$\psi - \psi_0 = \sqrt{\frac{4\chi_{mn}(1 - cos(m\alpha))}{\iota'(\psi_0)}}. \quad (2.16)$$

The width of the islands is

$$\Delta\psi = \sqrt{\frac{32\chi_{mn}}{\iota'(\psi_0)}}. \quad (2.17)$$

As magnetic fields without continous symmetry are in general chaotic, they do not have nested flux surfaces everywhere and chains of islands can form, which consist if nested flux surfaces on their own with their own rotational transform.
The existence and calculation of such nested flux surfaces is carried out by the VMEC code based on the variational principle [15],[14].

2.2 Magnetic field in Wendelstein 7-X

In the stellarator experiment Wendelstein 7-X [1],[3], the magnetic field has been optimized for good MHD-stability and good neoclassical confinement [12]. It is the first device of the HELIAS line [24], in which the plasma currents, i.e. the bootstrap current, the Pfirsch-Schlüter current and the diamagnetic current and their effects on the magnetic field are minimized [10].
The main magnetic field in Wendelstein 7-X is generated by 50 non-planar and 20 planar superconducting magnetic field coils that are cooled with liquid helium [30]. According to the five-fold symmetry of Wendelstein 7-X, there exist five different types of non-planar and two types of planar magnetic field coils made of NbTi, which are installed in each of the ten half modules of the stellarator. The flip-symmetric

installation of two half modules results in the five identical modules composing Wendelstein 7-X.
The magnetic field generated by the coils and plasma currents determines the heat and particle load onto the plasma facing components, which are controlled by a chain of magnetic islands at the edge of the plasma. The number of islands depends on the edge rotational transform and is shown in figure 2.1 for the standard configuration, where an edge ι of 5/5 creates five islands.

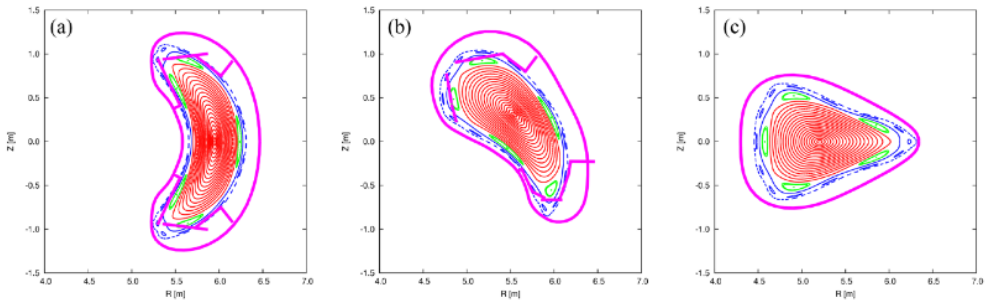

Fig. 2.1: Poincaré plots of the vacuum magnetic field in standard configuration for three different poloidal cross sections from [32]. The islands intersecting the plasma facing components are shown in green, the target plates in pink.

The rotational transform can be varied between edge $\iota = 5/6$ in the so-called low-iota configuration and edge $\iota = 4/5$ in the high-iota configuration, with six and four magnetic islands respectively [19]. Following the island divertor concept [2], [29], target plates made of graphite intersect the islands to divert the heat and particle load and are shown in figure 2.1.

2.3 Heat load on the plasma facing components

The heat load on the plasma facing components that intersect the islands is limited by the material properties of the plasma facing components and therefore needs to be monitored closely. While the divertor targets are designed for a heat flux of up to $10\,\mathrm{MW/m^2}$, significantly lower heat loads of $0.5\,\mathrm{MW/m^2}$ can be tolerated on the surrounding baffle structures [18]. Therefore, the surface temperature of the divertor targets and the surrounding structures are monitored closely by a set of infrared diagnostics to avoid overloading of the components and to study the particle and heat load deposition pattern on the plasma facing components. Based on the tenperature of the components, the heat flux can be derived using the two-dimensional thermal model THEODOR [31]. During the last experimental campaign,

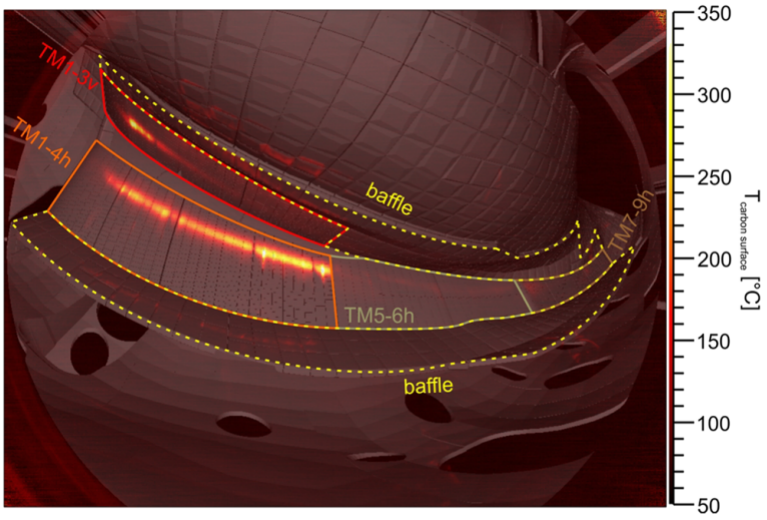

Fig. 2.2: Infrared image of one divertor module and the surrounding plasma facing components overlaid with a CAD model, from [18]

nine out of the ten divertor modules in Wendelstein 7-X - a lower and an upper divertor unit in each of the five modules of the torus - have been monitored by a set of different infrared and visible cameras. Infrared microbolometric cameras, which have been specifically designed to work in a magnetic field of up to 3 T use a fish eye lens to provide a wide angle view of the whole divertor units. Detailed specifications of the cameras in use are given in [18]. The cameras provide a spectral response in the range of 8-10 μm and can therefore be used to measure surface temperatures between 30 and 5000°C. An example for the temperature distribution measured in one divertor unit is given in figure 2.2, where the temperature distribution is overlaid by a CAD-model of the plasma facing components. This representation allows for a detailed sudy of the heat load deposition pattern on the plasma facing components [23] as well as the localization of individual thermal hotspots depending on the magnetic field configuration and the location of the magnetic islands.

Dropping here for now.

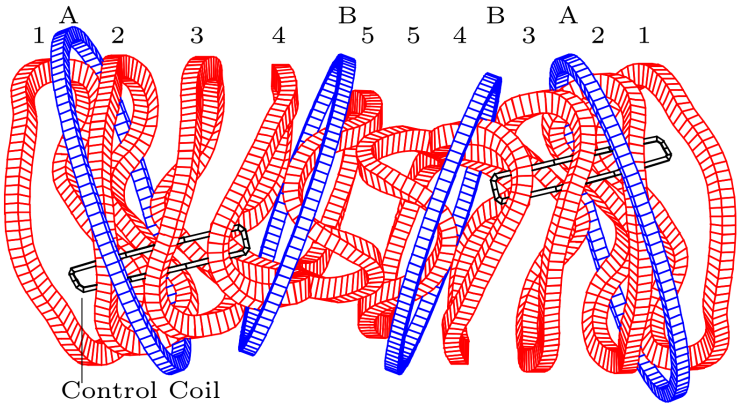

Fig. 2.3: Coil diagram from one of the five W7X modules. The red coils are the non-planar coils. The blue coils are the two planar coils, A and B. Taken from [6]

3 Related Work

3.1 Prior related work:

The topic of this thesis was built upon the work of many of colleagues at IPP. The work done by D. Böckenhoff et al. in their 2018 paper [6] laid down the fundamental research that motivated this thesis. They set out to use artificial neural networks to reconstruct I_B, the coil current from the B-coils (seen in the following diagram 2.3), the from heat load in the experiment. Since they used the ι-scan data from the prior OP1.1 experimental campaign, which used a limiter, their work is applied to a very different data set. The limiter that was used in W7X for the OP1.1 campaign is a piece of graphite designed to interface with the plasma to prevent it from interacting with the wall, potentially damaging the wall surface. A divertor plays a similar role to a limiter but in addition to acting as a plasma interface to protect the wall there is a subdivertor space that helps compress neutral gas which helps with the pumping process. Because of this additional function the form of the divertor used in the OP1.2 data is quite different (see figure 3.1). As an additional note: the OP2 experiment will include a water cooled divertor, which can impact the heat load recorded by the thermal cameras.

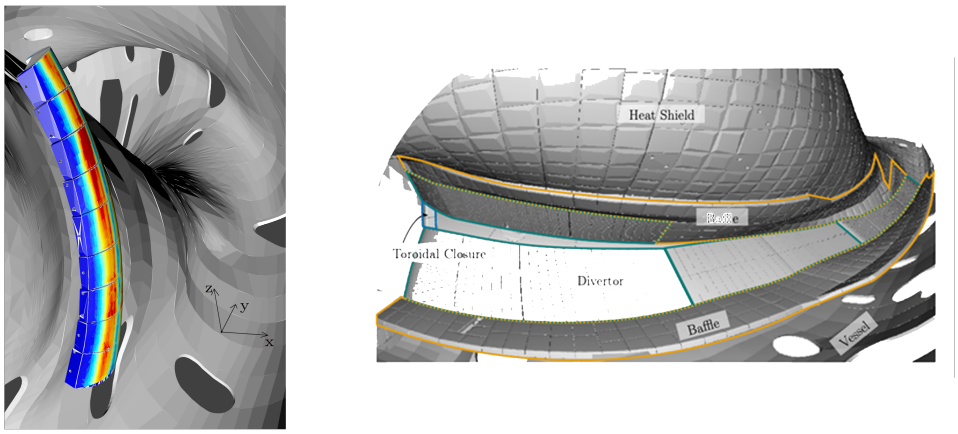

Fig. 3.1: Left: Three-dimensional representation of heat flux on the surface of a limiter in W7X. Taken from [6] Right: Labeled image of the OP1.2 divertor and plasma facing components

In the paper from D. Böckenhoff et al., data was limited to 6 values of I_B taken from actual experimental data. While they were able to reconstruct I_B using only the real world data with an $rmse$ (root mean squared error) of 0.047 for the test set results when using interpolated and extrapolated data. $rmse$ is defined as:

$$rmse = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{3.1}$$

However, the network did not perform as well when applied to extrapolated data. To improve the performance, the authors used synthetic data generated from simulated ι-scan data to train a new version of the network. The validation set was also entirely synthetic. While the new network performed well, there was an observed systematic underestimation of I_B by -0.026 ± 0.001.

The best results came from using the neural network pretrained on synthetic data and trained on the real data set after. The combination yielded the best results, with an $rmse$ of 0.013 and better performance on extrapolated results overall.

In the 2018 paper "Neural network performance enhancement for limited nuclear fusion experiment observations supported by simulations" from M. Blatzheim et al. [4], the authors refined the methods used in the 2018 paper to improve the neural network performance. This paper also works with the data from the OP1.1 and worked with a mix of measured and simulated inputs and outputs.

In this paper they compared several neural network architectures, a fully connected feed-forward neural network with 3 layers of 64 notes each, a neural network composed of a series four 5x5, eight 3x3, and 16 more 3x3 convolutional filters followed by two fully connected layers, and a shallow network composed of an inception module, followed by pooling, a 1x1 convolutional layer, and two fully connected layers.

Different partitions of the input image were used. The input resolutions were $9x5$, $18x8$, $72x15$, $144x30$. Also, three different input parameterization of the partitions were tested: (μ, σ), (μ, δ), and ρ. The μ and σ parameterization is the standard deviation of the heat load, μ is the mean of the heat load, and δ is the difference between the maximum and minimum heat load values. The ρ parameterization is the relative heat load. Increasing the number of inputs also resulted in a higher number of network parameters to accommodate the increased input size.

Solid results were found with the $9x5$ and $36x12$ resolutions using a convolutional NN and mixed real world and simulated data using the ρ parameterization, achieving

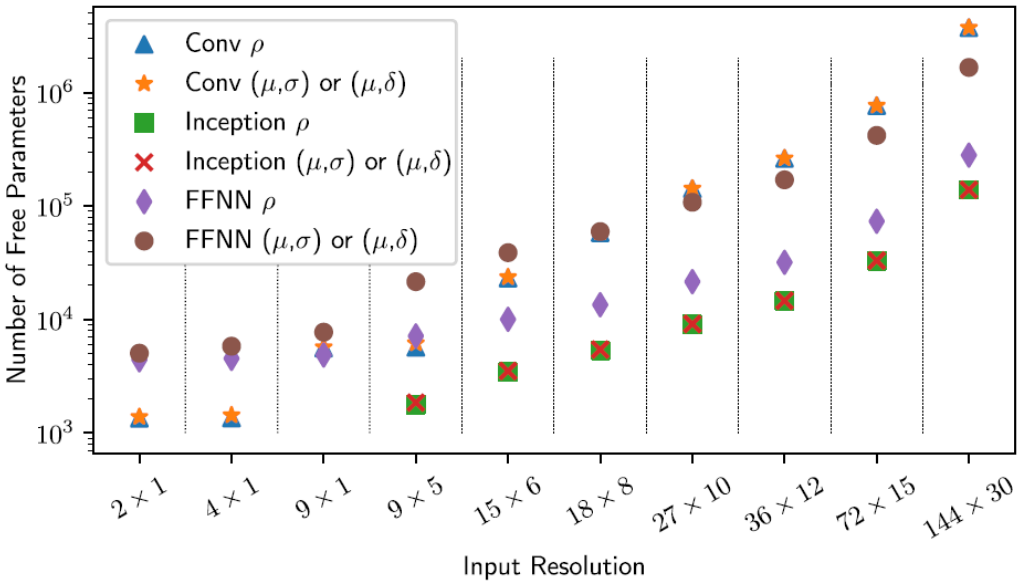

Fig. 3.2: The free parameters of the NN based on the input resolution, parameterization, and architecture.

an $rmse$ of 0.008. The inception model showed improved performance for higher resolution inputs and could be useful for data like that from the OP1.2 campaign.

The results from M. Blatzheim et al. were used as a baseline for the final paper in the series, "Neural Network Regression Approaches to Reconstruct Properties of Magnetic Configuration from Wendelstein 7-X Modeled Heat Load Patterns" [5]. The authors used the results from the previous papers to improve the neural network performance but unlike the prior papers which were using data from the OP 1.1 campaign which used a limiter, this paper looks at heat load data from the divertor that was added for the OP 1.2 campaign. Since this thesis is also using data from the OP 1.2 campaign, the results from this paper helped to inform the methods used in this thesis.

In this paper the authors used two targets for training.

One of the issues with the dataset used in the 2019 paper is that the reconstruction problem is more complex compared to the limiter configurations used in the 2018 papers. Despite the complexity, the papers authors produced several neural architectures that performed well both in $rmes$ and with a 5-fold cross validation. Good results were achieved with a convolutional neural network but the network was shallow compared to some of the deeper networks which tested better.

4 Model Design

4.1 Neural Networks

Since confined plasmas are highly complex systems with many degrees of freedom, it is difficult to predict the behavior of the plasma even with the most advanced simulation tools. Even direct measurements of the plasma state can vary significantly from method to method. [citation and examples might be helpful here].

Machine learning (ML) models are algorithms that use data and statistical operations to optimize the parameters that compose the model. Unlike simulation, ML models are trained on data and can be used to predict the behavior of complex systems in real time.

There are two broad catagories of ML models, supervised and unsupervised. Supervised models are ones which used labeled data, or data for which the expected output of the model is known in advance. Unsupervised models are models for which a pattern is learned from the data directly. The model use for this project is a supervised ML model because we know what the expected inputs and outputs should be and can produce labels for both.

One such ML model is a neural network, which is a mathematical model that is composed of a large number of interconnected processing elements, called neurons, which are organized into layers. The input layer receives data, in this case an image, and the output layer produces the final output of the network, which would be a prediction of the plasma state. The neurons in the hidden layers process the data and produce an output.

A nerual network can learn to recognize patterns in data through a training process, optimizing the weights after each step to find the best combination for a given task. During the training process, the weights $w_{i,j}$, and biases b_i, where i is the index of the layer and j is the index of the weight, are adjusted to minimize the error between the predicted output and the true output, thus improving the accuracy of the network.

A non-linear function called the activation function, which is typically a sigmoid [see eq.4.1] or a rectified linear unit (ReLU) function [see eq.4.2]. The sigmoid function is a non-linear activation function that maps any real-valued number to a value between 0 and 1. It is commonly used in neural networks to convert the output of a linear function to a value that can be interpreted as a probability.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{4.1}$$

$$f(x) = \max(0, x) \tag{4.2}$$

The output of the activation function is then passed through another set of weights and summed to produce the final output of the network. Because the activation function is non-linear, the network can learn to recognize complex patterns in the data. The process of passing the output of one layer to the next is called forward propagation. Below is an example of a neural network with 3 layers, an input layer, a hidden layer, and an output layer.

$$\begin{aligned}
z_1 &= w_{1,1}x_1 + w_{1,2}x_2 + ... + w_{1,n}x_n + b_1 \\
a_1 &= \sigma(z_1) \\
z_2 &= w_{2,1}a_1 + w_{2,2}a_2 + ... + w_{2,m}a_m + b_2 \\
a_2 &= \sigma(z_2) \\
&\vdots \\
z_k &= w_{k,1}a_{k-1} + w_{k,2}a_{k-1} + ... + w_{k,m}a_{k-1} + b_k \\
a_k &= \sigma(z_k)
\end{aligned}$$

These weights and biases are adjusted during the training process to improve the accuracy of the network. Backpropagation [see eq. 4.3] is a commonly used algorithm for training a neural network. It is based on the idea of gradient descent, which is a mathematical optimization technique for finding the minimum of a function. In the context of a neural network, the function we are trying to minimize is the error between the predicted output and the true output.

Backpropagation works by calculating the gradient of the error function with respect to each of the weights in the network, and then adjusting the weights in the opposite

direction of the gradient. This process is repeated for a number of iterations, with the weights being adjusted each time to reduce the error.

$$\Delta w_j = \eta \frac{\partial E}{\partial w_j} = \eta \sum_k \frac{\partial E}{\partial o_k} \frac{\partial o_k}{\partial net_k} \frac{\partial net_k}{\partial w_j} \tag{4.3}$$

Here, Δw_j is the change in weight w_j, η is the learning rate, E is the error function, o_k is the output of neuron k, and net_k is the net input to neuron k. The learning rate is a hyperparameter that controls how much the weights are adjusted at each step. If the learning rate is too large, the weights may overshoot the minimum and never converge. If the learning rate is too small, the weights will not be adjusted each training step enough to converge quickly. The error function is a function that measures the difference between the predicted output and the true output. The error function is typically the root mean sum of the squared differences ($rmse$) [see eq. 4.4] between the predicted output and the true output.

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2} \tag{4.4}$$

Here, n is the number of samples in the dataset, y_i is the true value of the i-th sample, and $\hat{y}_i$ is the predicted value of the i-th sample. The RMSE is calculated by taking the square root of the MSE, which is the average of the squared differences between the predicted values and the true values.

The error function is minimized by adjusting the weights in the opposite direction of the gradient of the error function with respect to the weights. This process is repeated for each weight in the network, which represents one training step. Training is repeated for a number of iterations, with the weights being adjusted each time to reduce the error. G. Cybenko proved that a neural network with a single hidden layer can approximate any continuous function to any desired accuracy, given enough neurons in the hidden layer. This is why neural networks are so powerful, and why they are used in a wide variety of applications.

4.2 Convolutional Neural Networks

The primary input for the model will be thermal images taken from the infrared cameras at W-7X. Since the thermal camera data is sparse, with most of the pixels

for any given shot containing zeros or noise, a convolutional neural network with that is optimized for sparse data would be ideal.

Convolutional neural networks (CNNs) are a type of neural network that is specifically designed to process data with a grid-like structure, such as an image. They are composed of a number of interconnected layers, including convolutional layers, pooling layers, and fully connected layers.

The convolutional layer is the core building block of a CNN. It applies a series of variable-dimensional convolutional filters to the input data. The convolution operation is used to apply a filter to the input data, producing a feature map 4.1), which is a representation of the input data that has been transformed by the filter.

These convolutions are stacked, being applied to the same input to produce multiple outputs which are concatenated along the z-axis. Typically, strides (the number of pixels skipped over when translating the kernel) and padding (adding pixels around the edge of the image) allow convolutions to change the output x and y dimensions. So as more layers of convolutions are successively applied, the outputs of each layer grow in z-height, and shrink in x and y. The final output of the convolutional network is then passed to a fully connected network.

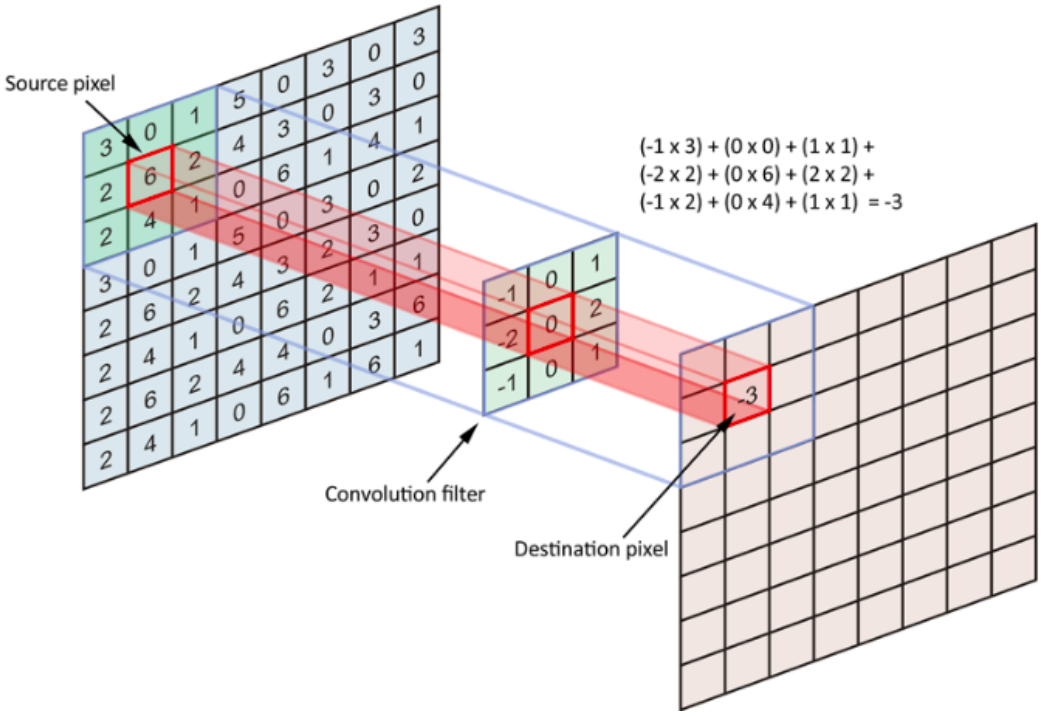

Fig. 4.1: Example of a single 3x3 convolution operation. The source image is convolved with a 3x3 kernel. The kernel's weights are determined using backpropagation. [cite source]

The inceptnet architecture [33] has been demonstrated by D. Böckenhoff et al. [7] to be effective on a simplified version of this problem and by M. Blatzheim et al. [5] to be effective on data from the same divertor configuration. In initial tests it was very promising for this application and met an important boundary conditions, mainly the network fit into the available video memory on the system used to train the network. Inceptnet uses a series of sub-networks, known as inception modules (see fig. 4.2). Each inception module is made up of 3 convolutional paths (1x1, 3x3, and 5x5) and a pooling path. To assure the outputs from each path have the same resolution, padding is added to the 3x3 and 5x5 convolutions. The number of each convolutional filter can be adjusted at each step and the outputs are concatenated together. The idea behind having multiple convolution sizes is that some features of the input might be lost if only size of kernel is applied.

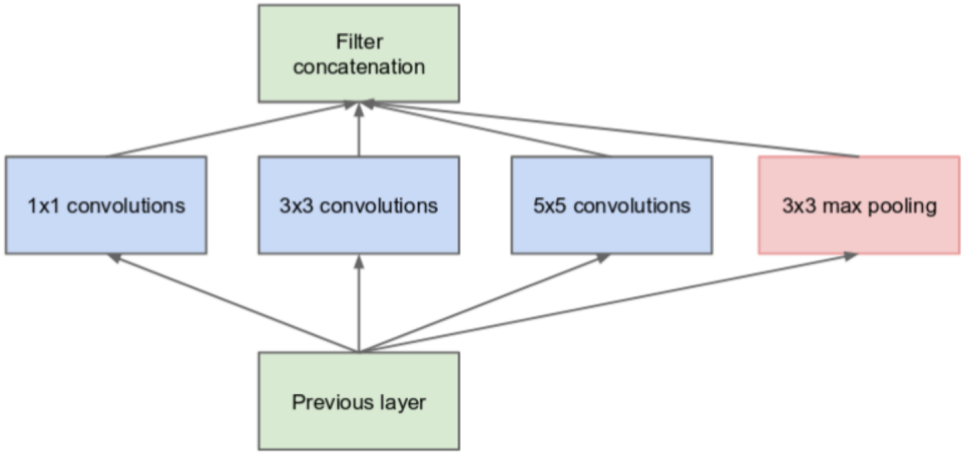

Fig. 4.2: A diagram of the inception module. The inputs from the prior layer are fed into 4 paths, applying 1x1, 3x3, 5x5, convolutions with the final layer being 3x3 max pooling. The outputs of each layer are concatenated and passed to the next incept module. The number of 1x1, 3x3, and 5x5 convolutions per incept module can be adjusted.

Each of the paths in the four branches of the inception module play a different role. The 1x1 filter doesn't take into account any patterns in the inputs height and width, but it does work across channels (depth) of the image. This acts to reduce the dimension of the input while connecting the values from the other channels of the input. The 1x1 convolution can also be used to reduce the number of channels as the output of the filter will be 1x1xn, where n is the number of 1x1 filters being applied [20]. This is why each branch has a 1x1 convolution. The 1x1 convolutions marked in yellow in fig. 4.2 are designed to reduce the number of channels input to the higher computational cost which dramatically reduces the number of weights needed for a module.

The 3x3 and 5x5 convolutions look at spacial patterns across the width, height, and depth of the input. In this way they are actually a 3-dimensional convolution, but most convolutions are referred to by their 2-dimensional cross-section. These two filters are typical optimized for finding features of the input at different scales. Prior to this multi-branch approach, machine learning scientists would hand select the ordering of the various convolution sizes to be applied in succession. By providing multiple paths and allowing the training process to determine which branches are necessary for a given step takes the human bias out of the training process in this regard.

There are several other structures that appear in the inception module. One being a 3x3 max pooling layer, which is like a 3x3 convolution, but instead of taking the element-wise product with the filter, it returns the max pixel value over the region of the feature map that overlaps with the filter. Pooling layers are a method of downsampling the input while attempting to preserve the most relevant pixels and are a staple of most deep convolutional networks. The concatenation layer just concatenates the outputs from the various branches together. Each branch has padding so that the height and width channels are aligned to make concatenation simple.

Lastly, there are batch normalization layers after every operation on each brach of the inception module. Batch normalization is powerful tool in any deep learning model. It has been shown that batch normalization solves the issue of internal covariate shift, where layers in the network shift the input distribution to the next layer [17]. Since the inputs of any layer in the network are affected by the weights of every prior layer, small changes in the distribution get amplified as they propagate throughout the network. To address this during training batches of data are normalized, transforming the neurons output using the first and second statistical moments (mean and standard deviation) across the batch. Two additional trainable parameters, γ and β are applied after normalization to specify the output distribution, so each layer can achieve a unique output distribution based on the optimal one for the next operation or layer.

Incept modules also have two major advantages over classic deep convolutional networks, they require far fewer parameters, which reduced the overall memory footprint of the model and far fewer floating point operations than prior convolutional modules. For example, using a 5x5 convolution to produce the same number of output channels as a incept module requires over 7 times the number of model parameters and floating point operations compared to the incept module. However, despite having significantly fewer model parameters, when the inceptnet was release,

it was the state of the art architecture. Inceptnet outperformed models with the higher parameter count 5x5 and 3x3 modules.

The inception modules are stacked repeatedly (see fig. 4.3), varying the number of filters in each brach of the incept modules in each layer of modules. Eight total inception modules were used in the network with an input and output network, which can be seen as the repeating structures in the figure.

The input network is a 2D convolution used to scale up the number of channels. Since the graph representation of the network is too fine to be clear in this format, the figure 4.4 takes a closer look at the input network and the first incept module. The output network flattens the output and passes it through a 128 neuron fully connected layer before reducing it to the output dimension of the network. While this was a reduction in the original scale of the Inceptnet design because the input images are higher resolution there are far more weights per step. For example, with an image resolution of 80% of the original image size (260x1040 pixels) and a batch si ze of 10, the amount of ram needed to store the weights for the forward and back propagation step alone is 21.4 GB, which exceeds the video memory of all but the highest end consumer graphics cards. Since one of the questions being addressed by this thesis will be the impact of scaling the input image resolution on the accuracy, it is important to design a model that runs with the hardware available.

Once the broad architecture was selected the next step is to tune the model. Tuning the model helps model level parameters, like the number of filters at a given module, to be optimized for the use case. Tuning was done systematically with Optuna [25] and results were recorded and compared using Tensorboard [34] and MLFlow [21].

The labels of the data set are the expected values the network will output given a specific input. ι, the rotational transform or the ratio of poloidal transits per toroidal turn of any field line on a flux surface, is the label used for training. Initially, the labels for the data set were not available since simulations were necessary to produce them and that process is computationally expensive. Instead, simplified versions of the labels which just took into account the sum of I_A and I_B, which provides a good first order approximation. Naturally, this will neglect the plasmas induced changes to the magnetic field. Simulations using VMEC allowed for a more accurate approximation for ι. However, it is prohibitively computationally intensive to run 22,000+ simulations using extender, a simulation tool necessary for extending the results from VMEC to regions outside of the plasma and the field line tracer, a tool used for tracking field lines from the area outside of the plasma to the internal

Fig. 4.3: A diagram of the entire inception network as used in this paper.

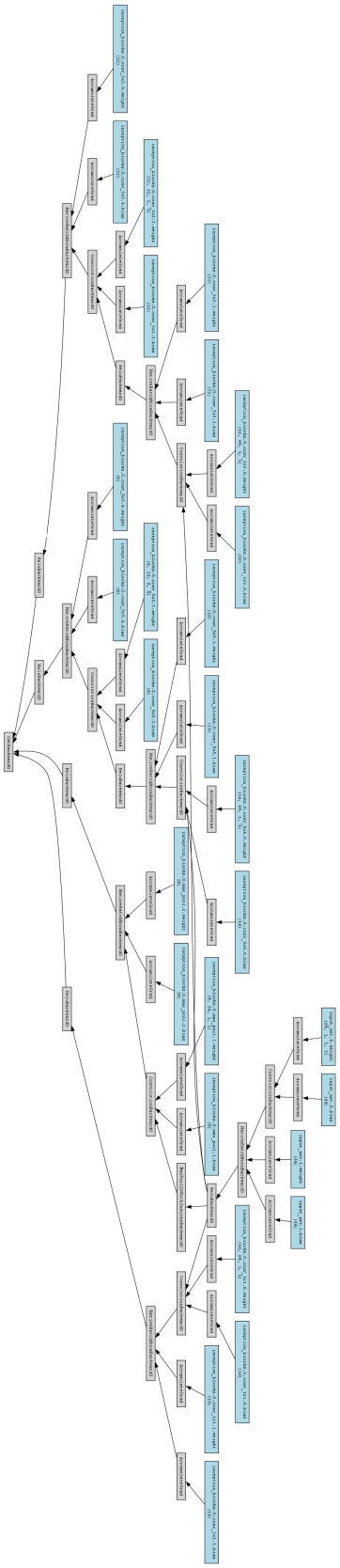

Fig. 4.4: A subsection of 4.3 diagram looking at the input network and the first inception module.

PFCs. The compromise is to use a value of ι at the edge of the plasma, as the VMEC simulations can be made much faster alone.

The design philosophy of this project was important. The code was designed to be a complete package, to be easily used by fellow colleges, and to work with the state of the art tools available in the machine learning space. PyTorch [27] was used at the primary framework, as many of members of the NEISS group used this as well. The PyTorch Lightning [28] package provided a powerful way to design an object-oriented machine learning package. For lower-level operations, like managing and recording the network configuration files, hyperparmeter searches, as well as structuring and analyzing results, the package Hydra [16], developed by Facebook, was used. Git [11] was used for the version control system and DVC [9] for data version control. This project was profiled and optimized with help from the PyTorch Lightning profiler. Code comments and documentation have been developed to help with quick adoption of the code for future use.

Since future changes to the divertor surface will result in drastically different inputs to the network, the code was designed to automatically scale with the inputs. The design philosophy was focused on making a project that could be quickly and easily taken over by a future student and be powerful and flexible enough to work with expected changes in the future. Reguardless of whether or not the code will never used again it is still a valuable lesson in Python software development and MLops. All of the code is available on GitHub at the following link:

`https://github.com/natephysics/hfcnn`

5 Data

Since the heat load image data was taken from real experiments, the data set is not uniform in the distribution of $\bar{\iota}$ values. This means the distribution the network is attempting to learn is not uniform, or more explicitly, it's multimodal. This is a common problem in machine learning and is often addressed by using a loss function that is robust to multimodal distributions. The loss function used in this case is the mean squared error (MSE). The MSE is a common loss function for regression problems, but it is not entirely robust to multimodal distributions. This was address by M. Blatzheim et al. [4] by using simulated input data that provided a continuos extension to the output distribution.

6 Results

6.1 Input resolution scaling

Better understanding the impact of input resolution on the neural network training results is important for optimizing the network. The neural network was trained on inputs of varying resolution. The resolution of the inputs was scaled and the network parameters were scaled accordingly to the inputs, so larger resolution images also had more network parameters.

In table 6.1 the input resolutions and the corresponding number of multiply–accumulate operations (MAC) are presented. MACs are a commonly used metric in machine learning because many of the tensor optimized accelerators compute $a \cdot x + b$ as a single operation [26], therefore one MAC is approximately 2 floating point operations (FLOPS).

When scaling the resolution to the labels used by the neural network there, it can been seen in figure 6.1 that higher resolution inputs do not necessarily result in better training results. In the case of the inception network that was tested, larger input resolutions also result in higher numbers of parameters and longer training times. The larger number of parameters could make it easier to overfit the network, especially with the limited size of the data set. However, when the resolution of the inputs is below 16900 pixels (260x65), the network is unable to resolve the features necessary to achieve an $rmse$ below $6 \cdot 10^{-3}$. If more training data was available, the network would be able to learn the features necessary to achieve a lower $rmse$ at the same resolution.

6.2 Network Performance

It is difficult to completely disentangle the increased network performance from the increased number of network parameters. However, it is not necessary to do so because ultimately the goal is to find a network that can be trained in a reasonable amount of time and that can be used to predict $\bar{\iota}$ values. The increased computational

Network input resolution, number of multiply–accumulate operations.		
x resolution	y resolution	MAC (G)
780	195	235.15
520	130	104.39
260	65	26.31
156	39	9.49
104	26	5.65
52	13	1.09

Tab. 6.1: This table contains the input resolutions for 6 neural network architectures and their corresponding multiply–accumulate operations, which is a measure of the computational resources needed to train the network. To facilitate the larger input size the network adjusts the number of parameters, which results in an increase of computational resources to train.

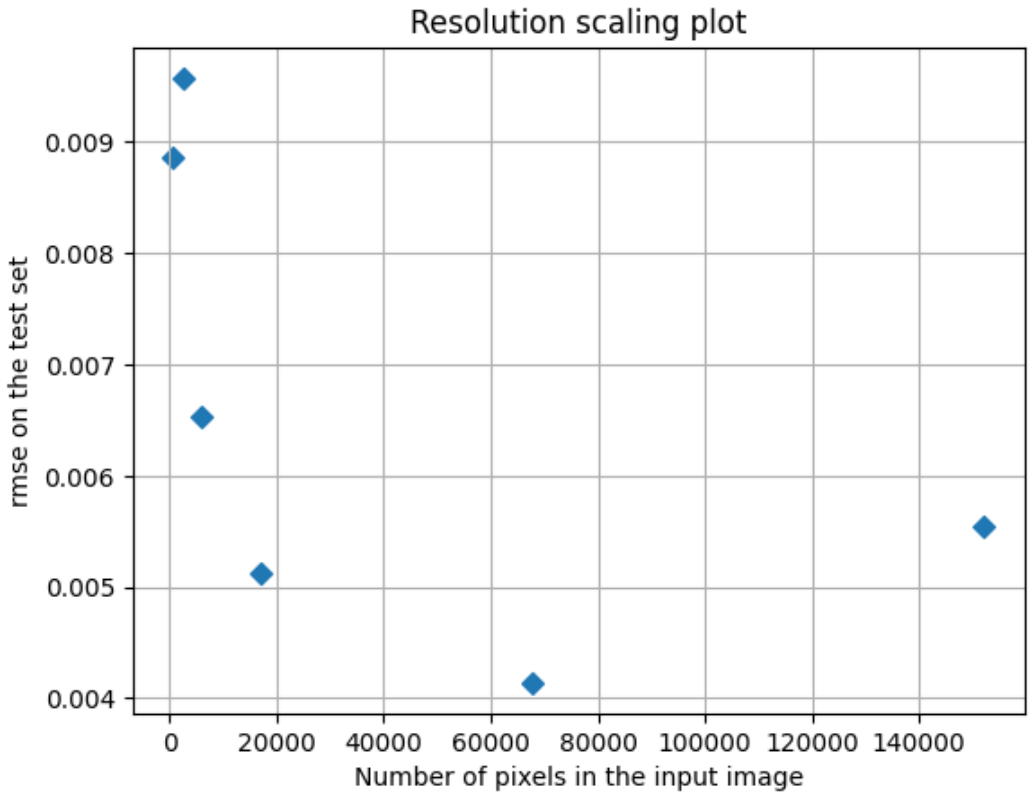

Fig. 6.1: $rmse$ after training the neural network on the inputs at a given resolution over number of pixels in the input image. The pixel values are a product of the input image from the thermal camera height and width values after being scaled.

costs of training the network amounts to a few hours of training time on a single GPU. The network that was trained on the 520x130 input images was able to achieve this goal. The 520x130 network was able to achieve an $rmse$ of $4.13 \cdot 10^{-3}$ in 8 hours of training time on a single GPU, and the next best result was given by the 260x65 network with an $rmse$ of $5.12 \cdot 10^{-3}$ for a 50% reduction in overall training time.

Figure 6.2 shows the simulated $\bar{\iota}$ values vs the neural network reconstruction of $\bar{\iota}$ for the validation and test data sets for the two highest performance networks. As mentioned in section **??**, the neural network was trained on a data set that was split into training, validation, and test data sets but the distribution of $\bar{\iota}$ values was not uniform. This can be clearly seen in this graph, with the five independent programs worth of data being used for testing. The values for the test data set are normalized to mean zero and standard deviation one for the purposes of this plot but in the actual training the training data set was used to normalize the test data. The reconstructed ι values are then normalized to the test set. It can be seen in the right plot, the network slightly overestimates lower $\bar{\iota}$ values and slightly underestimates higher $\bar{\iota}$ values. The left plot, which shows the highest performance model does not show as much bias as the right plot, but the bias is still present. The bias is likely due to the limited size of the data set and the limited distribution of $\bar{\iota}$ values.

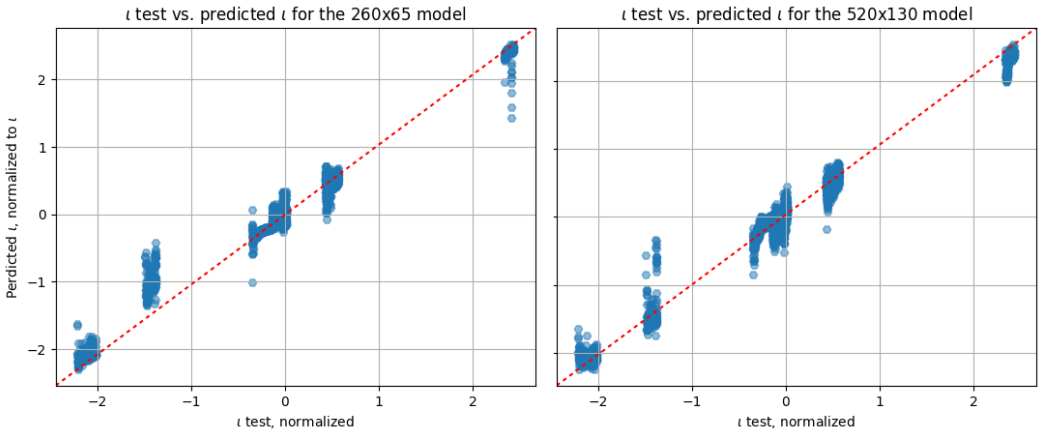

Fig. 6.2: $\bar{\iota}$ reconstruction with top two networks.

We can see a similar trend in the violin plots in figure 6.3. In this plot the distribution of the L2 error kernel density estimate (KDE) is shown for all six networks. It should be noted that the KDE is not a probability distribution, but it is a good way to visualize the distribution of the error, and it is a good way to compare the error distributions of the different networks. The point-wise L2 error is defined as:

$$\text{L2 error} = (y_i - \hat{y}_i)^2 \tag{6.1}$$

where y_i is the true value of $\bar{\iota}$ and $\hat{y}_i$ is the predicted value of $\bar{\iota}$. KDE is defined as:

$$\text{KDE} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h}\sum_{j=1}^{n}K\left(\frac{y_i - y_j}{h}\right) \tag{6.2}$$

where n is the number of data points, h is the bandwidth of the kernel, and K is the kernel function. The bandwidth is chosen to be 0.1 for all of the KDE plots. The kernel function used is the Gaussian kernel:

$$K(x) = \frac{1}{\sqrt{2\pi}}e^{-\frac{x^2}{2}} \tag{6.3}$$

Since values below zero are irrelevant to L2 error, the distributions are truncated at zero. The KDE is calculated for each network and then the KDE is normalized to the entire dataset. The KDE is then plotted as a violin plot, which is a box plot with the kernel density estimate plotted as a density function. The KDE is plotted as a density function because it is not a probability distribution. The two networks that exhibit best performance can be seen to have higher q-factors than the other networks.

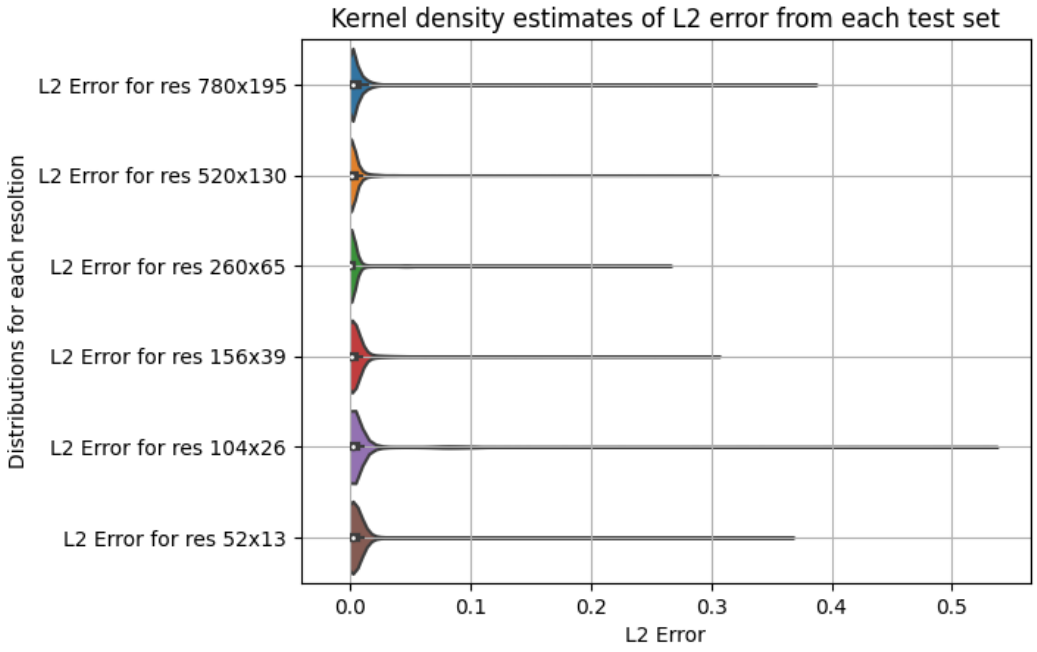

Fig. 6.3: Kernel density estimate of L2 error from each trained neural network on the test datasets.

7 Conclusion

In conclusion, this thesis presents a successful approach for inferring the edge rotational transform, $\bar{\iota}$, of the Wendelstein 7-X plasma experiment from heat load data. By training an inceptnet convolutional neural network on a dataset of infrared camera images, the network was able to infer the edge rotational transform with an $rmse$ of $4.13 \cdot 10^{-3}$. This is an improvement over prior work with similar data and indicates that the proposed approach is a viable solution for determining real-time extraction of $\bar{\iota}$ from the W7-X experiment. Since the network was trained on inputs with a 520x130 resolution, which is a good compromise between computational cost and network performance, further improvements can be quickly made by expanding the dataset, both in size and maybe with completely simulated labels for the gaps in the output $\bar{\iota}$ distribution.

Bibliography

[1] C. Beidler et al. “Physics and Engineering Design for Wendelstein VII-X”. In: 17 (1990) (cit. on p. 6).

[2] R. König et al. “The divertor program in stellarators”. In: 44 (2002) (cit. on p. 7).

[3] T. Klinger et al. “Towards assembly completion and preparation of experimental campaigns of Wendelstein 7-X in the perspective of a path to a stellarator fusion power plant”. In: 88 (2013) (cit. on p. 6).

[4] Marko Blatzheim, Daniel Böckenhoff, Hauke Hölbe, et al. “Neural network performance enhancement for limited nuclear fusion experiment observations supported by simulations”. In: *Nuclear Fusion* 59.1 (Dec. 2018), p. 016012 (cit. on pp. 11, 23).

[5] Marko Blatzheim, Daniel Böckenhoff, and the Wendelstein 7-X Team. “Neural network regression approaches to reconstruct properties of magnetic configuration from Wendelstein 7-X modeled heat load patterns”. In: *Nuclear Fusion* 59.12 (Oct. 2019), p. 126029 (cit. on pp. 12, 17).

[6] Daniel Böckenhoff, Marko Blatzheim, Hauke Hölbe, et al. “Reconstruction of magnetic configurations in W7-X using artificial neural networks”. In: *Nuclear Fusion* 58.5 (Mar. 2018), p. 056009 (cit. on pp. 9, 10).

[7] Daniel Böckenhoff, Marko Blatzheim, and the W7-X Team. “Application of improved analysis of convective heat loads on plasma facing components to Wendelstein 7-X”. In: *Nuclear Fusion* 59.8 (July 2019), p. 086031 (cit. on p. 17).

[8] Francis F Chen. *Introduction to plasma physics*. Springer Science & Business Media, 2012 (cit. on p. 5).

[9] *Data Version Control* (cit. on p. 22).

[10] Andreas Dinklage, CD Beidler, Per Helander, et al. “Magnetic configuration effects on the Wendelstein 7-X stellarator”. In: *Nature Physics* 14.8 (2018), pp. 855–860 (cit. on p. 6).

[11] *Git* (cit. on p. 22).

[12] G Grieger, W Lotz, P Merkel, et al. “Physics optimization of stellarators”. In: *Physics of Fluids B: Plasma Physics* 4.7 (1992), pp. 2081–2091 (cit. on p. 6).

[13] Per Helander. “Theory of plasma confinement in non-axisymmetric magnetic fields”. In: *Reports on Progress in Physics* 77.8 (2014), p. 087001 (cit. on pp. 3, 5).

[14] SP Hirshman, P Merkel, et al. “Three-dimensional free boundary calculations using a spectral Green’s function method”. In: *Computer Physics Communications* 43.1 (1986), pp. 143–155 (cit. on p. 6).

[15] Steven P Hirshman and JC Whitson. “Steepest-descent moment method for three-dimensional magnetohydrodynamic equilibria”. In: *The Physics of fluids* 26.12 (1983), pp. 3553–3568 (cit. on p. 6).

[16] *Hydra* (cit. on p. 22).

[17] Sergey Ioffe and Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. 2015 (cit. on p. 18).

[18] Marcin Jakubowski, Peter Drewelow, Joris Fellinger, et al. “Infrared imaging systems for wall protection in the W7-X stellarator”. In: *Review of Scientific Instruments* 89.10 (2018), 10E116 (cit. on pp. 7, 8).

[19] A Knieps, Y Suzuki, J Geiger, et al. “Plasma beta effects on the edge magnetic field structure and divertor heat loads in Wendelstein 7-X high-performance scenarios”. In: *Nuclear Fusion* 62.2 (2021), p. 026011 (cit. on p. 7).

[20] Min Lin, Qiang Chen, and Shuicheng Yan. *Network In Network*. 2013 (cit. on p. 17).

[21] *MLflow: An Open Source Platform for the Entire Machine Learning Lifecycle* (cit. on p. 19).

[22] HK Moffatt. “Magnetostatic equilibria and analogous Euler flows of arbitrarily complex topology. Part 1. Fundamentals”. In: *Journal of Fluid Mechanics* 159 (1985), pp. 359–378 (cit. on p. 5).

[23] H Niemann, P Drewelow, MW Jakubowski, et al. “Large wetted areas of divertor power loads at Wendelstein 7-X”. In: *Nuclear Fusion* 60.8 (2020), p. 084003 (cit. on p. 8).

[24] J Nührenberg and R Zille. “Stable stellarators with medium β and aspect ratio”. In: *Physics Letters A* 114.3 (1986), pp. 129–132 (cit. on p. 6).

[25] *Optuna: A hyperparameter optimization framework* (cit. on p. 19).

[26] *Programming tensor cores in CUDA 9*. Aug. 2022 (cit. on p. 24).

[27] *PyTorch* (cit. on p. 22).

[28] *PyTorch Lightning* (cit. on p. 22).

[29] H. Renner, J Boscary, H Greuner, et al. “Divertor concept for the W7-X stellarator and mode of operation”. In: *Plasma physics and controlled fusion* 44.6 (2002) (cit. on p. 7).

[30] Thomas Rummel, Konrad Ribe, Gunnar Ehrke, et al. “The superconducting magnet system of the stellarator Wendelstein 7-X”. In: *IEEE Transactions on Plasma Science* 40.3 (2012), pp. 769–776 (cit. on p. 6).

[31] B Sieglin, M Faitsch, A Herrmann, et al. “Real time capable infrared thermography for ASDEX Upgrade”. In: *Review of Scientific Instruments* 86.11 (2015), p. 113502 (cit. on p. 7).

[32]Y Suzuki and J Geiger. “Impact of nonlinear 3D equilibrium response on edge topology and divertor heat load in Wendelstein 7-X”. In: *Plasma Physics and Controlled Fusion* 58.6 (2016), p. 064004 (cit. on p. 7).

[33]Christian Szegedy, Wei Liu, Yangqing Jia, et al. *Going Deeper with Convolutions*. 2014 (cit. on p. 17).

[34]*TensorBoard: Visualizing Learning* (cit. on p. 19).

List of Figures

List of Tables

Declaration

You can put your declaration here, to declare that you have completed your work solely and only with the help of the references you mentioned.

City, Dec 10th, 2022

Nathan Belmore