Machine Learning Accelerated Tomographic Reconstruction for Multispectral Imaging on TCV

van Leeuwen, Loek S.

Award date:
2022

Machine Learning Accelerated Tomographic Reconstruction

for Multispectral Imaging on TCV

Loek van Leeuwen

Supervised by:
Jonathan Citrin and Maarten Schoukens

Science and Technology of Nuclear Fusion
Electrical Engineering: Artificial Intelligence for Engineered Systems

Machine Learning Accelerated Tomographic Reconstruction for Multispectral Imaging on TCV

Loek van Leeuwen

Supervised by:

Jonathan Citrin
Maarten Schoukens

Eindhoven, November 7, 2022

Acknowledgements

First and foremost I would like to thank my supervisors Jonathan and Maarten, with whom I met almost every week to discuss new ideas and to whom I tried to explain what I found (and in that process understanding it better myself!). You motivated me to try out new things, but at the same time kept me focussed when I got distracted. Meeting every Wednesday kept me on track, but also forced me to look back and realize what I had done that week. I am very grateful to you both for your help and supervision.

I would also like to thank Artur and all other people at the SPC, for introducing me to everything MANTIS and TCV related, and making Lausanne feel like home; you taught me a great deal, and all your answers to both my simple and complex questions helped me a lot.

Eefje, you helped me a lot with my project: from proofreading everything I wrote, to lending me an ear to tell all my ideas and mini day-planning to. Without you, this thesis would be full of weirdly constructed sentences, more difficult to read, and much more repetitive[1].

Finally, I would like to thank everyone else: family, friends, colleagues and housemates, for listening to my stories about how cool nuclear fusion is, and taking my mind off my thesis when needed.

[1] Also thanks a lot for introducing me to https://www.thesaurus.com

Abstract

A multispectral camera setup is used to infer a 2D map of plasma parameters in a tokamak from spectral emissions. However, the light measured by these cameras is line integrated in the toroidal direction, whereas emissivities on the poloidal plane are necessary for the inference. The poloidal plasma emissivity can be obtained by tomographic reconstruction, but classical techniques are too slow to use these emissivities for real-time control. To understand the strengths and shortcomings of the model-based approaches, we analyse the existing methods, and use this knowledge into the development of the model-informed deep learning architectures.

We present two machine-learning based approaches to accelerate the reconstruction of the poloidal emissivities. One based on the back-projection operator, with a learned non-linear filter to process the back-projection. The other approach based on the model-based iterative approach, where the proximal projection operators are learned.

Both approaches yield more accurate results on synthetic data than the iterative approach while being near fast enough for real-time control applications. The two approaches generalize well to other machines and geometries, such as MAST-U. Only changing the geometry matrix and retraining is required, but no additional tuning is needed. An additional analysis is done to find out what the options are to transfer these machine-learning based solutions from a local development and research environment to a production environment.

Contents

1 Introduction

1.1 Nuclear Fusion

It is currently estimated that the world energy demand in 2050 will be a factor of two to three times the current energy demand, caused by social economical developments all over the world [1]. Ideally, the present and future energy production is from a sustainable source. One of these future sustainable energy sources is nuclear fusion power, where energy is created by fusing together deuterium and tritium, which are isotopes of hydrogen. In this reaction a helium ion and a neutron are created, both with a high amount of kinetic energy:

$$ {}^{2}_{1}\mathrm{H} + {}^{3}_{1}\mathrm{H} \rightarrow {}^{4}_{2}\mathrm{He}(3.5\,\mathrm{MeV}) + {}^{1}_{0}\mathrm{n}(14.1\,\mathrm{MeV}). \tag{1.1} $$

A sufficient reaction rate requires high temperatures in the order of 10 keV to 20 keV. At these temperatures the fuel is ionized and becomes a plasma. Because the plasma consists of charged particles, it can be contained with magnetic fields created by (superconducting) coils. In the core of the plasma the particles gyrate around closed magnetic field lines, constraining them to the magnetic surface defined by these magnetic field lines. However, by collisions with other particles on nearby magnetic surfaces and by turbulence, the particles (and thus the heat) move perpendicular to the magnetic field lines. This results in a particle and heat flux out of the plasma core towards the edge of the plasma, which is called the scrape-off-layer (SOL).

The helium particles created by the fusion reaction are not desired in the plasma as they dilute the relative fuel density. Thus, to improve the avoidance of helium ash build-up, as well as avoiding the accumulations of impurities sputtered from the wall, a specific magnetic 'divertor' geometry was designed that better isolates the plasma exhaust region from the main core plasma [2, 3].

To avoid damage to the divertor wall materials, the heat flux and particle flux of the plasma in this region should be minimized [2]. By careful injection of neutral particles, such as nitrogen or hydrogen, one can control this heat and particle flux towards the target [4]. In the case of hydrogen neutrals injection, also called fuelling, the plasma will transfer energy and momentum to these neutral particles [5]. In the case of nitrogen injection, called seeding, the energy loss is mainly caused by an increase in radiation power [6]. The atomic and molecular reactions causing a loss of energy and momentum result in a colder and denser plasma near the divertor target. The state with a pressure gradient along the field lines to the target, in combination with a low pressure at the target, is referred to as detachment [7], which should be controlled for optimal performance and material durability.

The aim of detachment control is to make sure that the heat and particle fluxes towards the target do not exceed the material threshold, such that the divertor tiles do not get damaged [8]. However, one would also need to ensure that the plasma does not cool too much, as this can result in the detachment front moving towards the core. If the front comes too close to the x-point, the plasma closer to the core cools too much, resulting in a degradation of the plasma confinement and performance [9].

1.2 Multispectral Imaging on TCV

For control of the fluxes in the divertor region, one also needs measurements in this region and a way to reconstruct the state using these measurements. In this work we focus on the reconstruction for the

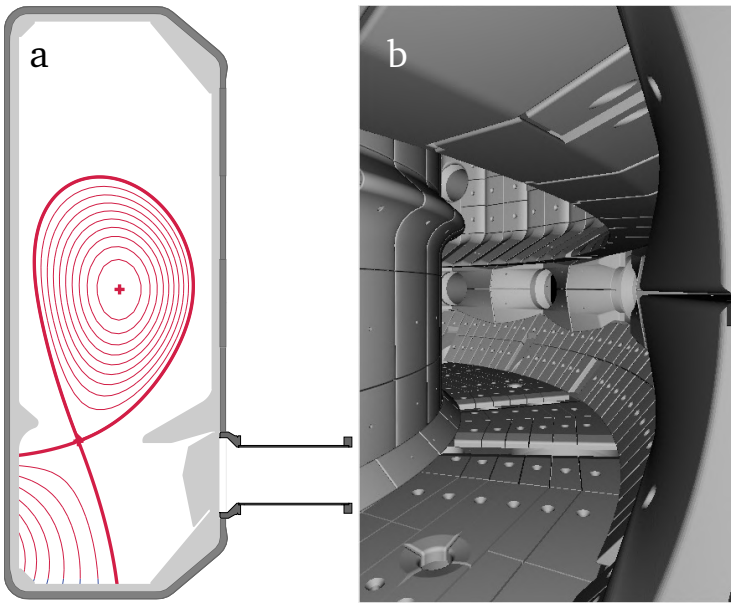


Figure 1.1: *(a) A poloidal cross-section of the TCV tokamak, with the magnetic field lines indicated in red. The outer red blue line indicates the scrape-off-layer (SOL), which diverts the particles towards the targets. (b) The view of the MANTIS camera in the tokamak is not perpendicular to the poloidal plane, but at an angle, measuring line-integrated emissions. Adapted from [5].*

Tokamak à configuration variable (TCV), at the Swiss Plasma Centre in Lausanne. Various diagnostics, such as bolometry [10], Langmuir probes [11], and multispectral imaging [12, 13] are used in the TCV to reconstruct the plasma state. The benefit of the latter is that it is real-time capable, non-invasive, and allows for 2D imaging, and has been achieved on TCV using the Multispectral Advanced Narrowband Tokamak Imaging System (MANTIS). Figure 1.1 illustrates the location of the MANTIS camera, along with the rendered view from the same angle as the MANTIS camera. Using this camera system, feedback control of the detachment front has already been achieved on TCV [14].

In the core of a fusion plasma, the electron temperature is high enough to assume that the hydrogen ions are fully ionized. This means that no hydrogen ion-electron reactions take place in the core, from which photons are emitted. Mind that this is not the case for some impurities, which can be not fully ionized in the core. We will focus on the excitation-relaxation reaction and the recombination reaction.

In the excitation reaction a collision with an electron excites an ion [15]:

$$e + A \rightarrow e + A^{*,n}, \tag{1.2}$$

where A^* is an atom in the excited state. The n is the principal quantum number, denoting the energy level above the ground state. The excited atom will undergo radiative decay resulting in the emission of energy, which was previously kinetic energy, as a photon:

$$A^{*,n} \rightarrow A^{*,k} + h\nu, \tag{1.3}$$

where k is a principal quantum number, h the Planck constant and ν the frequency of the photon. In the case where $k = 2$ and $A = H$, the transitions from any n are referred to as the Balmer lines. Another possibility of radiation is by recombination, in which an ion recombines with an electron, releasing energy in the process:

$$A^{+,n} + e \rightarrow A^{*,k} + h\nu. \tag{1.4}$$

These emitted photons can then be analysed and used to infer the electron temperature, electron density and neutral density [16, 17, 18], using the MANTIS camera system.

The MANTIS camera system allows for measuring up to ten different spectral lines simultaneously. The spectral filters are chosen such that the excitation-relaxation and the recombination reactions of the species in the plasma are inferred. The reactions rates are a function of the temperature, ion density and neutral density in the plasma. Thus, by using the 2D emission information from the different emission lines emitted

by different species in the plasma, combined with collisional-radiative modelling, one can make a 2D reconstruction of multiple plasma parameters in the divertor region, which define the plasma state in this region. These 2D maps would then allow for advanced control schemes by determining particle sources and sinks. The measurements MANTIS makes are however line integrated, meaning that light from different (R, Z)-coordinates is summed. For plasma state reconstruction one needs a poloidal cross-section of the emissions, rather than line-integrated measurements. The inversion process from the measured camera image to the poloidal plane is computationally intensive, making it unsuitable for real-time control. The plasma dynamics have a timescale of approximately 15 ms. To prevent sampling issues, we need to sample twice as fast, and in practice a factor of eight to ten is used. This means that the inversion calculation should take less than 2 ms [14], which is difficult to achieve using traditional iterative techniques.

1.3 Machine Learning

Neural networks have been shown to excel at solving a range of regression and classification tasks, including but not limited to machine translation, time series analysis, data-driven control, image classification, style transfer and image generation [19, 20, 21]. Because of this major success in a range of fields, we are interested if these techniques can also be applied to our problem setting, with the goal of reducing the inversion estimation time.

Tomographic reconstructions are also necessary in other fields of science dealing with imaging, such as geophysical imaging and medical imaging. Research in these fields shows that neural networks can be used to approximate the tomographic inversion while being computationally more efficient [22, 23] and can even yield more accurate results. Moreover, neural networks have already been used in fusion tomography applications, where 1D bolometry data in JET and soft X-ray data in COMPASS was used to reconstruct 2D profiles [24].

1.4 Research Questions and Thesis Outline

In this study we investigate how we can achieve a fast poloidal emissivity reconstruction in the divertor region of TCV, using (model-informed) machinelearning techniques. We will answer the following research questions:

How can the poloidal emissivity of a plasma in the TCV tokamak be reconstructed from MANTIS measurements for real-time control?

How can machine-learning be used in the estimation of tomographic reconstructions, incorporating physical knowledge of the system?

In Chapter 2 we introduce the imaging problem and discuss the existing iterative techniques that are used to solve this problem. Next, the theory behind convolutional neural networks and their application in reconstruction problems is discussed in Chapter 3. The main results are presented and discussed in Chapter 5, and is to be submitted for publication in the journal Nuclear Fusion. Information about the real-time deployment is presented Chapter 7. Finally, the work is concluded, and ideas for future work are introduced in Chapter 8.

2 Classical Methods for Tomographic Reconstructions

In this chapter we first look into how we can model the tomographic reconstruction problem. Then we discuss several existing reconstruction methods, which form the basis for the machine learning based approaches in the chapter thereafter.

2.1 Linear Imaging Model

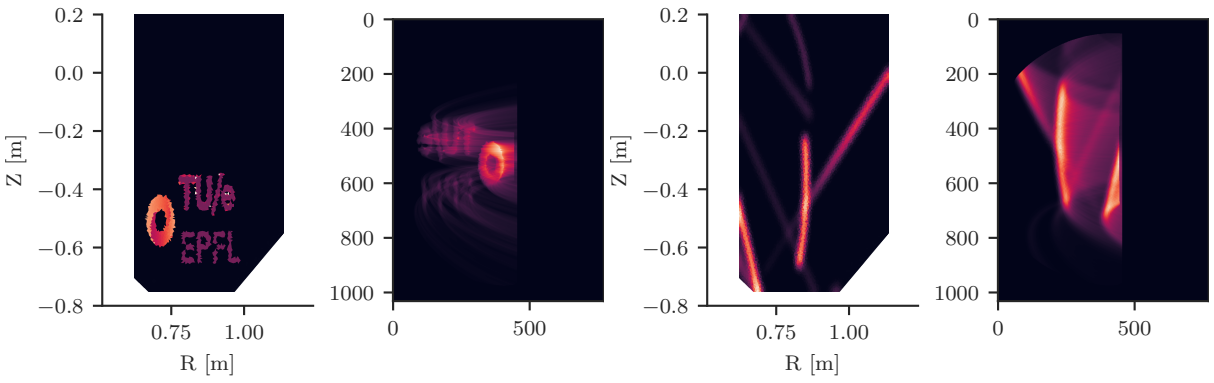
To reconstruct the emissivity of the plasma in the poloidal plane, we need a forward model $g : X \to Y$, where we define X as the poloidal space, and Y as the camera image space. If we now assume that the measured intensity on the camera cells are a linear combination of the emissivities in different locations on the poloidal plane, we can use a linear map, represented by the matrix G:

$$\mathbf{y} = G\mathbf{x} + \mathbf{e}, \tag{2.1}$$

where $\mathbf{y} \in Y$ is a measured camera image, $\mathbf{x} \in X$ the poloidal emissivity and $\mathbf{e}$ the measurement noise, but also any unmodeled effects. The matrix G is referred to here as the geometry matrix, which is constructed by point fitting locations in a measured image to a CAD model of the TCV tokamak [13] using the Calcam software package [25]. This package performs a ray-tracing from each pixel sensor to each grid cell in X and thus constructing G while assuming toroidal symmetry.

The camera sensor has a resolution of 1032 by 772 pixels. The inversion grid is generally a triangular grid consisting of 14 549 cells. This results in a large and tall geometry matrix $G \in R_+^{796704\times14549}$. Because of the carbon tiles in the TCV tokamak which have a low reflectivity, we can model the projection to the camera without any reflections. Each camera cell now only measures the emissivity from a few poloidal cells, resulting in a high sparsity of G, approximately 0.9955. Two examples of a forward modelled poloidal emissivity are shown in Figure 2.1.

In this figure we can see the mapping from the poloidal plane to the camera plane: because of the assumed toroidal symmetry, the emissivity turns around the centre column. We can also see that the camera image is



(a) *A poloidal emissivity from logos.* **(b)** *A poloidal emissivity from randomly drawn lines.*

Figure 2.1: *Two examples of poloidal plasma emissivities with their corresponding camera images.*

brighter at the high field side, while dimmer more towards the low field side, as is expected. Consequently, the data more towards the high field side and lower Z-values, is easier to reconstruct.

2.2 The Reconstruction Problem and Methods

Reconstructing the poloidal emissivity from the measured camera image can be formulated as an (unconstrained) optimization problem for some norm p:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \|\mathbf{y} - G\mathbf{x}\|_p = \arg\min_{\mathbf{x}} L(\mathbf{x}), \tag{2.2}$$

where L is the optimization objective, also referred to as the cost function or the loss function. If we use the Euclidean norm ($p = 2$) and square it, the objective reduces to the least squares objective:

$$L(\mathbf{x}) = \|\mathbf{y} - G\mathbf{x}\|_2^2. \tag{2.3}$$

2.2.1 Direct Reconstruction Methods

First we analyse direct reconstruction methods. This method is not used in the end, but is included for discussion, context and completeness. We can solve the optimization problem using the direct method by first rewriting the optimization objective as:

$$\frac{1}{2}\|G\mathbf{x} - \mathbf{y}\|_2^2 = \frac{1}{2}(G\mathbf{x} - \mathbf{y})^T(G\mathbf{x} - \mathbf{y}) = \frac{1}{2}\mathbf{x}^T G^T G\mathbf{x} + \mathbf{y}^T G\mathbf{x} + \frac{1}{2}\mathbf{y}^T\mathbf{y}, \tag{2.4}$$

where the factor $\frac{1}{2}$ can be added without changing the optimum. If we now take the gradient of the objective function with respect to $\mathbf{x}$:

$$\nabla L(\mathbf{x}) = G^T(G\mathbf{x} - \mathbf{y}), \tag{2.5}$$

from which we can express the normal equations as:

$$L(\mathbf{x} = 0) \implies G^T G\mathbf{x} = G^T\mathbf{y}. \tag{2.6}$$

If we now assume that G is full rank, we find the least squares solution:

$$\mathbf{x}^* = \overbrace{(G^T G)^{-1} G^T}^{G^+}\mathbf{y}, \tag{2.7}$$

where G^+ is called the Moore-Penrose pseudo-inverse. However, we can not assume that in our case G is full rank, and that the pseudo-inverse can be used. Or to be more specific, the product $G^T G$ should be non-singular.

In the case that G is not full rank, we can calculate (an approximation to) the pseudo-inverse using the (truncated) singular value decomposition (SVD). The SVD is a matrix factorization for $G \in \mathbb{R}^{m\times n}$ such that:

$$G = U\Sigma V^T, \tag{2.8}$$

where $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ are unitary matrices and the columns of these matrices are called the left- and right-singular vectors respectively. $\Sigma \in \mathbb{R}^{m\times n}$ is a diagonal matrix containing the singular values. The pseudo-inverse can now be calculated as:

$$G^+ = (U\Sigma V^T)^{-1} = V\Sigma^+ U^T, \tag{2.9}$$

where Σ^+ is calculated by taking the reciprocal of all non-zero (singular) values. The problem however with this factorization is the loss of sparsity: the matrices we use in this thesis all have a high sparsity. The SVD on the other hand does not necessarily produce in a sparse factorization, resulting in more memory usage to store the matrices and more operations to perform calculations with these matrices. We can rewrite Equation (2.9) to minimize the density of the 'inverse information':

$$G^+ = V\Sigma^+ U^T G^T. \tag{2.10}$$

Another problem with this approach is the possibility of a (near) singular G^+: combined with a noisy measurement, approximating the inverse might result in an amplification of the noise. Both problems

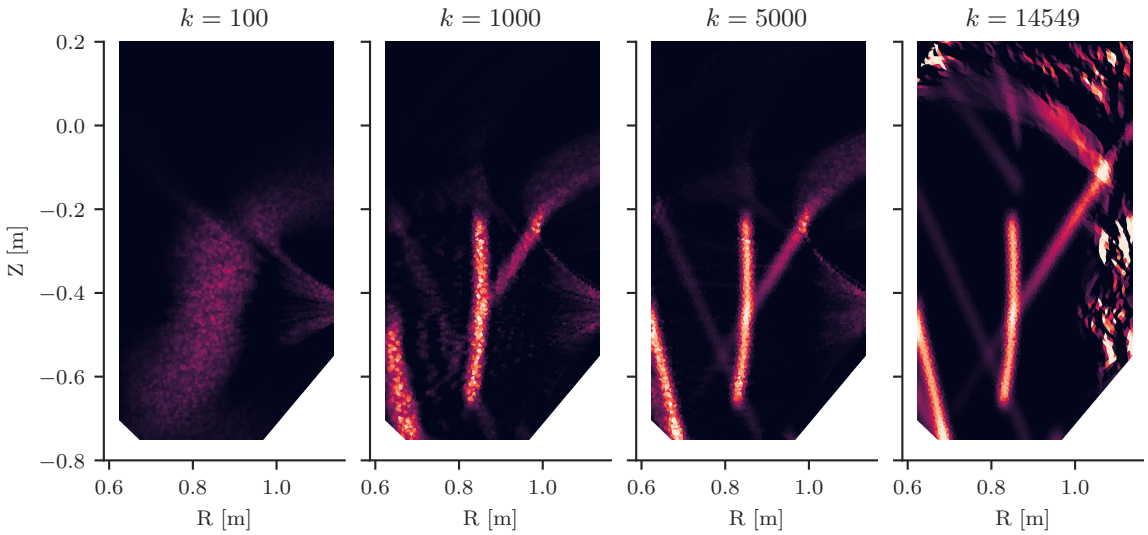


Figure 2.2: *Reconstruction of the poloidal emissivity using the truncated singular value decomposition for estimating the inverse geometry matrix. The value of k represents the number of singular values used.*

can be solved by using a truncated SVD. In this approach we reduce all the singular values smaller than some threshold to zero. The reduction of singular values also allows for removing values from the U and V matrices, reducing the memory need of the factorization. Furthermore, by only using the larger singular values we effectively regularize the problem making this approach less sensitive to noise, which we will further discuss in Section 2.3 for the other methods. The amplification of noise becomes more clear by rewriting the solution of the inverse problem with the SVD as:

$$\mathbf{x}^* = G^+\mathbf{y} = V\Sigma^+U^T\mathbf{y} = \sum_i \frac{\mathbf{u_i}^T\mathbf{y}}{\sigma_i}\mathbf{v_i}. \tag{2.11}$$

Here we can see that for smaller singular values the fraction becomes large really fast. There is a trade-off in how many singular values we want to use: a lower number of singular values results in less operations and memory usage, and regularizes more, while a larger number of singular values results in more operations and memory usage. The optimal low rank approximation $\tilde{G}_k$ for a rank k can be calculated as:

$$\tilde{G}_k = U_k\Sigma_k V_k^T, \tag{2.12}$$

where only the first k column and row vectors of U and V are used, and only the first k singular values from Σ, such that $U_k \in \mathbb{R}^{m\times k}$, $\Sigma_k \in \mathbb{R}^{k\times k}$ and $V_k \in \mathbb{R}^{k\times n}$. Using this low rank approximation we can similarly calculate a low rank approximation of the pseudo-inverse. This corresponds to filtering out the smaller singular values and components:

$$\tilde{G}_k^+ = \sum_i \phi_k(i)\frac{\mathbf{u_i}^T\mathbf{y}}{\sigma_i}\mathbf{v_i}, \tag{2.13}$$

with filtering function

$$\phi_k(i) = \begin{cases} 1 & i \le k \\ 0 & i > k \end{cases} \tag{2.14}$$

Figure 2.2 illustrates a reconstruction using the truncated singular value decomposition for different ranks k. This figure shows us that indeed for $k = 5000$ the reconstruction is already of good quality. However, in the full rank case artefacts are introduced because of numerical precision errors in calculating the singular value decomposition, amplifying these small differences. The truncated SVD yields great results for the estimation of the poloidal emissivity, but we will mainly focus on iterative methods because of the following reasons:

1. Calculating the singular value decomposition does not scale well with the dimensions of the geometry matrix, in particular with the number of reconstruction cells.
2. Prior physical knowledge, such as the emissivity should be positive, can be implemented into iterative algorithms more easily. Different regularization functions/cost functions can be used in iterative methods, while it is set for the singular value decomposition.

3. There less of an amplification of noise; iterative methods themselves regularize, especially the SIRT algorithm we discuss later. This also means that less tuning of the amount of singular values k with respect to the noise is necessary.

2.2.2 Iterative Reconstruction Methods

An approach to utilize the sparsity in G is to use an iterative approach. Instead of trying to approximate a direct solution, we iteratively update our estimate to approach a stationary point and thus a solution to the inverse problem. We will focus on line-search methods where the current estimate $\mathbf{x_k}$ is updated with a search direction $\mathbf{p_k}$ scaled by the step size μ_k [26]:

$$\mathbf{x_{k+1}} = \mathbf{x_k} + \mu_k \mathbf{p_k}. \tag{2.15}$$

If we use the gradient of the previously defined objective function in Equation (2.3) as the search direction, we use the steepest descent direction, resulting in the following iterative update scheme:

$$\mathbf{x_{k+1}} = \mathbf{x_k} - \mu \nabla f(\mathbf{x_k}) = \mathbf{x_k} - \mu G^T(G\mathbf{x_k} - \mathbf{y}), \tag{2.16}$$

which is known as the gradient descent algorithm, and the basis for many optimization algorithms (in machine learning). Finder a proper value for the step size, also known as the learning rate, is of importance: a step size that is too small will cause no change at all in $\mathbf{x_k}$, while a step size that is too large causes the solution to diverge. To find a good value for μ one could use for example exact line search methods or Armijo line search methods [26], however, we will not focus on these methods in this thesis.

Instead, we use a family of methods that has been specifically developed for CT imaging called the ART family. Specifically, we will use the simultaneous and iterative reconstruction technique (SIRT). This algorithm and variants (SART / ART) have been extensively used in the field of medical and geophysical imaging [27, 28, 29, 30, 31].

First we use the fact that G is a forward mapping which describes how all the pixels are combined and summed into inversion cells. This means that the transpose G^T projects the inversions cells back onto the pixels, i.e. it tells us for each inversion cell, which pixel correspond to that cell [32, 33].

To get more of a feeling for the forward mapping and its back projection, we work through a reduced projection example[1]. We assume a two-by-two square volume-grid, denoted by $[x_0, x_1, x_2, x_3]$. We are able to measure the integrated intensity along the diagonals and the edges of the square, as illustrated in Figure 2.3. The y_i denote the different sensors. This problem is similar to the original problem in that the number of sensors is larger than the number of volume cells, and that each volume cells is measured multiple times. If we assume a larger contribution for emissivities closer to the sensor (2), and a smaller

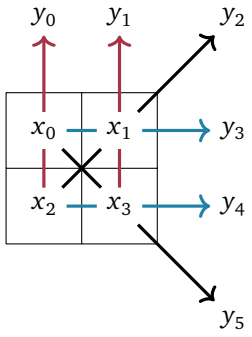


Figure 2.3: *An illustration of the setting for the reduced problem. We have emissivities from x_i, measured by different sensors y_i. The colours are used to differentiate between the different projection lines.*

contribution for those further away (1), we can find a mapping $G : X \to Y$ between the emissivities and the

[1]This example is inspired by and adapted from https://www.12000.org/my_notes/image_projection_matrix/index.htm

measurements as:

$$G = \begin{array}{c} \\ y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{array} \begin{array}{c} \begin{array}{cccc} x_0 & x_1 & x_2 & x_3 \end{array} \\ \begin{pmatrix} 2 & 0 & 1 & 0 \\ 0 & 2 & 0 & 1 \\ 0 & \frac{2}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 1 & 2 & 0 & 0 \\ 0 & 0 & 1 & 2 \\ \frac{1}{\sqrt{2}} & 0 & 0 & \frac{2}{\sqrt{2}} \end{pmatrix} \end{array}, \tag{2.17}$$

where an additional factor of $\frac{1}{\sqrt{2}}$ is added for the diagonal measurements. By transforming according to G we can thus find the measurement data y_i given the emissivities x_i. However, we can also express data in measurement space in emissivity space by multiplying with G^T. To understand what this means, we first consider only y_0 and y_1. If we would now do a back projection we would 'smear out' the data, according to G: consider the following emissivities: $\mathbf{x} = [2,4,1,3]^T$, then we would have $y_0 = 5$ and $y_1 = 11$. By only using the rows corresponding to y_0 and y_1 from the geometry matrix, we have that:

$$\hat{\mathbf{x}} = \overbrace{\begin{pmatrix} 2 & 0 \\ 0 & 2 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}}^{G^T} \overbrace{\begin{pmatrix} 5 \\ 11 \end{pmatrix}}^{\mathbf{y}} = \begin{pmatrix} 10 \\ 22 \\ 5 \\ 11 \end{pmatrix}. \tag{2.18}$$

In this example we can see that the back-projection does not estimate the emissivities. The relative values between $x_0 and x_2$ hold, but not $x_1 and x_3$. If we do the same, but now using all measurements, we obtain:

$$\mathbf{y} = \begin{pmatrix} 5 \\ 11 \\ \frac{9}{\sqrt{2}} \\ 10 \\ 7 \\ \frac{8}{\sqrt{2}} \end{pmatrix}, \qquad \hat{\mathbf{x}} = \begin{pmatrix} 24 \\ 51 \\ 16.5 \\ 33 \end{pmatrix}. \tag{2.19}$$

Again we do not recover the emissivity values $\mathbf{x}$, but we do recover approximate relative values. Thus, this example shows us that we can use a back-projection to project the measurement values back to emissivity space. This back-projection is not a good pseudo-inverse, but it is still very useful: if we have an error in measurement space, this is a way to express the error in emissivity space!

Going back to the original problem, Equation (2.16) can also be better understood with this knowledge: the term $G\mathbf{x_k}$ maps the current inversion estimate to the measurement space, after which the error in measurement space is calculated $G\mathbf{x_k} - \mathbf{y}$, which is then projected back to inversion space $G^T(\cdot)$. As previously mentioned, finding a good value for the step size is an iterative process, and the convergence speed and stability of the iterative algorithm depends on this choice. In the SIRT algorithm, the step size problem is solved by preconditioning the geometry matrix to ensure a stable convergence. First the error correction term $G\mathbf{x_k} - \mathbf{y}$ is multiplied with the matrix R, which is a diagonal matrix containing the inverse row sums of G, so $r_{ii} = 1/\sum_j g_{ij}$. This is equivalent to solving the weighted least-squares problem [32, 33, 34]:

$$L(\mathbf{x}) = \|G\mathbf{x} - \mathbf{y}\|_R^2. \tag{2.20}$$

The R matrix reduces the error contribution of pixels in measurement space that correspond with a longer ray length. Without this R, the steepest descent direction would be biased towards minimizing the error for pixels that receive the most contribution from the reconstruction space.

If we now precondition the normal equations corresponding to the weighted least squares problem by a matrix C we have that:

$$CG^T RG\mathbf{x} = CG^T R\mathbf{y}, \tag{2.21}$$

where C is a diagonal matrix containing the inverse column sums of G: $c_{jj} = 1/\sum_i g_{ij}$. The C matrix plays the same role for G^T as R does for G. To understand the effect of the C and R matrix better, we continue

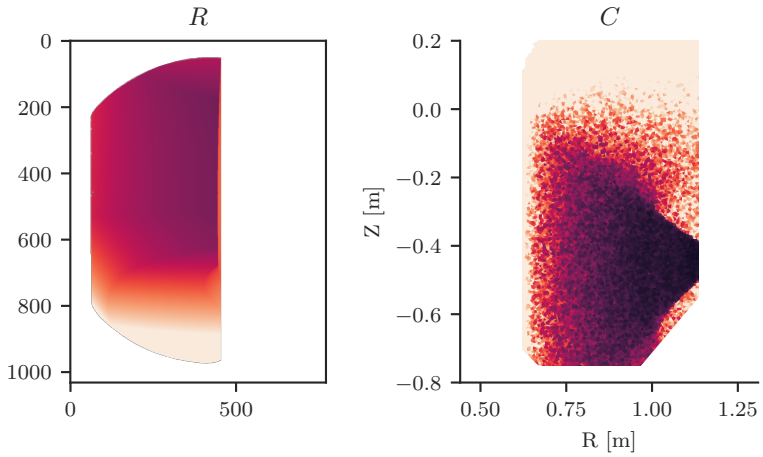


Figure 2.4: *The R and C matrices that 'precondition' in the SIRT algorithm. The R matrix compensates for the ray length and thus the amount of light that a pixel receives. The C matrix compensates for the amount of rays that contribute to a volume cell. The direct view line of the camera can be seen clearly in the values of C.*

the example from Figure 2.3. The R and C matrix are given by:

$$C = \begin{pmatrix} \left(3+\frac{1}{\sqrt{2}}\right)^{-1} & 0 & 0 & 0 \\ 0 & \left(4+\frac{2}{\sqrt{2}}\right)^{-1} & 0 & 0 \\ 0 & 0 & \left(2+\frac{1}{\sqrt{2}}\right)^{-1} & 0 \\ 0 & 0 & 0 & \left(1+\frac{3}{\sqrt{2}}\right)^{-1} \end{pmatrix}, \quad R = \begin{pmatrix} \frac{1}{3} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{3} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{\sqrt{2}}{3} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{3} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{3} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{\sqrt{2}}{3} \end{pmatrix} \tag{2.22}$$

If we now calculate $\hat{\mathbf{x}}$ from the SIRT-based back-projection we can see the effect of the C and R matrix on the emissivity estimates:

$$\hat{\mathbf{x}}_{\text{SIRT}} = CG^T R\mathbf{y} \approx \begin{pmatrix} 2.3 \\ 3.7 \\ 2.3 \\ 2.7 \end{pmatrix}. \tag{2.23}$$

This result shows us that using the SIRT-based back projection the estimated emissivity is already much closer to the actual emissivity (which we elaborate on later in this chapter). All values are close to the actual emissivity, except for x_2. To better understand the C and R matrix in context of the actual problem, the diagonals of the C and R matrix are shown in Figure 2.4. The effect of the C and R matrix on a measured image are shown in Figure 2.5. Going back to the actual problem, rewriting the normal equations in Equation (2.21) to an iterative update scheme results in:

$$\mathbf{x_{k+1}} = \mathbf{x_k} - \mu CG^T R(G\mathbf{x_k} - \mathbf{y}), \tag{2.24}$$

which can then be rearranged to:

$$\mathbf{x_{k+1}} = (I - \mu CG^T RG)\mathbf{x_k} - \mu CG^T R\mathbf{y}, \tag{2.25}$$

from which we can derive the stability properties of the update scheme. The following derivation is adapted from [32, 33]. For a stable convergence we must have that the term $(I - \mu CG^T RG)$ has a spectral radius $\rho(\cdot)$ smaller than one, i.e. all eigenvalues λ are inside the unit circle. The eigenvalues are equal to:

$$\lambda = 1 - \lambda_i;\ \lambda_i = \lambda_i(CG^T RG). \tag{2.26}$$

Because the matrices C, R and $G^T G$ are all positive semi-definite, we have that the eigenvalues λ_i are greater than or equal to zero. Next, we use that the spectral radius of a matrix is smaller than or equal to the matrix norm, for any natural matrix norm:

$$\rho(A) \leq \|A\|. \tag{2.27}$$

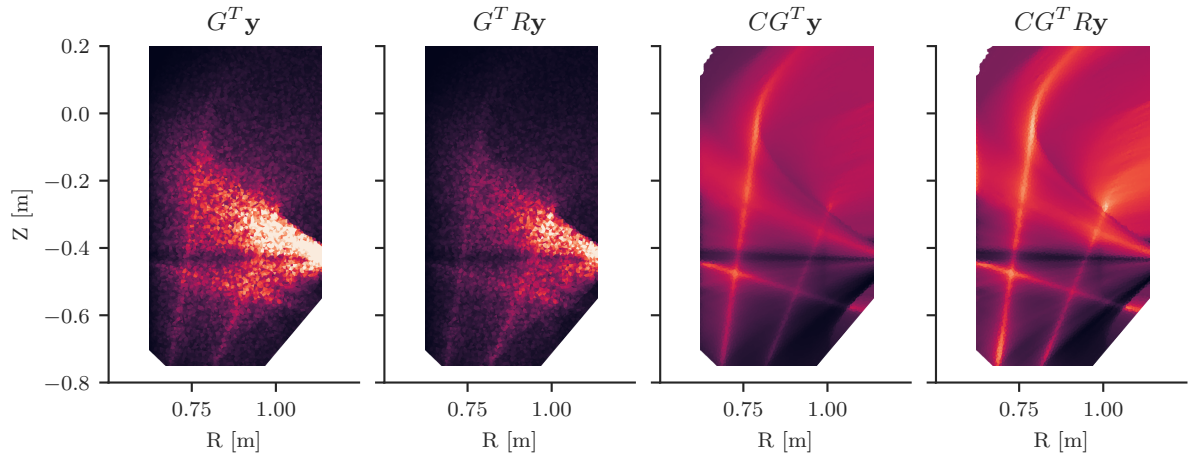


Figure 2.5: *The effect of the C and R matrix on the back projection of a camera image. The R matrix compensates for the ray length, which roughly corresponds with the radial coordinate: the rightmost poloidal plane shows a higher emissivity for higher Z[m] and R[m], compared to the third pane. The C matrix highlights the structure of the emissivity and compensates for the direct field of view of the camera.*

By rewriting this condition using the ∞ norm, we have that:

$$\rho(CG^TRG) \leq \left\|CG^TRG\right\|_\infty \leq \left\|CG^T\right\|_\infty \|RG\|_\infty, \tag{2.28}$$

where the last inequality follows from the sub-multiplicativity of a matrix norm. We choose to use the ∞ norm here, because the explicit expression for the ∞ norm of a matrix is given by the largest row sum:

$$\|A\|_\infty = \max_k \sum_l \left|a_{k,l}\right|, \tag{2.29}$$

which motivates the choice for the C and R to equal the inverse row (and column) sum of G. We now have that $\|RG\|_\infty = 1$ and $\left\|CG^T\right\|_\infty = 1$. Combining these results gives us:

$$\rho(I - \mu CG^TRG) = \max_i |1 - \mu\lambda_i| = |1 - \mu|. \tag{2.30}$$

The requirement for stability is that the spectral radius is smaller than one, so the bounds on the step size μ can be determined as:

$$|1 - \mu| < 1 \implies 0 < \mu < 2, \tag{2.31}$$

where a larger μ leads to a faster convergence.

2.3 Regularization Techniques for Inverse Problems

One problem with the methods described previously is that the system matrix G might be ill-conditioned, resulting in an amplification of measurement noise. Furthermore, these methods do not include any additional information about the physics of the problem or prior information we have, such as that the emissivity on the poloidal plane is always positive. One solution is to use regularization techniques, with which unwanted characteristics in $\mathbf{x}$ are penalized, and desired properties in $\mathbf{x}$ are promoted. We can add a regularization term to the objective function as:

$$L_r(\mathbf{x}) = \|G\mathbf{x} - \mathbf{y}\|_2^2 + \mu R(\mathbf{x}), \tag{2.32}$$

where μ is the regularization factor, and R the regularization function. In the case that the regularization function is convex and differentiable, we can solve the regularized problem by finding the normal equations corresponding with Equation (2.32).

It can however be the case that we want to use a regularization function that is not differentiable: in signal processing it is often known if a signal $\mathbf{x}$ is sparse or not. In this case, we would like to use the ℓ_1-norm as a regularization function, as this sparsifies the solution. However, the function corresponding to the ℓ_1-norm $R(\mathbf{x}) = \sum_i |x_i|$ is not differentiable at the origin. Instead of directly solving the equation, we can use the proximal gradient descent method [35], where a two-step process is used:

1. Calculate the state update to minimize the unconstrained cost function $L(\mathbf{x})$
2. Project the updated state with a proximal operator to minimize for the regularization function $R(\mathbf{x})$

This can also be expressed as:

$$\mathbf{z_{k+1}} = \mathbf{x_k} + \mu_k \mathbf{p_k} \tag{2.33}$$
$$\mathbf{x_{k+1}} = P(\mathbf{z_{k+1}}), \tag{2.34}$$

where $P(\cdot)$ is a proximal operator. An example of a proximal operator corresponding to the previously discussed ℓ_1 regularization function is the soft thresholding operator:

$$P_\mu(z) = \begin{cases} z-\mu, & z \geq \mu \\ 0, & |z| < \mu \\ z+\mu, & z \leq -\mu \end{cases}, \tag{2.35}$$

which combined with the update scheme in Equation (2.33) results in the Iterative Soft Thresholding Algorithm (ISTA), which is an algorithm commonly used in signal processing [36]. If we apply this same method to the SIRT algorithm, we arrive at the following proximal SIRT algorithm:

$$\mathbf{z_{k+1}} = (I - \mu C G^T R G)\mathbf{x_k} - \mu C G^T R \mathbf{y} \tag{2.36}$$
$$\mathbf{x_{k+1}} = P(\mathbf{z_{k+1}}). \tag{2.37}$$

The proximal operator can be designed by hand, however this can be involved. An example of a physics-informed proximal operator is a max operation: we know that the emissivity can not be negative. So, the proximal operator then becomes: $P(\mathbf{z}) = \max(0, \mathbf{z})$. We can also try to parametrize the proximal operator as a neural network, which will be the basis for a model-informed machine-learning architecture. Interestingly, the term μ is now not only a term for stability, but can also be seen as a data-consistency term. If μ is large, a lot of the measurement data and the model is included. If μ approaches zero, no measurement data or model knowledge is used, and the update only consists of the (neural) proximal mapping.

2.4 A Bayesian Perspective to Tomographic Reconstruction

In the previous sections we have derived different methods for reconstructing the poloidal emissivity. The basis for these methods is the minimization of the squared error. However, we can better motivate this choice, and analyse what assumptions we make when using the least-squares objective. We start with the linear model defined in Equation (2.1):

$$\mathbf{y} = G\mathbf{x} + \mathbf{e}. \tag{2.38}$$

The noise term $\mathbf{e}$ is of high interest here, as the noise model determines our final optimization objective. Ideally, we would have that we model the following elements:

1. The measurement error of the sensor
2. The error because of a mismatch between the actual geometry and the (linearly)-fitted geometry matrix G
3. The error because of additional reflections in the tokamak that are currently not taken into account in the creation of the geometry matrix

We know that the error for the first two elements can as well be positive as negative. For the last item we have that the because of reflections, only more emission is measured and thus present in $\mathbf{y}$, thus the error here can only be positive. However, modelling the error using different distributions results in an expression that is unsuitable for a symbolic derivation. Instead, we make a major assumption that the error follows a zero-mean Gaussian distribution. Furthermore, we also assume independence between the random variables, such that the covariance matrix only has on its diagonal. So, we can model the probability of a certain camera image now as:

$$p(\mathbf{y}|G, \mathbf{x}) = \mathcal{N}(\mathbf{y}|G\mathbf{x}, \Sigma). \tag{2.39}$$

In our case, we would like to know the probability of the poloidal emissivity given the camera image $p(\mathbf{x}|G, \mathbf{y})$. This can be derived from Bayes' theorem as:

$$p(\mathbf{x}|G, \mathbf{y}) = \frac{p(\mathbf{y}|G, \mathbf{x})p(\mathbf{x}|G)}{p(\mathbf{y}|G)}, \tag{2.40}$$

where $p(\mathbf{x}|\mathbf{y})$ is the posterior, $p(\mathbf{y}|\mathbf{x})$ the likelihood, $p(\mathbf{x})$ the prior and $p(\mathbf{y})$ the evidence. For brevity, we have omitted the dependence on the G in notation. Now, we would like to optimize the posterior and find an estimate for the poloidal emissivity $\hat{\mathbf{x}}$ by calculating the Maximum A-Posteriori estimate (MAP):

$$\hat{\mathbf{x}} = \arg\max_{\mathbf{x}} p(\mathbf{x}|\mathbf{y}) \propto \arg\max_{\mathbf{x}} p(\mathbf{y}|\mathbf{x})p(\mathbf{x}), \tag{2.41}$$

where we can remove the evidence because it does not influence the maximum. By writing out the expression for the probability distribution $p(\mathbf{y}|\mathbf{x})$, we have that:

$$p(\mathbf{y}|\mathbf{x})p(\mathbf{x}) = \mathcal{N}(\mathbf{y}|G\mathbf{x}, \Sigma)p(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^d\,|\Sigma|}} \exp\left(-\frac{1}{2}(\mathbf{y}-G\mathbf{x})^T\Sigma^{-1}(\mathbf{y}-G\mathbf{x})\right)p(\mathbf{x}), \tag{2.42}$$

where d is the dimensionality of the distribution (size of $\mathbf{x}$). To simplify the optimization we can take the logarithm of this expression, reducing the product here to an (easier to work with) sum:

$$\log\left(p(\mathbf{y}|\mathbf{x})p(\mathbf{x})\right) = \log p(\mathbf{y}|\mathbf{x}) + \log p(\mathbf{x}) \tag{2.43}$$

$$= -\frac{d}{2}\log(2\pi) - \frac{1}{2}\log(|\Sigma|) - \frac{1}{2}(\mathbf{y}-G\mathbf{x})^T\Sigma^{-1}(\mathbf{y}-G\mathbf{x}) + \log p(\mathbf{x}). \tag{2.44}$$

Because we are optimizing for $\mathbf{x}$, we can omit the first two terms which are not dependent on $\mathbf{x}$. If we now multiply the function with -1 and minimize instead of maximize the probability density function we have that:

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \frac{1}{2}\underbrace{(\mathbf{y}-G\mathbf{x})^T\Sigma^{-1}(\mathbf{y}-G\mathbf{x})}_{\|\mathbf{y}-G\mathbf{x}\|^2_{\Sigma^{-1}}} - \underbrace{\log p(\mathbf{x})}_{-R(\mathbf{x})}. \tag{2.45}$$

We have recovered the previously discussed least-squares objective, however, now with a good understanding of what assumptions we make. Mind that the regularization function is represented by the prior on the data. One difference is that we now weigh the least-squares problem by the covariance matrix, so the problem reduces to the least-squares problem if we assume a variance of one for all elements. One final simplification step we can take is to assume an equal variance for all elements and no covariance between the elements:

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \frac{1}{2\sigma^2}\|\mathbf{y}-G\mathbf{x}\|_2^2 - \log p(\mathbf{x}). \tag{2.46}$$

3 Convolutional Neural Networks for Inverse Problems

In this chapter the theory needed for machine learning based tomographic reconstructions is discussed. First, a general introduction to deep learning and convolutional neural networks is presented. Next, we take a look at U-Net based architectures and how we can incorporate physical knowledge of the system into U-Net based architectures. Finally, we discuss the application of deep unfolded networks, in which a neural network is incorporated into the iterative algorithms discussed.

3.1 Convolutional Neural Networks

Before the wide-spread adoption of neural networks (and the computational capacity to train them), convolutions were already used in the field of image processing. In these computer vision applications, the kernel was often designed by hand[37, 38]. Examples of these kernels are ridge detection kernels, sharpening kernels, and gradient detection kernels[1][2]. The convolution is calculated by element-wise multiplication of the kernel and a portion of the image, and summing these multiplied values. The expression for a 2D convolution is then given by [3]:

$$O_{ij} = \sum_{s_0=0}^{k_0-1} \sum_{s_1=0}^{k_1-1} H_{s_0,s_1} I_{i+s_0,j+s_1}, \tag{3.1}$$

where $O \in \mathbb{R}^{(n_0-2)\times(n_1-2)}$ is the output, $I \in \mathbb{R}^{n_0\times n_1}$ is the input and $H \in \mathbb{R}^{k_0,k_1}$ the kernel. The design of kernels for specific applications is time-intensive, making the automatic learning of these kernels beneficial. Consecutive convolutions with learned kernels are called convolutional neural networks (CNN) and have been used with great success for time signal classification [39], image classification [40, 41] and many more applications.

The reason we use a convolutional neural network is that in the case of images of size 256 by 256, a conventional neural network would have $256^2 \cdot 256^2 \approx 4.3 \cdot 10^9$ learnable parameters in the weight matrix! By using convolutions, we impose a prior on the data that there is a local correlation between neighbouring data points. A learnable convolutional layer usually convolves the input, adds a bias and then passes it through a non-linear activation function f [42]:

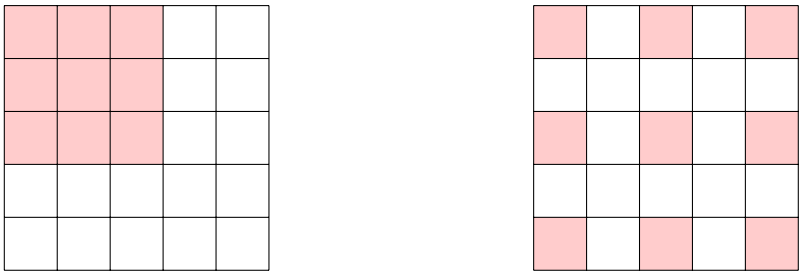
$$\mathbf{y} = f\left(b + \sum_{s_0=0}^{k_0-1} \sum_{s_1=0}^{k_1-1} H_{s_0,s_1} \mathbf{x}_{i+s_0,j+s_1}\right), \tag{3.2}$$

where a 2D input and output are assumed. This is equivalent to learning a weight matrix with only elements on the tridiagonal in a Toeplitz structure, with all other elements set to zero. More complexity can be added to the layer by increasing the number of learnable kernels H. If we denote the convolutional operator

[1] Take a look at this website for a great interactive visualization of different kernels: https://setosa.io/ev/image-kernels/

[2] For a more hands-on introduction using the Julia programming language, from the open course 'Introduction to Computational Thinking' by MIT, you can watch https://www.youtube.com/watch?v=8rrHTtUzyZA.

[3] This site also helped a lot in writing this section: https://deeplearningmath.org/convolutional-neural-networks.html



No dilation Dilation rate = 1

Figure 3.1: *The filled red squares in the grid represent the kernel, which scans the image (the grid) in a convolution. On the left the kernel is not dilated, while on the right the kernel is dilated with a dilation rate of 1. The dilated kernel has a bigger receptive field as it 'sees' more pixels at the cost of detecting high frequency data directly with the kernel.*

with filter H as $T_H(\mathbf{x})$, we can combine the different kernels as:

$$\mathbf{y_k} = f\left(b_k + \sum_{l}^{L} T_{H_k}(\mathbf{x_l})\right), \tag{3.3}$$

where k denotes the kernel index of the output $\mathbf{y}$ and L the number of kernels of the input $\mathbf{x}$.

These convolutional kernels have a limited receptive field, as this is determined by its size ($w \times h$). However, we would sometimes like to increase the receptive field of the network, such that the network also uses information not in the direct vicinity of a pixel. Several solutions exist to increase the receptive field: one could increase the size of the kernel, however this leads to more learnable parameters in the system thus increasing the learning time of the network. The next option is to dilate the convolutional kernel, which means that the convolution is now calculated as:

$$O_{ij} = \sum_{s_0=0}^{k_0-1} \sum_{s_1=0}^{k_1-1} H_{s_0,s_1} I_{i+d\cdot s_0, j+d\cdot s_1}. \tag{3.4}$$

Figure 3.1 clarifies this dilated convolution by showing a convolutional kernel and its output for two different dilation factors. The third option is chaining several convolutional layers with dimensionality reduction layers, which we take a look at next.

Pooling layers, which map a portion of the input into a single number, can be used to reduce the dimensionality of the input image $\in \mathbb{R}^{n_0 \times n_1 \times n_c}$, where n_c is the number of channels. An example is the max-pooling layer with a width w, where the input is grouped in $w \times w$ matrices, and then applying the max function to that sub matrix. Instead of using a pooling operation one could also increase the stride of the convolutional layers, meaning that rows/columns are skipped, and thus reducing the output size.

Even though the spatial resolution is decreased in convolutional neural networks, one generally wants to transform this spatial information into feature information. Thus, it is common to see an increase in the number of filters (kernels) as the spatial resolution of the image decreases.

In the classical application of CNNs: classification, the goal is to learn a reduced representation through consecutive convolutional layers, and then use this representation as an input in a classifier. This classifier (often a fully-connected neural network) then learns to maps the reduced representation to a class. So, this classifier has ten outputs if we have ten different classes. An illustration of a convolutional neural network is shown in Figure 3.2[4].

Encoder Decoder Networks

A convolutional neural network can be used to compress and classify data, as shown above, but can also be used to decompress or deconvolute data. An application of this is to transform 1D data into 2D images, which has been successfully done in the field of fusion on JET for bolometry data and for COMPASS for

[4]An interactive visualization of CNNs can be found here: https://poloclub.github.io/cnn-explainer/. There is also one for fully-connected neural networks: https://playground.tensorflow.org/

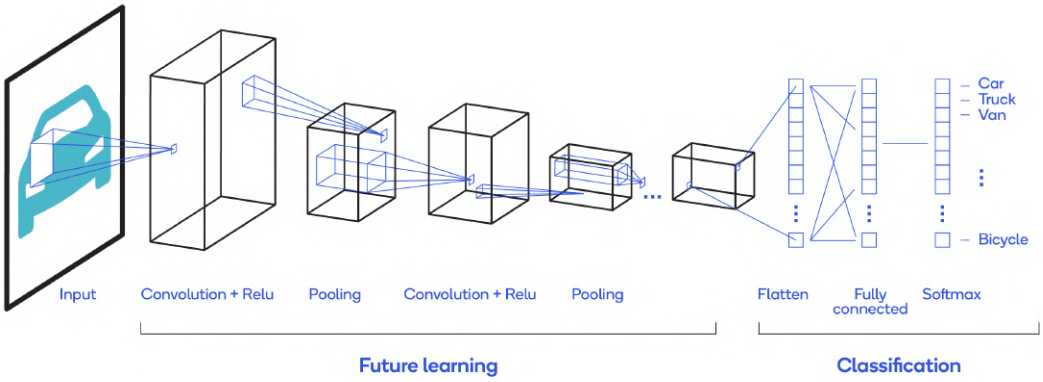


Figure 3.2: *An illustration of a convolutional neural network consisting of a feature learning backbone and a classification head. The feature learning backbone learns a reduced representation of the data. The classification head maps this representation to the different classes. Image copied from https://developer.qualcomm.com/software/qualcomm-neural-processing-sdk/learning-resources/cnn-architectures/deep-learning-convolutional-neural-networks-computer-vision.*

soft X-ray data [24]. This deconvolution is often implemented in two different ways, the first of which is using a transposed convolution operator, which maps a (smaller) input to a (larger) output by multiplying the kernel with each element of the input. The second approach is upsampling the input and applying a normal convolution to this image. It has been shown that former approach leads to high frequency noise artefacts in the output, showing up as chequerboard patterns [43], so the latter approach will be used in this project.

The compressing convolutional network and the decompressing deconvolutional network can be chained together, such that the output of the first is the input of the second network. This is called an encoder-decoder network (or auto encoder if both networks have the similar architecture). The connection layer between these two networks is often called the latent space Z, also referred to as the bottleneck, that contains a compressed representation of the system. This bottleneck can consist of multiple fully connected (FC) layers or convolutional layers. The benefits of using convolutional layers here are that fewer parameters are needed for the network, resulting in faster training, and a network that is agnostic to image size. The goal of the Encoder-Decoder architecture is to learn the networks $E_\theta : Y \to Z$ and $D_\theta : Z \to Y$ such that:

$$\mathbf{y} - (D_\theta \circ E_\theta)(\mathbf{y}) \tag{3.5}$$

is minimized. The input to the network can be identical to the output, but it is also possible that the input is altered and that the original image needs to be reconstructed, for example, in the case of noise removal.

One problem with the standard ED architecture is that the results heavily depend on the bottleneck size: a small (spatial and filter dimension) bottleneck allows for insufficient information flow, while a network with a too large (spatial dimension) size relies too much on the spatial correlation of the input data, both resulting in a suboptimal reconstruction. The latter is especially important in our problem setting, in which the input and output data are not spatially correlated in contrast to denoising EDs. Furthermore, some information might already be extracted to useful features after the first or second down-sampling step, instead of passing it through all the other encode and decode blocks.

A solution to this problem is to add residual connections between the encoder and decoder network, such that the filters for every encoder block are appended to the decoder block filters with the same spatial dimension. Such an architecture is called U-Net, which has also been used with great success for a variety of tasks like image segmentation, image denoising, image super resolution and tomographic reconstruction [44, 45].

In our problem setting we are not interested in reconstructing or segmenting the input to the network; we would like to perform a domain transformation $F : Y \to X$ consisting of an encoder $E_\theta : Y \to Z$ which

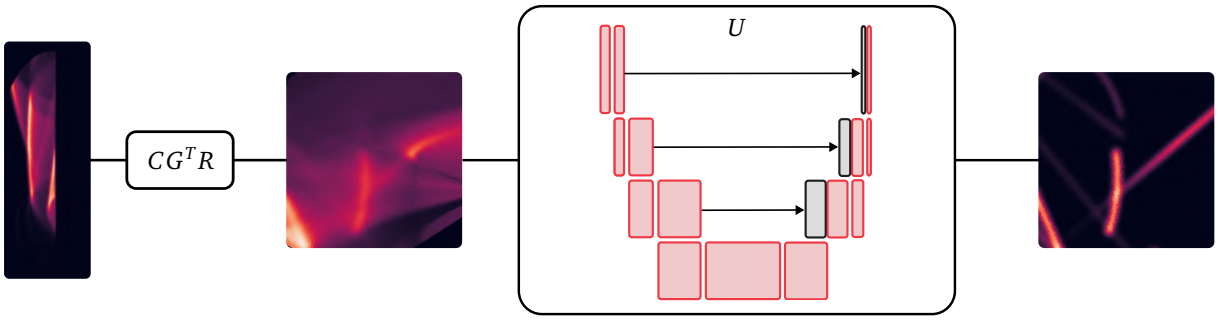


Figure 3.3: *The model-informed U-Net architecture, consisting of the SIRT-based back-projection which projects the camera image onto the poloidal space, followed by a typical U-Net. The block inside the U-Net denote convolutional layers. The black blocks and lines are the residual connections of the U-Net architecture, allowing for more information flow.*

maps the camera image information to a latent space Z of different dimensions because of the residual connections and using a decoder $D_\theta : Z \to X$ which reconstructs the poloidal emissivity from this latent space.

However, we have prior knowledge of the system available as the geometry matrix G. We can incorporate this knowledge into the network design by first back projecting the measured camera image to the poloidal space, and then post-processing the back projection using the U-Net architecture:

$$\hat{\mathbf{x}} = (U \circ CG^TR)(\mathbf{y}), \tag{3.6}$$

where $U : X \to X$ is a (U-Net) post-processing operator and where the SIRT-based back-projection is used. This solution can be seen as a learned regularized version of the direct pseudo-inverse estimation corresponding to the normal equations in Equation (2.21):

$$\hat{\mathbf{x}} = \overbrace{(CG^TRG)^{-1}}^{\approx U(\cdot)} CG^TR\mathbf{y}. \tag{3.7}$$

Figure 3.3 shows the model-informed U-Net architecture as described above.

3.2 Deep Unfolded Networks

We can also incorporate physical model knowledge into the network design by considering the proximal gradient descent algorithm in Equation (2.33). The purpose of the proximal operator P is to infuse prior knowledge of the problem into the iterative solver, e.g. sparsifying the solution or making sure that the solution is always positive. Designing these proximal operators is a task which requires substantial problem knowledge, so instead we can parameterize these by neural networks $P_{k,\theta}$ and learn them from the data. Mind that this is equivalent to learning the prior on the solution $\mathbf{x_k}$. To achieve this, the iterative loop described in Equation (2.36) is unfolded for K iterations, resulting in a directed computational graph, equivalent to a feedforward neural network:

$$\mathbf{z_{k+1}} = (I - \mu_k CG^TRG)\mathbf{x_k} - \mu_k CG^TR\mathbf{y} \tag{3.8a}$$

$$\mathbf{x_{k+1}} = P_{k,\theta}(\mathbf{z_{k+1}}). \tag{3.8b}$$

Mind that we use a different proximal operator in each unrolled step, so there are K learned proximal operators in total. Furthermore, the step size parameters μ_k are also learnable and differ for each unfold k. This allows the network to 'choose' how much of the measured data it includes into each step; the step size parameter μ is sometimes also referred to as the data consistency term. A U-Net architecture is used as the proximal operator in the unfolded network. However, this is a smaller U-Net than in the case of the post-processing U-Net, as precision is gained due to the unfolded iterations. The final architecture of the unfolded network is illustrated in Figure 3.4. The estimate $\hat{\mathbf{x}}_\mathbf{0}$ is initialized with all zeros. The unrolling of iterative loops and replacing some operators with neural networks is referred to as deep unfolding and has been used in the field of (medical) imaging and signal processing [46, 47, 48, 49, 50, 51, 52]. Interesting to note is that the informed U-Net architecture is a special case of the deep unfolded architecture, where $K = 1$. Because $\mathbf{x_0}$ is initialized at zero, we have that the $\mathbf{z_1}$ is given by the back projection of $\mathbf{y}$, which corresponds to the model informed step of the U-Net based architecture.

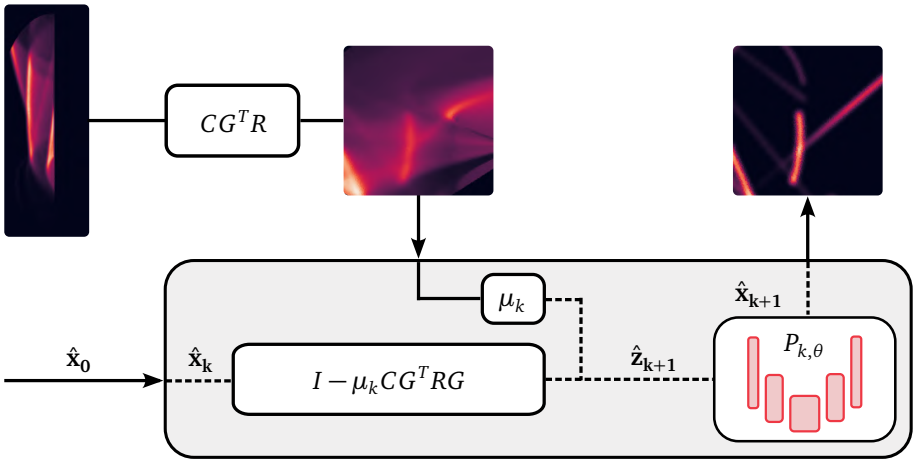


Figure 3.4: *The deep unfolded architecture. The initial estimate of the poloidal emissivity $\hat{\mathbf{x}}_{\mathbf{0}}$ is initialized with all zeros. The grey box represents the iterative part for K unfolds. The dashed lines represent data that are different for each iteration k. U-Net architectures are used as the proximal operators.*

4 Design and Setup of the Neural Network Training and Evaluation

In the previous chapter the two model-informed architectures were introduced. In this chapter we dive deeper into the hyperparameters of the network. Furthermore, we also look why an experimentally obtained dataset is not optimal, and how we can create a good synthetic dataset. Finally, we analyse the loss function used and motivate the evaluation criteria.

4.1 Synthetic Dataset Generation

Although existing data from the MANTIS camera system are available, creating a training dataset from these data is suboptimal. We would like to train data for a single geometry matrix, but different shots have different geometry matrices because of displacement and readjusting of the MANTIS cameras. Secondly and more importantly, the ground truth poloidal emissivity is not available for the experimentally obtained data, as this reconstruction is currently calculated using the SIRT algorithm. Using this inversion as training data would result in artefacts from the iterative algorithm showing up in the inversion estimated by the neural networks. Finally, as the MANTIS system most often measures a typical plasma scenario and shape, the network might overfit to these shapes, resulting in a network that cannot generalize to new scenarios or off-normal events.

Ideally, the dataset has similar characteristics to the experimentally obtained MANTIS data. The line emission is typically constrained to regions with specific ranges of plasma temperature, plasma density and neutral density, which particularly in the divertor region can have large gradients and magnetic-field-line induced anisotropy. Therefore, we impose the following features in the synthetic data:

1. Random lines of different lengths, thicknesses, positions, brightnesses and orientations, with lengths up to the order of the image size, but widths much smaller than the image size.
2. A random curvature is introduced to the lines, capturing the trends induced by the magnetic field geometry.
3. Random degrees of brightness gradient along the length of the lines, capturing the trends induced by the impact of parallel (to magnetic field) gradients of plasma temperature and density on line-emission.

Random line segments are generated based on these criteria and drawn on a poloidal plane. Furthermore, a Gaussian blur with a random kernel size is applied to the generated images. Finally, the drawn lines are forward projected to create a camera view image, and both images are saved as an input-label pair. The dataset used for training consists of 25 000 synthetically generated input-label pairs. A sample from the generated dataset is shown in Figure 4.1.

For evaluation, we would like to know how robust the networks are to noise, so images with noise are evaluated as well. We model the noise using two types of noise: the first one is constant noise, which represents the diffusive background reflections inside the tokamak:

$$\mathbf{y}_c = \mathbf{y} + c \cdot \max(\mathbf{y}), \tag{4.1}$$

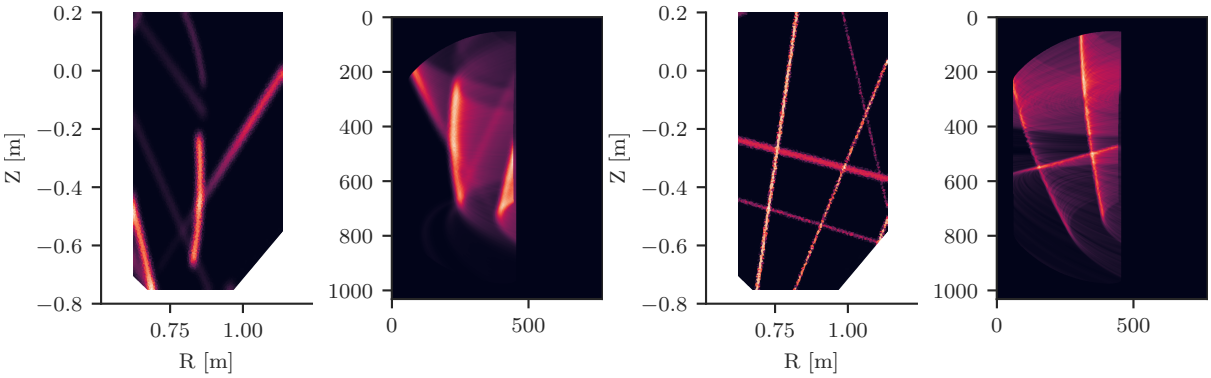


Figure 4.1: *Two examples from the synthetic dataset. The left image in each example shows a synthetic poloidal emissivity profile with the desired properties, such as lines with curvatures and gradients. The right image in each example shows the forward projection for the full camera sensor masked for the acquired region of interest and the field of view.*

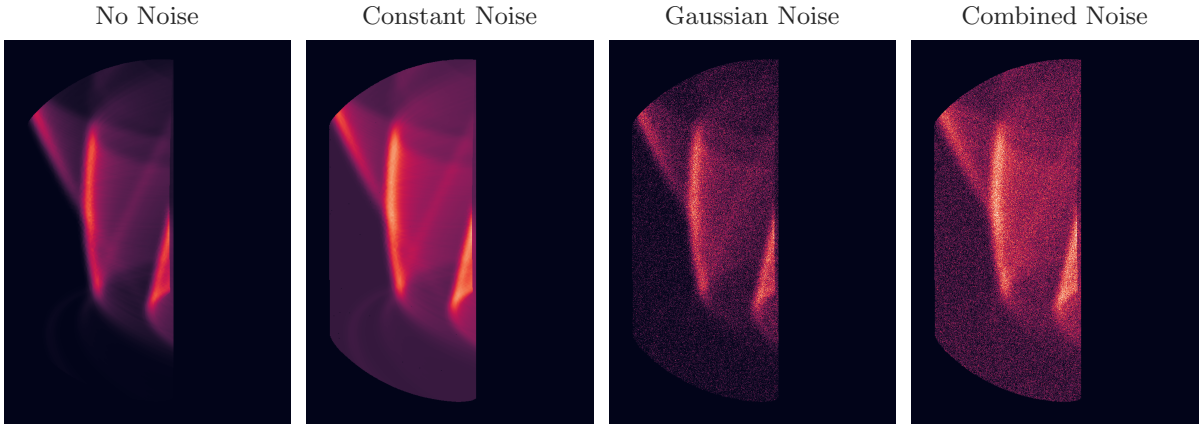


Figure 4.2: *The effect of the different noise functions on a synthetic camera image. A constant factor c of* 0.2 *is used, and a standard deviation σ of* 0.3 *is used.*

where c is the constant noise factor. The second noise type is Gaussian noise, which represents all other unmodelled elements and sensor noise:

$$\mathbf{y}_g = \mathcal{N}(\mathbf{y}, \sigma^2), \tag{4.2}$$

where σ is the standard deviation and $\mathcal{N}$ a normal distribution. Noise is only added to the inside of the tokamak, where light would be emitted. The two types of noise are visualized in Figure 4.2.

4.2 Network and Training Parameters

The U-Net architecture consists of $[64, 64, 64, 128, 128, 128]$ convolutional filters in the encoder followed by $[32, 32, 32, 16, 16, 16]$ convolutional filters in the decoder. The bottleneck consists of 64 filters. In each encoder block, the input is processed two times by a convolution layer, followed by an instance normalization layer, and a subsequent ReLU activation, and it is finally downsampled by a maximum pooling operator. In each decoder block we have upsampling, concatenation with the skip connection, convolution, instance normalization and afterwards a ReLU activation. The last three operations are performed twice.

For the deep unfolded network the loop is unfolded for $K = 4$ iterations and the parameterized proximal operator is a U-Net architecture with $[8, 8, 16, 32]$ filters for both the encoder and decoder, with a bottleneck of 32 filters. For each iteration there is a separate learned proximal operator, resulting in a total of five learned proximal operators.

Both networks are trained for a fixed number of epochs. The weights of the epoch with the lowest validation loss are used for evaluation and testing. An Adam optimizer is used with a learning rate of 10^{-3} and the

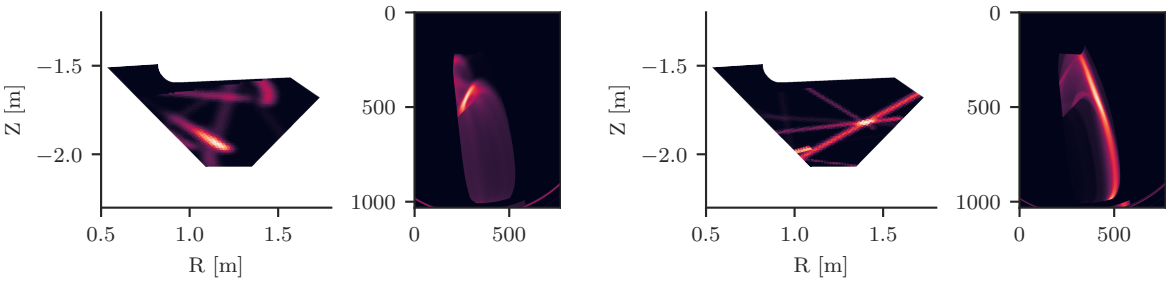


Figure 4.3: *Two synthetic data pair for the MAST-U tokamak. The left image in each pair shows a synthetic poloidal emissivity profile, the right image in each pair shows the forward projection. The toroidal direction in the camera image is vertical, compared to the horizontal toroidal direction in the case of TCV.*

mean squared error is used as the loss function. For the deep unfolded network, the loss is also calculated for each intermediate output, such that the total loss is calculated as the intermediate losses summed together [53, 54]:

$$L_{\text{unfolded}} = \sum_{k} \text{MSE}(\mathbf{x_{k+1}} = P_{k,\theta}(\mathbf{z_{k+1}}), \mathbf{x}). \tag{4.3}$$

The intermediate losses promote a decrease of the error after each unfold, similar to the non-neural network proximal gradient descent algorithm. Another benefit of using the loss at each unfolded step is that tuning the number of unfolds is now easier: the network can be trained for a higher number of unfolds. If there is no improvement after a certain amount of unfolds, we know that we can use that smaller amount of unfolds, reducing the complexity of the network.

Because of the field of view of the camera, and because of obstructing elements like baffles, it can be the case that the poloidal space is not completely observable. To account for this, the loss function is masked such that it only includes the observable poloidal space, i.e. the loss of the poloidal volume cells with a zero row in the geometry matrix are set to zero. The hyperparameters used in this study are determined by a combination of manual optimization and parameter sweeps. The networks have been trained with an NVIDIA V100 16 GB GPU on the MARCONI M100 supercomputer cluster. A single training takes 10 h to 24 h, depending on the initialization and the network architecture.

4.3 Performance Evaluation

The outputs of both networks are evaluated on both the synthetic dataset and on a dataset consisting of images experimentally obtained by MANTIS for plasma discharge #65903 on TCV. To show that the approach taken is also applicable to other machines, results for a synthetic sample of the MAST-U tokamak, with a different geometry matrix, are also shown [55]. The MAST-U sample is shown in Figure 4.3.

We evaluate the network performance on the mean squared error, the mean absolute error, and the inference time for a single sample on an NVIDIA TESLA V100 GPU.

In image generation tasks additional metrics are often used, which mainly focus on the shape and structure of the predicted image. An example of such a metric is the structural similarity index (SSIM). However, we care about the actual values produced by the network, as the output will be used for a Bayesian parameter inference model. Thus, SSIM and similar metrics are not used.

To analyse the sensitivity of the network, we use the Jacobian of the neural network [56]:

$$J_{ij}(\mathbf{y_h}) = \left.\frac{\mathrm{d}\hat{\mathbf{x}}_\mathbf{i}}{\mathrm{d}\mathbf{y}_\mathbf{i}}\right|_{\mathbf{y_h}}, \tag{4.4}$$

where $\hat{\mathbf{x}}_\mathbf{i}$ is estimated using the neural networks. The Jacobians are calculated for ten different synthetic samples, and the absolute average Jacobian is calculated to express the sensitivity. Using this sensitivity, we can approximate which parts of the image are more sensitive to changes in the input, and thus also to noise.

An implementation of the described networks and the synthetic dataset generation can be found in the following replication package: https://github.com/phaseolud/mantis-ml-inversion-replication.

5 Machine Learning Accelerated Tomographic Reconstruction for Multispectral Imaging on TCV

The contents of this chapter will be submitted as a paper to Nuclear Fusion, and therefore uses a two-column layout. The paper retells the story from the preceding chapters in a condensed format.

Abstract

A multispectral camera setup is used to infer a 2D map of plasma parameters in a tokamak from spectral emissions. However, the light measured by these cameras is line integrated in the toroidal direction, whereas emissivities on the poloidal plane are necessary for the inference. The poloidal plasma emissivity can be obtained by tomographic reconstruction, but classical techniques are too slow to use these emissivities for real-time control. We present two machine-learning based approaches to accelerate the reconstruction of the poloidal emissivities. Both approaches yield more accurate results on synthetic data than the iterative approach while being near fast enough for real-time control applications.

5.1 Introduction

In future fusion reactors such as DEMO and ITER, the alpha particles produced in the fusion reaction will play an important role in the heating of the plasma. However, these helium particles are not desired in the plasma after their thermalization, as they dilute the relative fuel density. In order to improve the avoidance of helium ash build-up the ashes should be pumped away. Furthermore, the plasma-wall interaction results in impurities sputtering from the wall, which are undesirable in the plasma. Finally, the power from the alpha particles exhausted from the reactor should be dispersed. For these three reasons, a specific magnetic 'divertor' geometry was designed that better isolates the plasma exhaust region from the main core plasma [2, 3].

To avoid damage to the divertor wall materials, the heat flux and particle flux of the plasma in this region should be minimized [2]. By careful injection of neutral particles such as nitrogen or hydrogen, one can control these fluxes towards the target [4]. In the case of hydrogen neutrals injection, i.e. fuelling, the plasma transfers energy and momentum to the neutral particles [5]. In the case of impurity enrichment, e.g. nitrogen seeding, the energy loss is mainly caused by an increase in radiation power [6]. The atomic and molecular reactions causing a loss of energy and momentum result in a colder and denser plasma near the divertor target. The state with a pressure gradient along the field lines to the target in combination with a low pressure at the target is referred to as detachment [7], which should be controlled for optimal performance and material durability.

The aim of detachment control is to prevent the exceedance of the material threshold by the heat and particle fluxes towards the target, thus preventing damage to the divertor tiles [8]. However, the plasma should not be cooled too much, as this can result in the detachment front moving towards the plasma core [9], where the detachment front corresponds to a temperature of approximately 5 eV [57].

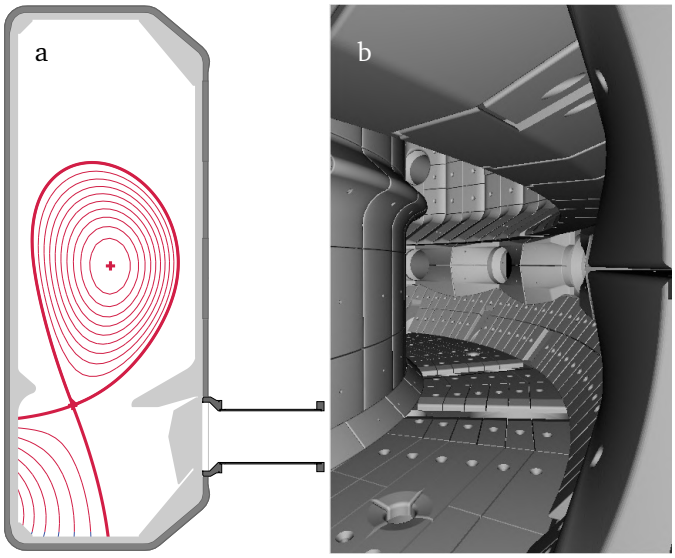


Figure 5.1: *(a) A poloidal cross-section of the TCV tokamak, with the magnetic field lines indicated in red. The outer thick red line indicates the scrape-off-layer (SOL), which diverts the particles towards the targets. (b) The view of the MANTIS camera in the tokamak is not perpendicular to the poloidal plane, but at an angle, measuring line-integrated emissions. Adapted from [5].*

If the front comes too close to the x-point, the temperature drop of the plasma closer to the core results in degradation of the plasma confinement and performance [9].

Both local measurements and their incorporation in a reconstruction of the divertor plasma state are necessary for control of the fluxes in the divertor region. In this work, we focus on the fast tomographic reconstruction of the emissivity in the divertor region for the tokamak à configuration variable (TCV), at the Swiss Plasma Centre in Lausanne. Various diagnostics systems, such as bolometry [10], Langmuir probes [11], and multispectral imaging [12, 13] are used in the TCV to reconstruct the plasma state. The latter is real-time capable, non-invasive, suitable for 2D imaging, and it has been achieved on TCV using the Multispectral Advanced Narrowband Tokamak Imaging System (MANTIS). Figure 5.1 illustrates the location of the MANTIS camera along with the rendered view of what the MANTIS camera sees. Feedback control of the detachment front has already been used on TCV using this camera system [14].

Different species in the plasma emit light at different wavelengths. By measuring this spectral and spatial information, a 2D reconstruction of the plasma state, temperature and densities, in the divertor region can be made. The inference of the plasma temperature and densities, as well as atomic process rates is based on collisional-radiative modelling [16]. These 2D maps allow for advanced control schemes through the determination of particle sources and sinks. For this reconstruction the poloidal emissivities are needed whereas MANTIS measures line integrated emissiv-

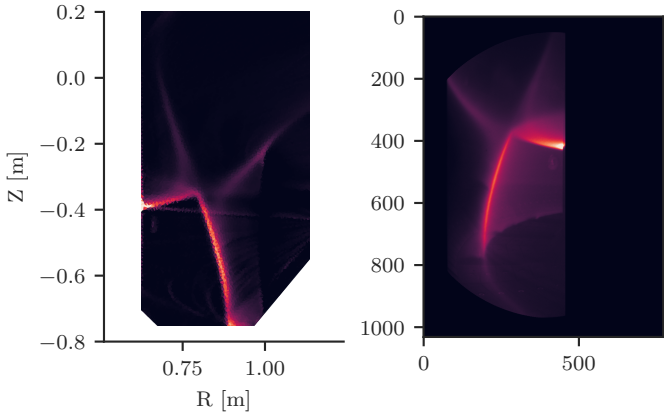


Figure 5.2: *An image measured with MANTIS on the right: the divertor geometry diverts the plasma to the targets through two legs. The measured light comes from the Deuterium Balmer line emission $D_{3\to2}$. The reconstruction of the poloidal emissivity on the left is calculated using the SIRT algorithm [33, 32].*

ities, meaning that light from different (R, Z) coordinates is integrated. An example of the measured camera image together with the corresponding poloidal emissivity is shown in Figure 5.2. For plasma state reconstruction one needs the poloidal emissivity, rather than line-integrated measurements. The inversion process from the measured camera image to the poloidal plane is computationally intensive, making it unsuitable for real-time control. The plasma dynamics have a temporal scale of approximately 15 ms. To prevent sampling issues, we need to sample at least two times faster than the temporal scale, and in practice a factor of eight to ten is used. This means that the inversion calculation should take less than 2 ms [14], which is difficult to achieve using traditional iterative techniques.

Previously, the reconstruction of the poloidal emissivity took approximately 5 s per frame using the method described in [13]. This results in a processing time of more than 1 h per shot, which hinders fast scientific iteration. So, accelerating the tomographic reconstructions does not only allow for real-time processing, but also provides the opportunity to have a faster offline data processing pipeline.

Tomographic inversions are also needed in other fields of science dealing with imaging, such as geophysical and medical imaging. Research in these fields has shown that neural networks can be used to approximate the tomographic inversion while being computationally more efficient and can even yield more accurate results [22, 23]. In this study we research how we can achieve a fast poloidal emissivity reconstruction in the divertor region of TCV, using model-informed machine learning techniques.

In Section 5.2 we introduce and review the imaging problem and discuss the existing iterative techniques which have been used to solve the tomographic reconstruction problem. We build upon this knowledge in the design of the neural networks in Section 5.3, and subsequently, dive into synthetic data generation and the experimental setup in Section 5.4. The results are presented and discussed in Section 5.5. This research is concluded in Section 5.6, based on which ideas for future work are introduced in Section 5.7.

5.2 Iterative Reconstruction Techniques

The mapping from the poloidal plane to the camera image can be modelled as a linear forward map $G : \mathbb{R}^{n_{\text{inv}}} \to \mathbb{R}^{n_c \cdot m_c}$ represented by the matrix G:

$$\mathbf{y} = G\mathbf{x} + \mathbf{e}. \tag{5.1}$$

n_c and m_c are the number of rows and columns of the acquired image, n_{inv} is the number of inversion cells, $\mathbf{y}$ the flattened measured camera image, $\mathbf{x}$ the poloidal emissivity and $\mathbf{e}$ a noise term to account for measurement noise, but also for unmodelled effects like reflections inside the tokamak. The matrix G is referred to as the geometry matrix and is constructed by point-fitting and ray-tracing locations in an image measured with MANTIS to a CAD model of the TCV tokamak [13] using the Calcam software package [25]. We assume both toroidal symmetry and that the data for a camera pixel only comes from a single line of sight. The single lines of sight for each camera image pixel, combined with not modelling the reflections, results in a sparse geometry matrix G. The carbon tiles inside TCV reflect some light diffusively, but with a lower magnitude than the direct emission from the plasma. This allows for the approximation of not modelling the reflections.

The goal of the reconstruction is to find the poloidal emissivity given a measured camera image; the aim is to maximize the probability density function $p(\mathbf{x}|\mathbf{y}, G)$. We express this maximization as:

$$\hat{\mathbf{x}} = \arg\max_{\mathbf{x}} p(\mathbf{x}|\mathbf{y}) = \arg\max_{\mathbf{x}} \frac{p(\mathbf{y}|\mathbf{x}, G)p(\mathbf{x}|G)}{p(\mathbf{y}|G)}, \tag{5.2}$$

which is called the maximum a posteriori (MAP) estimate. In this expression $p(\mathbf{y}|\mathbf{x}, G)$ is referred to as the likelihood, $p(\mathbf{x}|G)$ as the prior, and $p(\mathbf{y}|G)$ as the evidence. As the maximum does not depend on the evidence, we can omit this last term from the optimization. We drop the geometry matrix G from the notation for brevity.

If we assume a zero-mean Gaussian noise distribution for the likelihood, with covariance matrix Σ, and optimize for the log-transformed distributions we obtain:

$$\begin{aligned} &\arg\max_{\mathbf{x}} \log p(\mathbf{x}|\mathbf{y}) = \\ &\arg\max_{\mathbf{x}} \left(-\frac{1}{2}(\mathbf{y} - G\mathbf{x})^T \Sigma^{-1} (\mathbf{y} - G\mathbf{x}) + \log p(\mathbf{x}) \right), \end{aligned} \tag{5.3}$$

where we dropped the terms of the log-likelihood which do not depend on $\mathbf{x}$. If we now rewrite the maximization

problem to a minimization problem and assume independent and equal variances σ^2, we find the regularized least-squares objective:

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \frac{1}{2\sigma^2} \overbrace{(\mathbf{y}-G\mathbf{x})^T(\mathbf{y}-G\mathbf{x})}^{C(\mathbf{x})=\|G\mathbf{x}-\mathbf{y}\|_2^2} - \overbrace{\log p(\mathbf{x})}^{R(\mathbf{x})}, \tag{5.4}$$

where $R(\mathbf{x})$ is a regularization function, and $C(\mathbf{x})$ is the unregularized cost function.

To utilize the sparsity of the geometry matrix, iterative methods are most often used in the field of image reconstruction, in which large system matrices are involved. This way it is not required to compute an explicit approximation of (pseudo-) inverse. In iterative methods, we iteratively update our estimate to approach a stationary point and thus the solution to the inverse problem. In this study, we will focus on line-search methods, in which the current estimate $\mathbf{x_k}$ is updated with a search direction $\mathbf{p_k}$ which is scaled by the step size μ_k [26]:

$$\mathbf{x_{k+1}} = \mathbf{x_k} + \mu_k \mathbf{p_k}. \tag{5.5}$$

If we use the gradient of the unregularized objective function in Equation (5.4) as the search direction i.e. steepest descent direction, the resulting iterative scheme is given by:

$$\mathbf{x_{k+1}} = \mathbf{x_k} - \mu \nabla f(\mathbf{x_k}) = \mathbf{x_k} - \mu G^T(G\mathbf{x_k} - \mathbf{y}), \tag{5.6}$$

which is known as the gradient descent algorithm, and as the basis for many optimization algorithms. Finding a proper value for the step size, also known as the learning rate, is of importance: a too small step size will cause almost no change in $\mathbf{x_k}$, whereas a too large step size causes the solution to diverge. Methods such as exact line search or Armijo line search could be used to find a good value for μ [26]. However, we will not focus on these methods in this paper.

Instead, we use the ART family of methods, which has been specifically developed for CT imaging. Specifically, we will use the simultaneous and iterative reconstruction technique (SIRT). This algorithm and its variants (SART/ART) have been used extensively in the field of medical and geophysical imaging [27, 28, 29, 30, 31].

Equation (5.6) can also be understood as follows: the term $G\mathbf{x_k}$ maps the current inversion estimate to the measurement space. Subsequently, the error in measurement space is calculated $G\mathbf{x_k} - \mathbf{y}$, which is then mapped back to inversion space $G^T(\cdot)$. As previously mentioned, finding a good value for the step size is an iterative process, and the convergence speed and stability of the iterative algorithm depend on this choice. In the SIRT algorithm, the step size problem is solved by preconditioning the geometry matrix to ensure a stable convergence. First, the error correction term $G\mathbf{x_k} - \mathbf{y}$ is multiplied with the matrix R, which is a diagonal matrix containing the inverse row sums of G, so $r_{ii} = 1/\sum_j g_{ij}$. This is equivalent to solving the weighted least-squares problem [32, 33, 34]. The R matrix reduces the error contribution of pixels in measurement space that correspond with a longer ray length. Without this R, the steepest descent direction would be biased towards minimizing the error for pixels that receive the most contribution from the reconstruction space.

The gradient of the weighted least-squares problem is given by:

$$\nabla C_R(\mathbf{x}) = G^T R(G\mathbf{x} - \mathbf{y}). \tag{5.7}$$

If we precondition this gradient by the matrix C we obtain:

$$\nabla C_R^*(\mathbf{x}) = CG^T R(G\mathbf{x} - \mathbf{y}), \tag{5.8}$$

where C is a diagonal matrix containing the inverse column sums of G: $c_{jj} = 1/\sum_i g_{ij}$. The C matrix plays the same role for G^T as R does for G. Rewriting Equation (5.8) to an iterative update scheme results in:

$$\mathbf{x_{k+1}} = \mathbf{x_k} - \mu CG^T R(G\mathbf{x_k} - \mathbf{y}), \tag{5.9}$$

This can then be rearranged to:

$$\mathbf{x_{k+1}} = (I - \mu CG^T RG)\mathbf{x_k} + \mu CG^T R\mathbf{y}, \tag{5.10}$$

which is stable for $0 < \mu < 2$ [32, 33]. The proof by [32, 33] is rewritten, and shown in Appendix 5.A.

Another benefit of the SIRT algorithm is that the back projection ($CG^T R\mathbf{y}$) results in more structural information than the gradient descent-based back projection ($G^T\mathbf{y}$). The convolutional neural networks used for the machine learning acceleration utilize this local structural information, making the SIRT-based back projection better suited for incorporation in convolutional neural networks. To clarify this structural information, the back projection for both the SIRT and the gradient descent algorithm for a synthetic sample are shown in Figure 5.3.

The described SIRT algorithm is an effective way to solve the likelihood element of the Bayesian optimization. However, in the formulation we also introduced the prior distribution $p(\mathbf{x})$. This prior distribution corresponds to the regularization function $R(\mathbf{x})$ in the objective minimization function. In the case that the regularization function is differentiable, we easily integrate it into the line search method. However, not all regularization functions are differentiable, like the ℓ_1 regularization function, which is often used in signal processing if it is known that the solution is sparse. Instead, we can use the proximal gradient descent method [35], in which the objective is optimized in a two-step process:

1. Calculate the estimate update to optimize for the unregularized cost function $C(\mathbf{x})$.
2. Project the updated state with a proximal operator to optimize for the regularization function $R(\mathbf{x})$.

This can also be expressed as:

$$\mathbf{z_{k+1}} = \mathbf{x_k} + \mu_k \mathbf{p_k} \tag{5.11a}$$

$$\mathbf{x_{k+1}} = P(\mathbf{z_{k+1}}), \tag{5.11b}$$

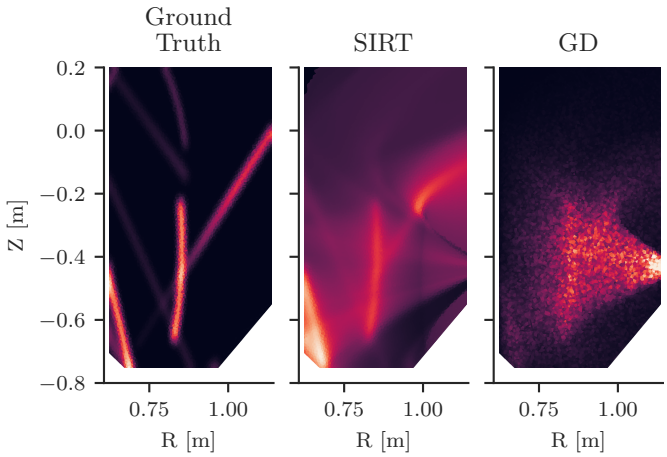


Figure 5.3: *A comparison of the SIRT-based back-projection and the gradient descent-based back-projection. The left panel shows a poloidal emissivity* $\mathbf{x}$ *with a corresponding synthetic camera image* $\mathbf{y} = G\mathbf{x}$*. The middle panel shows the SIRT-based back-projection* $CG^TR\mathbf{y}$*, and the right panel shows the gradient descent-based back-projection* $G^T\mathbf{y}$*. The colour axes are not standardized, as we would like to emphasize the structure and not the specific values.*

where $P(\cdot)$ is a proximal operator. An example of a proximal operator corresponding to the previously discussed ℓ_1 regularization function is the soft thresholding operator:

$$P_\mu(z) = \begin{cases} z-\mu, & z \geq \mu \\ 0, & |z| < \mu \\ z+\mu, & z \leq -\mu \end{cases}, \tag{5.12}$$

which combined with the update scheme in Equation (5.11a) results in the Iterative Soft Thresholding Algorithm (ISTA), which is an algorithm commonly used in signal processing [36]. If we apply this same method to the SIRT algorithm, we arrive at the following proximal SIRT algorithm:

$$\mathbf{z_{k+1}} = (I - \mu CG^TRG)\mathbf{x_k} - \mu CG^TR\mathbf{y} \tag{5.13a}$$
$$\mathbf{x_{k+1}} = P(\mathbf{z_{k+1}}). \tag{5.13b}$$

The proximal operator is a way to incorporate regularization information into the iterative scheme. Even more interesting is the fact that equivalently this operator projects the current estimate $\mathbf{z_{k+1}}$ according to the prior $p(\mathbf{x})$. The proximal operator or prior can be designed by hand from physical processes, but this can be a time-intensive and involved task. Instead, we can parameterize the proximal operator as a neural network, which will be the basis for Section 5.3.

5.3 Machine Learning Accelerated Reconstruction

Neural networks have been shown to excel at solving a range of tasks in regression and classification, and have already been applied to the field of medical and seismic imaging [50, 44, 23] and (linear) inverse problems [58, 59]. This success in similar domains give rise to an interest in the applications of these techniques to our setting, with the goal of reducing the reconstruction time. We discuss two types of network architectures: one based on the U-Net architecture, and another based on the iterative update scheme discussed in the previous section.

5.3.1 Nonlinear Filtering with U-Net

In the field of computer vision and image-based data, most often convolutional neural networks (CNN) are used, which are networks which consist of several parallel and sequential convolutions, where all the convolutional kernels are learned [42]. In image classification tasks, features are extracted using convolutional filters and pooling operators until the image can be represented in some (small) latent space. This part of a CNN is often called the encoder. The output of the encoder is fed into a classification head, which can be used to classify the images. However, we can also deconvolve the latent space back to an image, in which case the network is referred to as a decoder. If we combine both networks, we have an Encoder-Decoder (ED) network, which can represent an image in some lower dimensional latent space Z also referred to as the bottleneck, and then reconstruct it back to the image space [42, 60]. The goal of the Encoder-Decoder architecture is to learn the networks $E_\theta : Y \to Z$ and $D_\theta : Z \to Y$ such that:

$$\mathbf{y} - (D_\theta \circ E_\theta)(\mathbf{y}) \tag{5.14}$$

is minimized. The input to the network can be identical to the output, but it is also possible that the input is altered and that the original image needs to be reconstructed, for example, in the case of noise removal. More background information on CNNs is presented in Appendix 5.B.

In our problem setting we are not interested in denoising or segmenting the input to the network; we would like to perform a domain transformation $F : Y \to X$ consisting of an encoder $E_\theta : Y \to Z$ which maps the camera image information to a latent space Z of different dimensions and using a decoder $D_\theta : Z \to X$ which reconstructs the poloidal emissivity from this latent space. The network would then be trained on image-pairs consisting of synthetic poloidal emissivities with their corresponding camera images.

One problem with the standard ED architecture is that the results heavily depend on the bottleneck size: a small (spatial and filter dimension) bottleneck allows for insufficient information flow, while a network with a too large (spatial dimension) size relies too much on the spatial correlation of the input data, both resulting in a suboptimal reconstruction. The latter is especially important in our problem setting, in which the input and output data are not spatially correlated in contrast to denoising EDs. Furthermore, some information might already be extracted to useful features

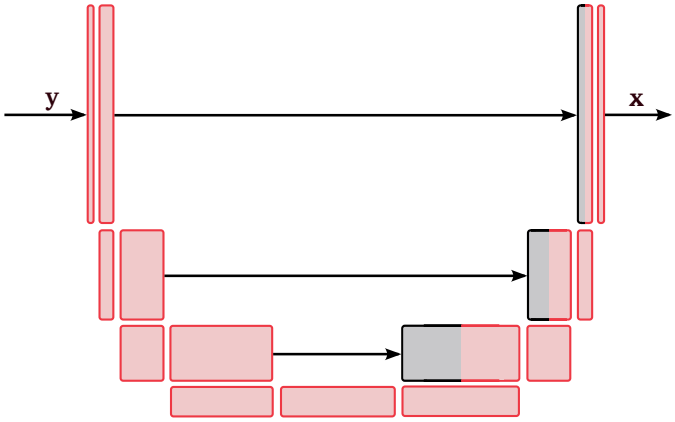


Figure 5.4: *The U-Net architecture. The coloured blocks are convolutional layers, with the height representing the spatial size, and the width the number of filters. The black arrows represent the residual connections, which concatenate the filters of the encoder to the filters of the decoder. This allows for information flow at different levels of spatial compression.*

after the first or second down-sampling step, instead of passing it through all the other encode and decode blocks.

A solution to this problem is to add residual connections between the encoder and decoder network, such that the filters for every encoder block are appended to the decoder block filters with the same spatial dimension. Such an architecture is called U-Net [61], which has also been used with great success for a variety of tasks like image segmentation, image denoising, image super resolution and tomographic reconstruction [44, 45]. An illustration of the U-Net architecture is shown in Figure 5.4. In this figure we can see there is a latent space at different spatial dimensions. A smaller spatial dimensional corresponds with a higher number of filters. Even tough we could train a U-Net on our data, we have prior knowledge of the system available as the geometry matrix G. So, it can be useful to include this knowledge into the neural network design. The incorporation of linear (potentially fitted) model knowledge has been used with great success before in the field of state-space identification methods [62], and in physics-driven systems where the network is then referred to as a physics informed neural network [58]. We incorporate this knowledge into the network design by first back projecting the measured camera image to the poloidal space, and then post-processing the back projection using the U-Net architecture:

$$\hat{\mathbf{x}} = (U \circ CG^T R)(\mathbf{y}), \tag{5.15}$$

where $U : X \to X$ is a (U-Net) post-processing operator. This solution can be seen as a learned regularized version of the direct pseudo-inverse estimation:

$$\hat{\mathbf{x}} = \overbrace{(CG^T RG)^{-1}}^{\approx U(\cdot)} CG^T R\mathbf{y}. \tag{5.16}$$

Figure 5.5 shows the model-informed U-Net architecture

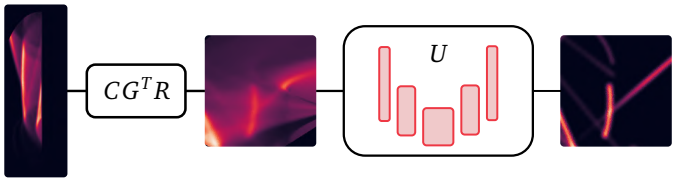


Figure 5.5: *The model-informed U-Net architecture, consisting of the SIRT-based back-projection which projects the camera image onto the poloidal space, followed by a typical U-Net.*

as described above. In Appendix 5.C we analyse the effect of adding the model-based back-projection to the U-Net architecture on the model performance.

5.3.2 Deep Unfolded Networks

We can also incorporate physical model knowledge into the network design by considering the proximal gradient descent algorithm in Equation (5.11a). The purpose of the proximal operator P is to infuse prior knowledge of the problem into the iterative solver, e.g. sparsifying the solution or making sure that the solution is always positive. Designing these proximal operators is a task which requires substantial problem knowledge. Instead, we can parameterize these by neural networks $P_{k,\theta}$ and learn them from the data. Mind that this is equivalent to learning the prior on the solution $\mathbf{x_k}$. To achieve this, the iterative loop described in Equation (5.13a) is unfolded for K iterations, resulting in a directed computational graph, equivalent to a feedforward neural network:

$$\mathbf{z_{k+1}} = (I - \mu_k CG^T RG)\mathbf{x_k} - \mu_k CG^T R\mathbf{y} \tag{5.17a}$$
$$\mathbf{x_{k+1}} = P_{k,\theta}(\mathbf{z_{k+1}}). \tag{5.17b}$$

Mind that we use a different proximal operator in each unrolled step, so there are K learned proximal operators in total. Furthermore, the step size parameters μ_k are also learnable and differ for each unfold k. This allows the network to 'choose' how much of the measured data it includes into each step; the step size parameter μ is sometimes also referred to as the data consistency term. A U-Net architecture is used as the proximal operator in the unfolded network. However, this is a smaller U-Net than in the case of the post-processing U-Net, as precision is gained due to the unfolded iterations. The final architecture of the unfolded network is illustrated in Figure 5.6. The estimate $\hat{\mathbf{x}}_\mathbf{0}$ is initialized with all zeros. The unrolling of iterative loops and replacing some operators with neural networks is referred to as deep unfolding and has been used in the field of (medical) imaging and signal processing [46, 47, 48, 49, 50, 51, 52]. Interesting to note is that the informed U-Net architecture is a special case of the deep unfolded architecture, where $K = 1$. Because $\mathbf{x_0}$ is initialized at zero, we have that the $\mathbf{z_1}$ is given by the back projection of $\mathbf{y}$, which corresponds to the model informed step of the U-Net based architecture.

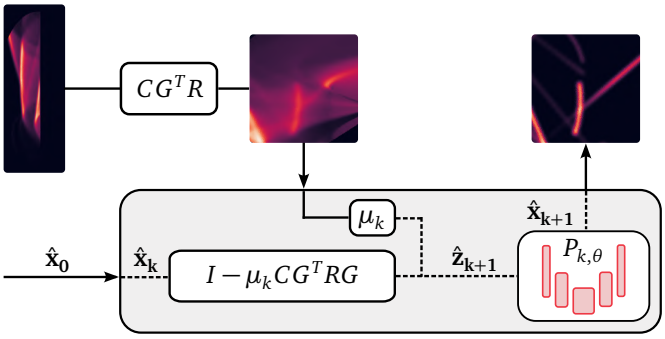


Figure 5.6: *The deep unfolded architecture. The initial estimate of the poloidal emissivity $\hat{\mathbf{x}}_0$ is initialized with all zeros. The grey box represents the iterative part for K unfolds. The dashed lines represent data that are different for each iteration k.*

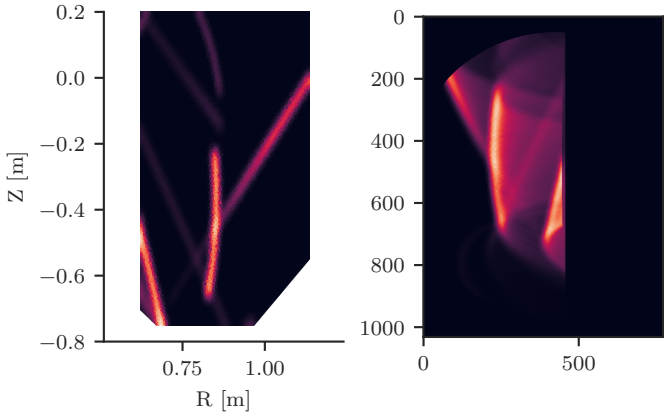


Figure 5.7: *An example from the synthetic dataset. The left image shows a synthetic poloidal emissivity profile with the desired properties, such as lines with curvatures and gradients. The right image shows the forward projection for the full camera sensor masked for the acquired region of interest and the field of view.*

5.4 Experimental Setup

5.4.1 Synthetic Dataset

Although existing data from the MANTIS camera system are available, creating a training dataset from these data is suboptimal. We would like to train data for a single geometry matrix, but different shots have different geometry matrices because of displacement and readjusting of the MANTIS cameras. Secondly and more importantly, the ground truth poloidal emissivity is not available for the experimentally obtained data, as this reconstruction is currently calculated using the SIRT algorithm. Using this inversion as training data would result in artefacts from the iterative algorithm showing up in the inversion estimated by the neural networks. Finally, as the MANTIS system most often measures a typical plasma scenario and shape, the network might overfit to these shapes, resulting in a network that cannot generalize to new scenarios or off-normal events.

Ideally, the dataset has similar characteristics to the experimentally obtained MANTIS data. The line emission is typically constrained to regions with specific ranges of plasma temperature, plasma density and neutral density, which particularly in the divertor region can have large gradients and magnetic-field-line induced anisotropy. Therefore, we impose the following features in the synthetic data:

1. Random lines of different lengths, thicknesses, positions, brightnesses and orientations, with lengths up to the order of the image size, but widths much smaller than the image size.
2. A random curvature is introduced to the lines, capturing the trends induced by the magnetic field geometry.
3. Random degrees of brightness gradient along the length of the lines, capturing the trends induced by the impact of parallel (to magnetic field) gradients of plasma temperature and density on line-emission.

Random line segments are generated based on these criteria and drawn on a poloidal plane. Furthermore, a Gaussian blur with a random kernel size is applied to the generated images. Finally, the drawn lines are forward projected to create a camera view image, and both images are saved as an input-label pair. The dataset used for training consists of 25000 synthetically generated input-label pairs. A sample from the generated dataset is shown in Figure 5.7.

For evaluation, we would like to know how robust the networks are to noise, so images with noise are evaluated as well. We model the noise using two types of noise: the first one is constant noise, which represents the diffusive background reflections inside the tokamak:

$$\mathbf{y}_c = \mathbf{y} + c \cdot \max(\mathbf{y}), \tag{5.18}$$

where c is the constant noise factor. The second noise type is Gaussian noise, which represents all other unmodelled elements and sensor noise:

$$\mathbf{y}_g = \mathcal{N}(\mathbf{y}, \sigma^2), \tag{5.19}$$

where σ is the standard deviation and $\mathcal{N}$ a normal distribution.

5.4.2 Network and Training Parameters

The U-Net architecture consists of [64, 64, 64, 128, 128, 128] convolutional filters in the encoder followed by [32, 32, 32, 16, 16, 16] convolutional filters in the decoder. The bottleneck consists of 64 filters. In each encoder block, the input is processed two times by a convolution layer, followed by an instance normalization layer, and a subsequent ReLU activation, and it is finally downsampled by a maximum pooling operator. In each decoder block we have upsampling, concatenation with the skip connection, convolution, instance normalization and afterwards a ReLU activation. The last three operations are performed twice.

For the deep unfolded network the loop is unfolded for $K = 5$ iterations and the parameterized proximal operator is a U-Net architecture with $[8, 8, 16, 32]$ filters for both the encoder and decoder, with a bottleneck of 32 filters. For each iteration there is a separate learned proximal operator, resulting in a total of five learned proximal operators.

Both networks are trained for a fixed number of epochs. The weights of the epoch with the lowest validation loss are used for evaluation and testing. An Adam optimizer is used with a learning rate of 10^{-3} and the mean squared error is used as the loss function. For the deep unfolded network, the loss is also calculated for each intermediate output, such that the total loss is calculated as the intermediate losses summed together:

$$L_{\text{unfolded}} = \sum_k \text{MSE}(\mathbf{x_{k+1}} = P_{k,\theta}(\mathbf{z_{k+1}}), \mathbf{x}). \tag{5.20}$$

Because of the field of view of the camera, and because of obstructing elements like baffles, it can be the case that the poloidal space is not completely observable. To account for this, the loss function is masked such that it only includes the observable poloidal space, i.e. the loss of the poloidal volume cells with a zero row in the geometry matrix are set to zero. The hyperparameters used in this study are determined by a combination of manual optimization and parameter sweeps. The networks have been trained with an NVIDIA V100 16 GB GPU on the MARCONI M100 supercomputer cluster. A single training takes 10 h to 24 h, depending on the initialization and the network architecture.

5.4.3 Performance Evaluation

The outputs of both networks are evaluated on both the synthetic dataset and on a dataset consisting of images experimentally obtained by MANTIS for plasma discharge #65903 on TCV. To show that the approach taken is also applicable to other machines, results for a synthetic sample of the MAST-U tokamak, with a different geometry matrix, are also shown [55]. The MAST-U sample is shown in Figure 5.8.

We evaluate the network performance on the mean squared error, the mean absolute error, and the inference time for a single sample on an NVIDIA TESLA V100 GPU.

In image generation tasks additional metrics are often used, which mainly focus on the shape and structure of the predicted image. An example of such a metric is the structural similarity index (SSIM). However, we care about the actual values produced by the network, as the output will be used for a Bayesian parameter inference model. Thus, SSIM and similar metrics are not used.

To analyse the sensitivity of the network, we use the Jacobian of the neural network [56]:

$$J_{ij}(\mathbf{y_h}) = \left.\frac{\mathrm{d}\hat{\mathbf{x}}_\mathbf{i}}{\mathrm{d}\mathbf{y}_\mathbf{i}}\right|_{\mathbf{y_h}}, \tag{5.21}$$

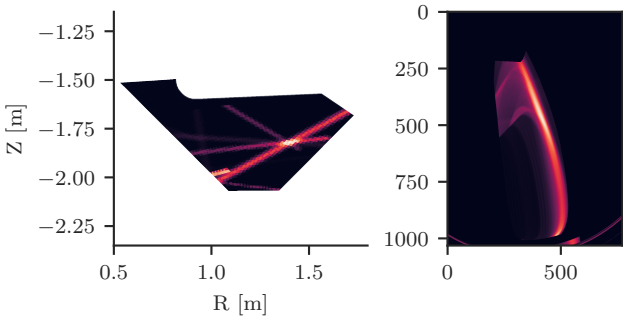


Figure 5.8: *A synthetic data pair for the MAST-U tokamak. The left image shows a synthetic poloidal emissivity profile, the right image shows the forward projection. The toroidal direction in the camera image is vertical, compared to the horizontal toroidal direction in the case of TCV.*

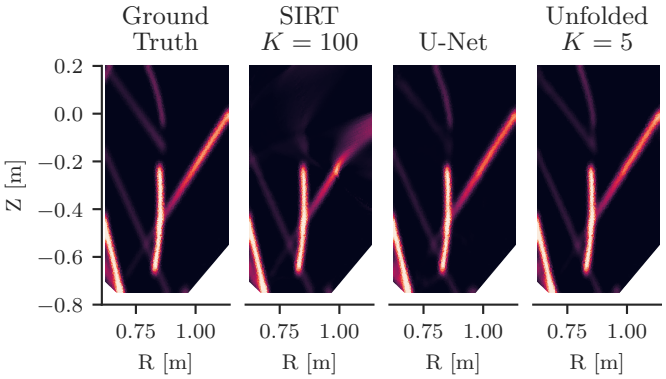


Figure 5.9: *The estimated poloidal emissivities using the classical and machine learning based methods for a sample from the synthetic dataset.*

where $\hat{\mathbf{x}}_i$ is estimated using the neural networks. The Jacobians are calculated for ten different synthetic samples, and the absolute average Jacobian is calculated to express the sensitivity. Using this sensitivity, we can approximate which parts of the image are more sensitive to changes in the input, and thus also to noise.

An implementation of the described networks and the synthetic dataset generation can be found in the following replication package: https://github.com/phaseolud/mantis-ml-inversion-replication.

5.5 Results and Discussion

5.5.1 Results on Synthetic Dataset

Figure 5.9 shows that both the U-Net based model and the deep unfolded model are successful in approximating the reconstruction of the emissivity in the poloidal plane. For data above $R = -0.2$ m, which correspond with information that is not directly in the field of view, the machine learning based solutions are more effective in reconstructing the emissivity. The emission in the image is strongly weighted towards the plane of tangency with the camera, which al-

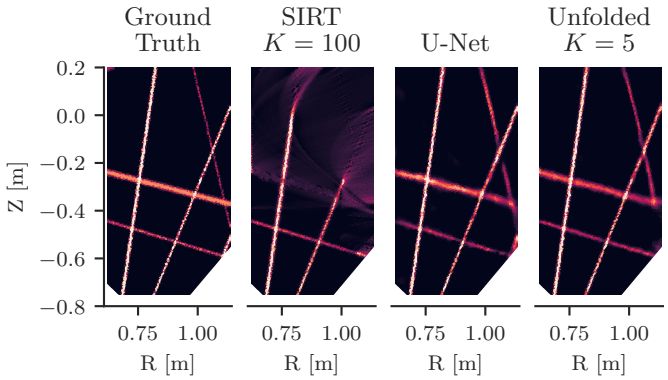


Figure 5.10: *The estimated poloidal emissivities using the classical and machine learning based methods for a synthetically generated sample, which highlights the reconstruction capabilities for emission outside the direct field of view of the camera, covering the plane of tangency to the plasma.*

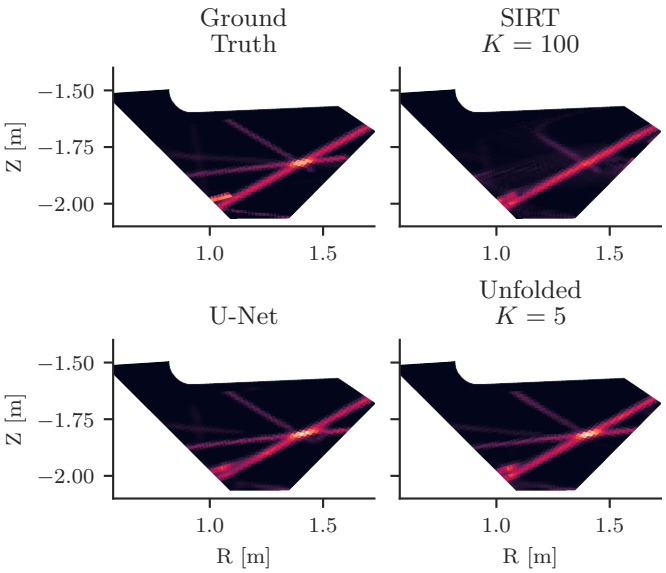


Figure 5.11: *The estimated poloidal emissivities using the classical and machine learning based methods for a synthetic sample for the MAST-U tokamak.*

lows a good reconstruction by the SIRT algorithm in these regions. However, SIRT is less successful where the plane of tangency is not captured by the camera. In this case, the poloidal emissivity is reconstructed from the toroidally symmetric emission, which is more difficult as the information is sparser in this region. The same result can be seen more clearly in Figure 5.10. In this figure one can see that the SIRT algorithm can reconstruct some emissivity for the lines higher in the poloidal plane and for the horizontal line, but an insufficient amount of data available to do a high quality reconstruction in these regions; emissions in the plane of tangency are accurately reconstructed by the SIRT algorithm, whereas emissions outside the direct field of view are more difficult to reconstruct.

Similar results can be observed for a synthetic sample for the MAST-U tokamak, shown in Figure 5.11. We can see that the machine learning based approaches outperform the SIRT algorithm, especially for the more horizontal line near $Z = -1.8$ m. The results presented for MAST-U are obtained without any tuning of the hyperparameters. Only the geometry matrix is changed, and the networks are retrained. This highlights the transferability of the approaches presented in this paper.

To quantify the performance of these three approaches, in terms of both image quality metrics and inference time, the mean squared error and mean absolute error for these three methods on a synthetic test dataset were determined. These are shown in Table 5.1. The inference times are shown in Table 5.2, which indicate the computation time for different batch sizes, evaluated on an NVIDIA V100 GPU. We can see that the error metrics correspond to the visual inspection of both figures: both the deep unfolded model and the U-Net outperform the SIRT algorithm. The deep unfolded model has a lower mean squared error, whereas the U-net has a lower mean absolute error for TCV than the other methods. Moreover, both machine learning accelerated methods are faster than the SIRT algorithm. Furthermore,

Table 5.1: *The evaluation metrics on a test dataset and the number of trainable model parameters for the different reconstruction methods.*

Model / Metric	MSE	MAE	# Parameters [M]
TCV			
SIRT	0.0596	0.0980	–
U-Net	0.0107	0.0390	1.38
Unfolded	0.0092	0.0418	0.35
MAST-U			
SIRT	0.0297	0.0511	–
U-Net	0.0044	0.0182	1.38
Unfolded	0.0077	0.0258	0.28

Table 5.2: *The inference time in* ms *for the different estimation methods for different batch sizes.*

Model / Batch Size	1	2	4	8
SIRT	52.5	67.7	80.3	88.4
U-Net	3.8	4.5	5.0	5.3
Unfolded	3.2	3.2	3.7	3.9

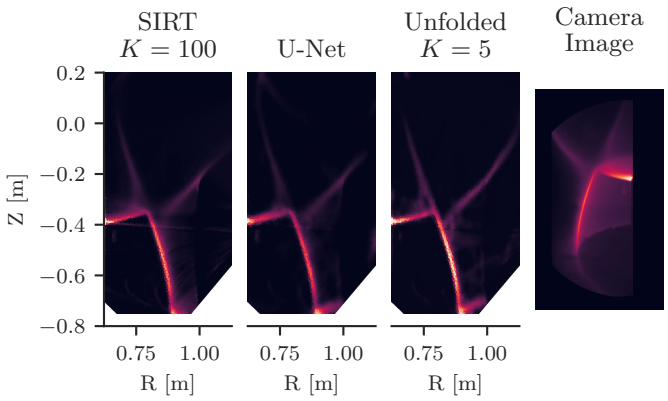


Figure 5.12: *The measured camera image and the estimated inversion for a sample from shot # 65903 measured with MANTIS. The camera image is cropped on the right to only include the region of interest containing emissions.*

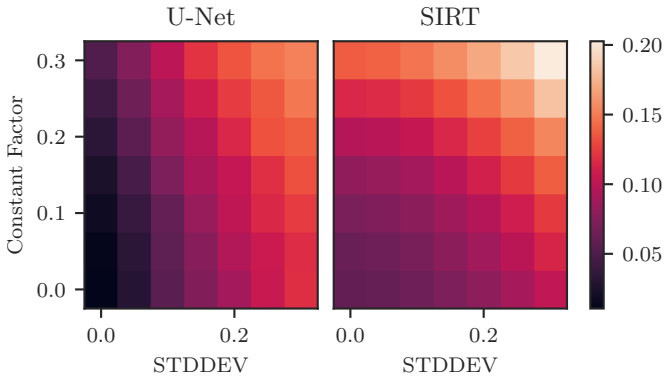


Figure 5.13: *The effect of Gaussian noise and constant noise on the MSE for both the iterative SIRT algorithm and the informed U-Net architecture.*

we see that the unfolded model is slightly faster than the U-Net model, and scales better than the U-Net model. Both machine learning solutions show results close to the target inference time of 2 ms, with room for optimization in the networks such as using half precision and using compilers such as TensorRT.

5.5.2 Results on Measured Data

In the previous subsection we have seen that the machine learning approaches are successful in reconstructing the poloidal emissivity, but the data was synthetic and thus ideal for these approaches, as the training images are generated with the same geometry matrix that is used in the networks. To verify that the machine learning approaches also yield good results on real world data, we investigate the results on an image measured with MANTIS.

The neural networks, in particular the U-Net, are also successful in approximating the inversion for an image measured using the MANTIS system, as can be seen in Figure 5.12. This figure shows us that the outputs of the neural network models are similar to the SIRT solution, especially in the region of interest in the direct field of view. Furthermore, the neural network models have fewer inversion artefacts, which are visible in the SIRT solution as dim curved lines in the bottom of the inversion and as the dim near-horizontal line at $Z = -0.4\,\mathrm{m}$. These lines arise from the non-toroidally symmetric elements in the tokamak, such as the tiles. The location of the camera and its view angle, combined with the inability of the camera to place emission along the line of sight, also create these artefacts. Interestingly, the deep unfolded network estimates higher emissivities on the divertor legs than the other methods. Results of both machine learning models show inversions corresponding to physical processes: the higher emission where the left leg touches the wall, caused by reionisation of the neutral particles coming back from the target, is correctly reconstructed. The informed U-Net model reconstructs the region around and above the x-point more accurately than the deep unfolded model: the emissivity is more uniform, and more in line with the physical processes at play, whereas the deep unfolded model creates some gaps between the emissivities. Quantifying the error on the experimentally obtained images is difficult, as no ground truth poloidal emissivity exists.

The training of the U-Net is more stable and converges to roughly the same errors for different seeds. In contrast, the deep unfolded network are much more sensitive to the random initialization of the weights, and a multi-initialization optimization is used to find the used deep unfolded network.

5.5.3 Sensitivity and Robustness

As the U-Net model is most consistent in training and thus more convenient to use in real-time applications, we analyse the sensitivity of the U-Net and its robustness compared to the SIRT algorithm for different noise levels. The effect of adding both constant and Gaussian noise to the input data is shown in Figure 5.13. This figure shows that the U-Net approach yields more accurate reconstructions for a constant noise factor. Furthermore, the SIRT algorithm yields a lower error compared to the U-Net for Gaussian noise with a higher standard deviation.

To analyse the sensitivity of the network we evaluate the network Jacobian for ten different generated samples. We then express the sensitivity as the absolute sum of these Jacobians. The resulting Jacobian has dimensions (1032, 772, 256, 256), resulting in a non-straightforward visualization. We visualize the sensitivity by downsampling and resizing the Jacobian to a 2D matrix by flattening the image dimensions. The image dimensions are reshaped with C-like ordering, so the height is the 'slow' dimension and the width the 'fast' dimension. In Figure 5.14 we see that the network is most sensitive (to noise) in the upper left, corresponding to the higher Z-coordinates in camera space and a higher vertical position in camera space. This

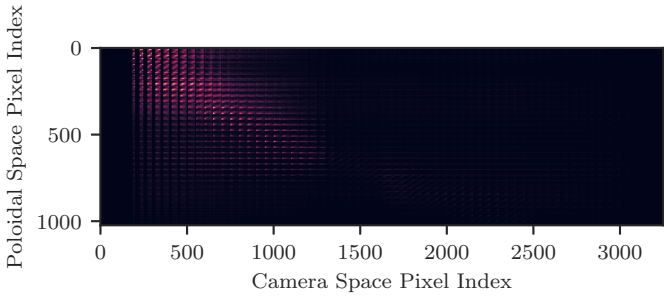


Figure 5.14: *The Jacobian to represent the sensitivity of the network. The input and output shape are reordered such that the Z-axis decreases for increasing poloidal space index and that camera space pixel index increases for a lower vertical position in the camera.*

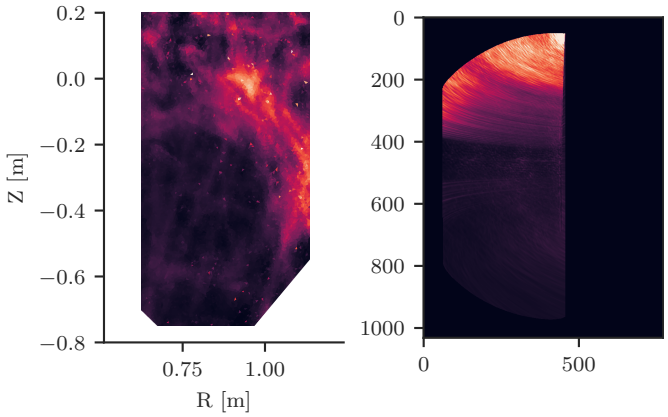


Figure 5.15: *The sensitivity summed over ten samples and the camera space and the reconstruction space, respectively. A lighter colour indicates a higher sensitivity. The higher sensitivity regions correspond to the regions where the network needs to reconstruct from a low information density.*

is according to expectations: for higher Z-coordinates the emissivity information becomes sparser as this is not in the direct field of view of the camera anymore. From this sensitivity analysis, we thus see that we should be cautious with interpreting the data for higher Z-values in the poloidal plane, as this region might be more sensitive to noise. We also see that for lower Z-values in the direct field of view the network sensitivity is low, meaning that the network results are more robust in these regions. To gain further insight in the sensitivity we sum over the poloidal and the camera space, respectively. This results in the total sensitivity for the corresponding space. This is illustrated in Figure 5.15. This figure shows that the network is most sensitive to the upper part of the camera image. This part corresponds to the low information density regions. Furthermore, the sensitivity is larger for higher R-values in the poloidal plane, which is emission closer to the camera. For a typical plasma scenario with a target at the high field side and the bottom of the tokamak the U-Net is not very sensitive to noise. The irregularities on the poloidal plane might result from the limited size of the dataset, which only consists of ten samples.

5.6 Conclusion

2D measurements of plasma emissivity can yield plenty of information about the plasma state, such as plasma temperature, plasma density and neutral density. These measurements are used to both control the heat and particle flux towards the target, and to reconstruct 2D plasma parameter profiles. For these reconstructions, emissivities in the poloidal plane are needed. However, 2D imaging camera system measurements, such as from the MANTIS diagnostic at TCV, are line integrated: each pixel contains information from multiple poloidal coordinates. The poloidal emissivity can be recovered with tomographic inversion techniques such as the SIRT algorithm. However, these techniques are generally slow and not suitable for real-time control.

To accelerate the inversion process, we developed two neural network architectures: an informed U-Net architecture, which uses model information by processing a back projected image, and a deep unfolded architecture, adapted from the SIRT algorithm. To effectively train these networks and reduce potential overfitting, we created a representative synthetic dataset. Both networks outperform the SIRT algorithm in terms of accuracy and inference time evaluated on the synthetic dataset. The machine learning based approaches are more successful in reconstructing emission which is not directly in the field of view of the camera. These architectures also outperform the iterative approaches for other geometries and tokamaks, even without any hyperparameter tuning. Only changing the geometry matrix and retraining the network is required. However, it is difficult to find a suitable metric for the reconstruction accuracy of these measured images as no objective ground truth is available. However, the classical methods share this same limitation.

5.7 Outlook

Thus far, we have evaluated the U-Net and unfolded models using the MSE and MAE with respect to the actual inversion or the camera image. The goal of the fast inversion approximations is to use them to infer plasma parameters on TCV based on collisional-radiative modelling. Therefore, an additional useful evaluation is to calculate the outputs for the different inversion methods and use them as inputs for the Bayesian parameter inference code. Another possible evaluation process is to apply the detachment front tracking algorithm used for control and check if the outputs of machine learning based inversions correspond with the SIRT output. Furthermore, the fast reconstructions allow for feedback control on tokamaks with geometries where the plane of tangency is not directly visible.

In this study the assumption to not model the reflections was made, as reflections would result in a denser geometry

matrix and thus a higher computational load. However, current and future reactors such as ITER and DEMO are expected to have a metallic wall. Analysis of the presented approaches for metallic wall devices is thus of interest.

The same architectures can also be applied to other geometries and machines. However, these methods might also have potential for other line-integrated measurements such as bolometry.

Finally, the fast poloidal reconstruction of the emissivities is only the first step in a two-step process for real-time plasma parameter inference. The second step is to infer the plasma state (temperatures and densities) from 2D poloidal emissivities. This is again an inverse problem, related to the collisional-radiative processes which relate the local plasma parameters to local emissivity. Machine learning techniques provide further opportunity in accelerating the inversion of this second step, allowing for the development and application of innovative control techniques for tokamak power exhaust.

Appendices

5.A Stability Proof of the SIRT Algorithm

For a stable convergence we must have that the term $(I - \mu C G^T R G)$ has a spectral radius $\rho(\cdot)$ smaller than one, i.e. all eigenvalues λ are inside the unit circle. The eigenvalues are equal to:

$$\lambda = 1 - \lambda_i;\ \lambda_i = \lambda_i(C G^T R G). \tag{5.22}$$

Because the matrices C, R and $G^T G$ are all positive semi-definite, we have that the eigenvalues λ_i are greater than or equal to zero. Next, we use that the spectral radius of a matrix is smaller than or equal to the matrix norm, for any natural matrix norm:

$$\rho(A) \leq \|A\|. \tag{5.23}$$

By rewriting this condition using the ∞ norm, we have that:

$$\rho(C G^T R G) \leq \left\|C G^T R G\right\|_\infty \leq \left\|C G^T\right\|_\infty \|R G\|_\infty, \tag{5.24}$$

where the last inequality follows from the sub-multiplicativity of a matrix norm. We choose to use the ∞ norm here, because the explicit expression for the ∞ norm of a matrix is given by the largest row sum:

$$\|A\|_\infty = \max_k \sum_l \left|a_{k,l}\right|, \tag{5.25}$$

which motivates the choice for the C and R to equal the inverse row (and column) sum of G. We now have that $\|R G\|_\infty = 1$ and $\left\|C G^T\right\|_\infty = 1$. Combining these results gives us:

$$\rho(I - \mu C G^T R G) = \max_i |1 - \mu\lambda_i| = |1 - \mu|. \tag{5.26}$$

The requirement for stability is that the spectral radius is smaller than one, so the bounds on the step size μ can be determined as:

$$|1 - \mu| < 1 \implies 0 < \mu < 2. \tag{5.27}$$

5.B Convolutional Neural Networks

In conventional feedforward neural networks a connection between two hidden layers $\mathbf{x}, \mathbf{y}$ is given by:

$$\mathbf{y} = f\left(W\mathbf{x} + \mathbf{b}\right), \tag{5.28}$$

where $\mathbf{x}$ are the inputs, $\mathbf{y}$ the outputs, $\mathbf{b}$ the bias, W the weights and f the non-linear activation function. The weights and biases are learned through back propagation. For inputs and outputs with large dimensions, in the case of images, feedforward neural networks would not be tractable: an image to image transformation with images of size 256 by 256 would result in a weight matrix with $256^2 \cdot 256^2 \approx 4.3 \cdot 10^9$ learnable parameters.

By using the assumptions that the data is locally correlated, the full connections can be replaced by a (2D) convolution, where the kernel $H \in \mathbb{R}^{k_0, k_1}$ (also called filter) is learned:

$$\mathbf{y} = f\left(b + \sum_{s_0=0}^{k_0-1} \sum_{s_1=0}^{k_1-1} H_{s_0,s_1} \mathbf{x}_{i+s_0, j+s_1}\right), \tag{5.29}$$

where a 2D input and output are assumed. This is equivalent to learning a weight matrix with only elements on the tridiagonal in a Toeplitz structure, with all other elements set to zero. More complexity can be added to the layer by increasing the number of learnable kernels H. If we denote the convolutional operator with filter H as $T_H(\mathbf{x})$, we can combine the different kernels as:

$$\mathbf{y_k} = f\left(b_k + \sum_l^L T_{H_k}(\mathbf{x_l})\right), \tag{5.30}$$

where k denotes the kernel index of the output $\mathbf{y}$ and L the number of kernels of the input $\mathbf{x}$.

5.C Comparison Between Different U-Nets

In this paper we have focussed on a model-informed architectures. It is also possible to learn a direct mapping from camera image to poloidal emissivity. However, this would require the neural network to learn a more difficult mapping; the back projection already yields information about the location of the emissivity. Furthermore, the back projection yields a mapping from the camera image to the reconstruction grid. For the direct mapping there should be a preprocessing and resizing step to transform the camera image dimensions to the appropriate input shape of the U-Net.

The results for the synthetic poloidal emissivities and camera images, and the experimentally obtained camera images that are presented in this paper are used to evaluate the uninformed U-Net architecture. Figure 5.16 shows a good reconstruction of the poloidal emissivity by the uninformed U-Net, better than the SIRT algorithm. The capabilities

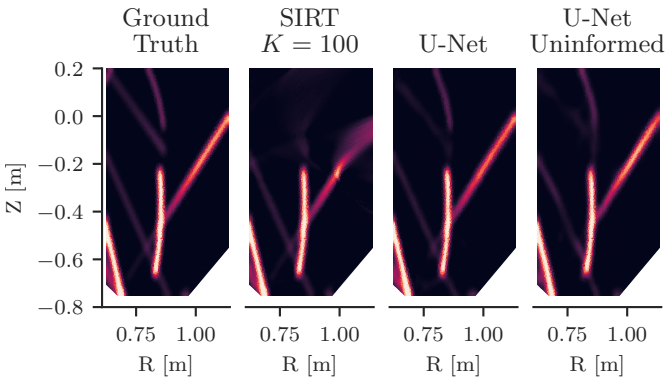


Figure 5.16: *A synthetically generated poloidal emissivity and the reconstructions with the SIRT algorithm, the model-informed U-Net and the uninformed U-Net.*

of the uninformed become more clear in Figure 5.17. In

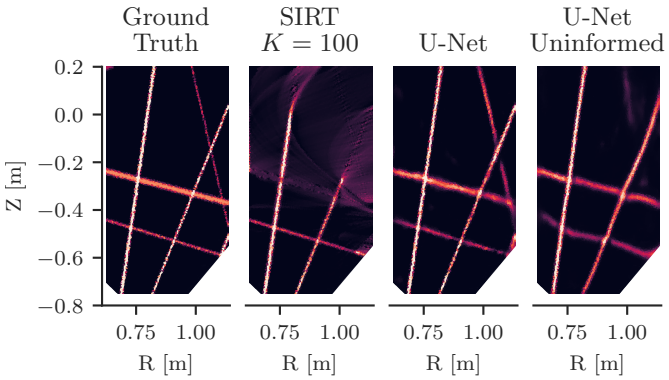


Figure 5.17: *A synthetically generated poloidal emissivity and the reconstructions with the SIRT algorithm, the model-informed U-Net and the uninformed U-Net.*

this figure we can see that the uninformed U-Net creates more 'wavy' lines, in contrast to the straight lines estimated by the SIRT algorithm and the informed U-Net. Nevertheless, the results of the uninformed U-Net are more in line with the ground truth than the SIRT algorithm. This is confirmed by the quantitative evaluation in Table 5.3. The shortcomings of the uninformed U-Net become clear with the experimentally obtained camera images, shown in Figure 5.18. This figure shows that the uninformed U-Net fails to do an accurate reconstruction of the poloidal emissivity: the X-point is estimated at a different position in the poloidal plane. Moreover, the estimates emissivities are not smooth and underestimate the emissivities near the walls. The uninformed U-Net does not generalize well for out of distribution data, as is the case with the experimentally obtained camera image. Phrased differently: the inclusion of

Table 5.3: *The evaluation metrics on a test set for the TCV tokamak for the different reconstruction methods.*

Model / Metric	MSE	MAE
SIRT	0.0596	0.0980
U-Net	0.0107	0.0390
Unfolded	0.0092	0.0418
U-Net Uninformed	0.0238	0.0597

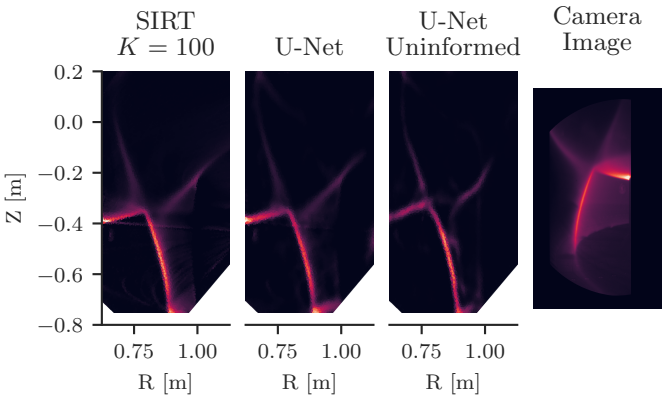


Figure 5.18: *The measured camera image and the estimated poloidal emissivities for a sample from shot # 65903 measured with MANTIS.*

the geometry matrix in the network architecture allows for generalizability and makes the network more performant on out of distribution data.

6 Additional Results and Discussion

In this chapter we discuss additional results and analysis that were not presented in the paper. First we analyse the intermediate Representation of the deep unfolded network. Next, we look at the calculated poloidal emissivities for the noisy inputs. Finally, we discuss the limitations of the synthetic dataset.

6.1 Intermediate Representation of the Deep Unfolded Network

The used deep unfolded network is unfolded for $K = 5$ iterations. The loss during training is calculated as a sum of the loss at all intermediate representations, so it might be interesting to see if the estimated emissivity improves for each iteration. Table 6.1 shows the errors on the synthetic test dataset for each unfold, together with learned values for the step size parameters μ_k. The MSE and MAE decrease monotonically, as intended by the loss at every unfold step. Furthermore, the step sizes mu_k are high compared to the stability limit of 1.99 for unfold two and four. This also means that more of the measurement data and model is incorporated in that specific step. To gain more insight in the intermediate representations, the outputs for each layer are visualized in Figures 6.1 and 6.2.

These figures show that for 'simple' poloidal emissivities the results after the first unfold are already satisfactory. However, the more difficult to see emission profiles (e.g. the more horizontal lines in Figure 6.2), need a higher number of unfolds to be reconstructed correctly.

6.2 Limitations of the Synthetic Dataset

Using a synthetic dataset has plenty of advantages, such as: fast data creation, less change on overfitting and having a ground truth target. However, the synthetic dataset is not perfect: this became clear in the results of both the uninformed U-Net architecture and in the deep unfolded architecture. The uninformed U-Net architecture estimated the x-point in a different location, whereas the deep unfolded network estimated a higher brightness on the leg, which are errors mostly visible for experimentally obtained data.

There are two main sources of mismatches between the synthetic dataset and the experimentally obtained data:

Table 6.1: *The MSE and MAE calculated using the outputs of the learned proximal layer of each iterative step. The bottom row contains the learned step sizes used in the SIRT algorithm. The final column denotes the output of the network.*

Unfold Step k	0	1	2	3	4	O
MAE	0.0903	0.0848	0.0755	0.0672	0.0655	0.0418
MSE	0.0370	0.0332	0.0259	0.0134	0.0114	0.0092
μ_k	1.16	3.13	6.94	1.31	6.61	–

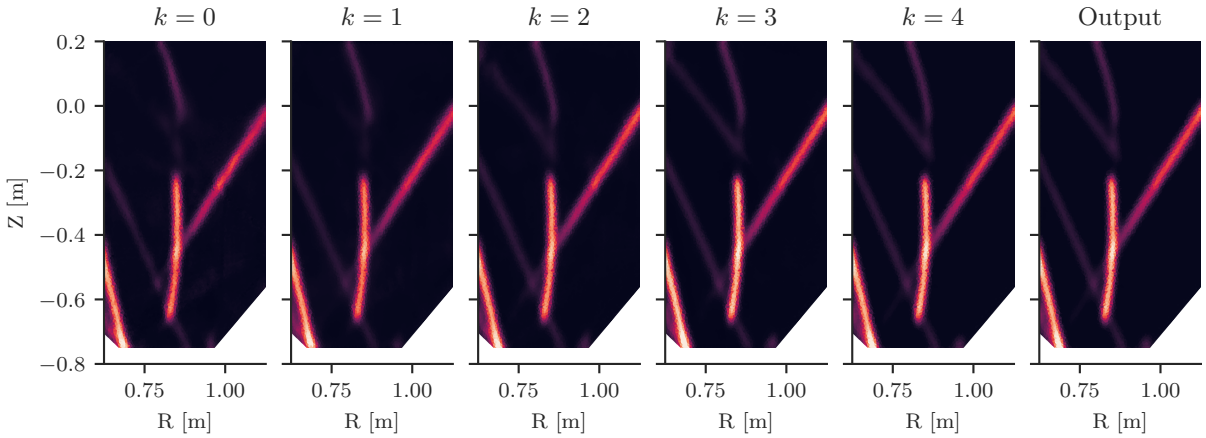


Figure 6.1: *The intermediate outputs for the deep unfolded architecture for a synthetic dataset sample. The results are already good after the first unfold, because of the 'simple' poloidal emissivity profile.*

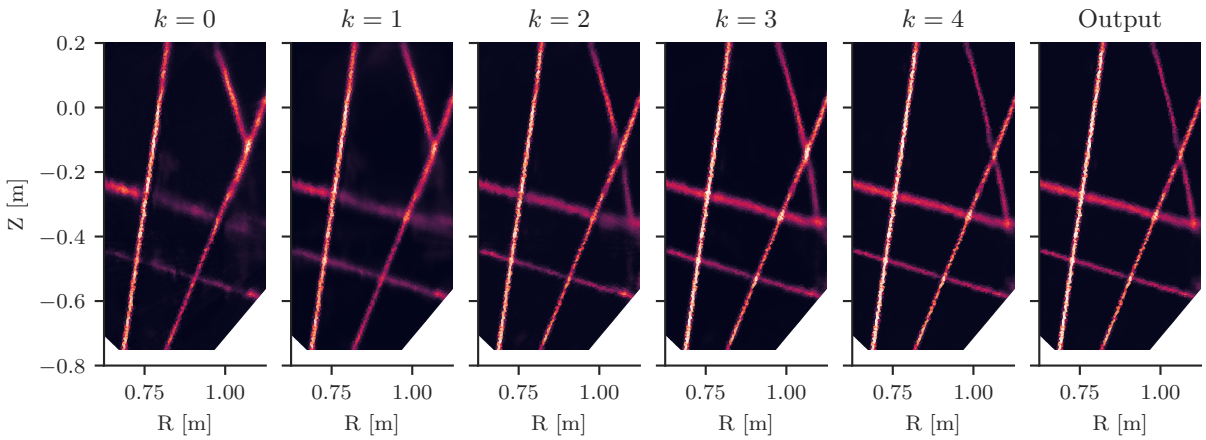


Figure 6.2: *The intermediate outputs for the deep unfolded architecture for a synthetic dataset sample. The simple to reconstruct emissivities are already reconstructed on the first iteration, whereas the emissions not in the direct field of view only become more clear for higher number of unfolds.*

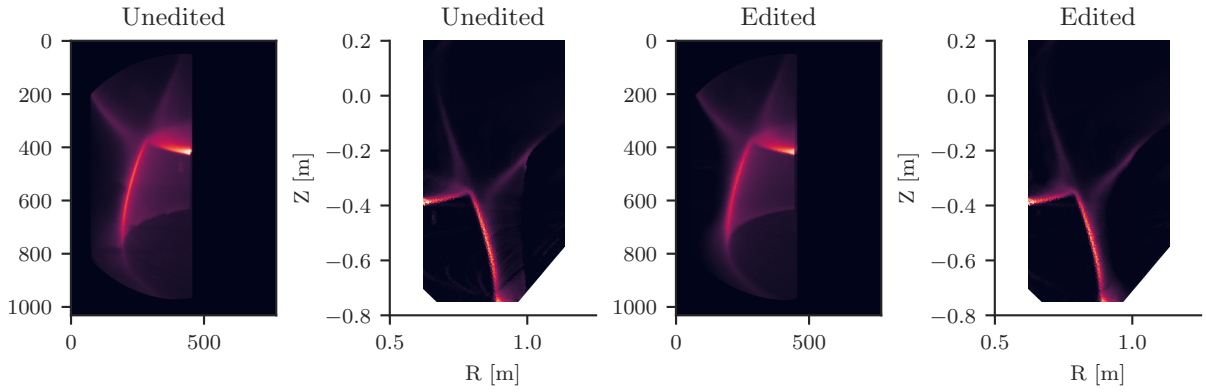


Figure 6.3: *The measured camera image and the calculated tomographic reconstruction using the SIRT algorithm on the left. The poloidal emissivity is edited to remove most of the reconstruction artefacts. The camera image on the right is generated by forward modelling the edited poloidal emissivity. Note the absence of tiles at the bottom, and the absence of the port on the right (middle vertical) for the synthetic camera image.*

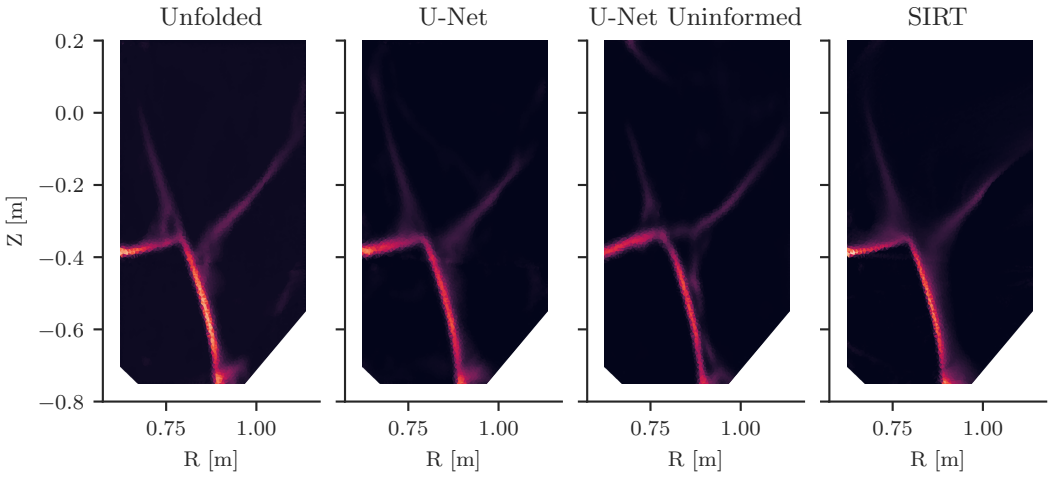


Figure 6.4: *The estimated poloidal emissivities with the different machine-learning methods for the synthetic camera image. The estimated poloidal emissivities are almost identical to the ones estimated from the experimentally obtained camera image.*

1. The geometry matrix G is ray-traced from a fitted calibration. The fitted calibration is not perfect (hence a fit), and already introduces some error. Furthermore, the geometry matrix G is a linear approximation of the system, omitting potential non-linear effects of the lenses or light aberrations. Finally, we do not model reflection, whereas they are (slightly) present in the tokamak.
2. The synthetic dataset samples have certain characteristics, such as curves and gradients. Furthermore, different line widths and blur kernels are being used to make the dataset more diverse. However, the dataset does not include the exact emissivities present in the experimental data: the experimental data has more of glow around the legs and has more uniform regions of emissivity near the X-point.

We can find out more about the shortcomings of the synthetic dataset by analysing the experimentally obtained data. We use the SIRT estimated poloidal emissivity, and remove the reconstruction artefacts by hand. The original camera image and reconstruction, and the edited reconstruction and forward modelled camera image are shown in Figure 6.3. By using the synthetic camera image from the edited poloidal emissivity, we can find out if the suboptimal results for the uninformed U-Net and the higher emission on the leg result from modelling errors or from a synthetic dataset that is not fully representative. If the machine-learning methods yield undesirable results for the synthetic camera image (i.e. we have a perfect model), we know that the dataset could be improved. The estimated poloidal emissivities for the synthetic camera image are shown in Figure 6.4. In this figure we can see the same undesirable features as in the emissivities estimated from the experimentally obtained camera image. This indicates that the synthetic

dataset could still be improved to better emulate the emissivities measured in a tokamak. However, this is also good news: the (asymmetric) reflections such as the ports and tiles do not influence the estimated poloidal emissivity a lot, indicating that additional effort in a higher-fidelity model or geometry matrix G is not as important as improving the dataset.

7 Deployment for Real-Time Tomographic Reconstructions

The machine learning accelerated methods presented in this thesis yield highly satisfactory results. However, the timing and testing has all be done on a local laptop and the Marconi M100 supercomputer. This does not represent a real-time environment where robustness and easy of deployment is required. At the time of writing a suitable GPU just arrived. However, there has not yet been an opportunity to test the GPU for real-time deployment. Nonetheless, some of the solutions have been implemented already on a local computer running Linux. This chapter describes possible methods we have considered (in advance), in collaboration with NVIDIA.

7.1 Integration into the MANTIS TCV system

The current real-time MANTIS architecture consists of up to ten PCI Express (PCIe) compatible cameras. PCIe is a serial interface, which is the standard for connecting elements of a computer such as graphics cards and storage drives to the motherboard. The camera data streams are aggregated in a single PCIe interface. The data is then streamed from the interface to a host computer through optical fibres. The data is then directly streamed to the RAM, after which the data is processed by the CPUs, with a separate CPU for each camera, allowing for parallel processing. The processed data is then aggregated and being prepared for the digital control system (SCD) in a separate CPU. This CPU puts the data in the reflective memory, which is similar to a shared RAM between computers. The SCD reads from the processed data from this reflective memory and then uses this to compute and send a control signal. All this information is from [12].

In the case of the machine-learning based tomographic reconstructions, the pipeline should be altered. First, the data should not be written to the RAM, but directly into the GPU memory. Then, the machine-learning reconstruction algorithms should have a pointer to the memory location, allowing for direct processing without copying the data through the CPU, as is often the case. The direct writing to GPU memory is not a standard feature available on every GPU, so for NVIDIA GPUs, the GPU should support GPUDirect RDMA[1]. Finally, the obtained poloidal reconstructions should then be used to infer the 2D plasma parameters, such as the plasma temperature, plasma density and neutral density. The CPU based and GPU based data flow is illustrated in Figure 7.1.

7.2 Real-Time Deployment Architectures

The above-mentioned GPU-based data flow would require up to eleven GPUs which is not only costly but also adds an overhead to manage the communication between those GPUs. Furthermore, motherboards with this amount of PCIe slots are not very common. We discuss two different approaches we can take: using multi-instance GPU virtualization or using NVIDIA Triton.

[1] See https://docs.nvidia.com/cuda/gpudirect-rdma/index.html

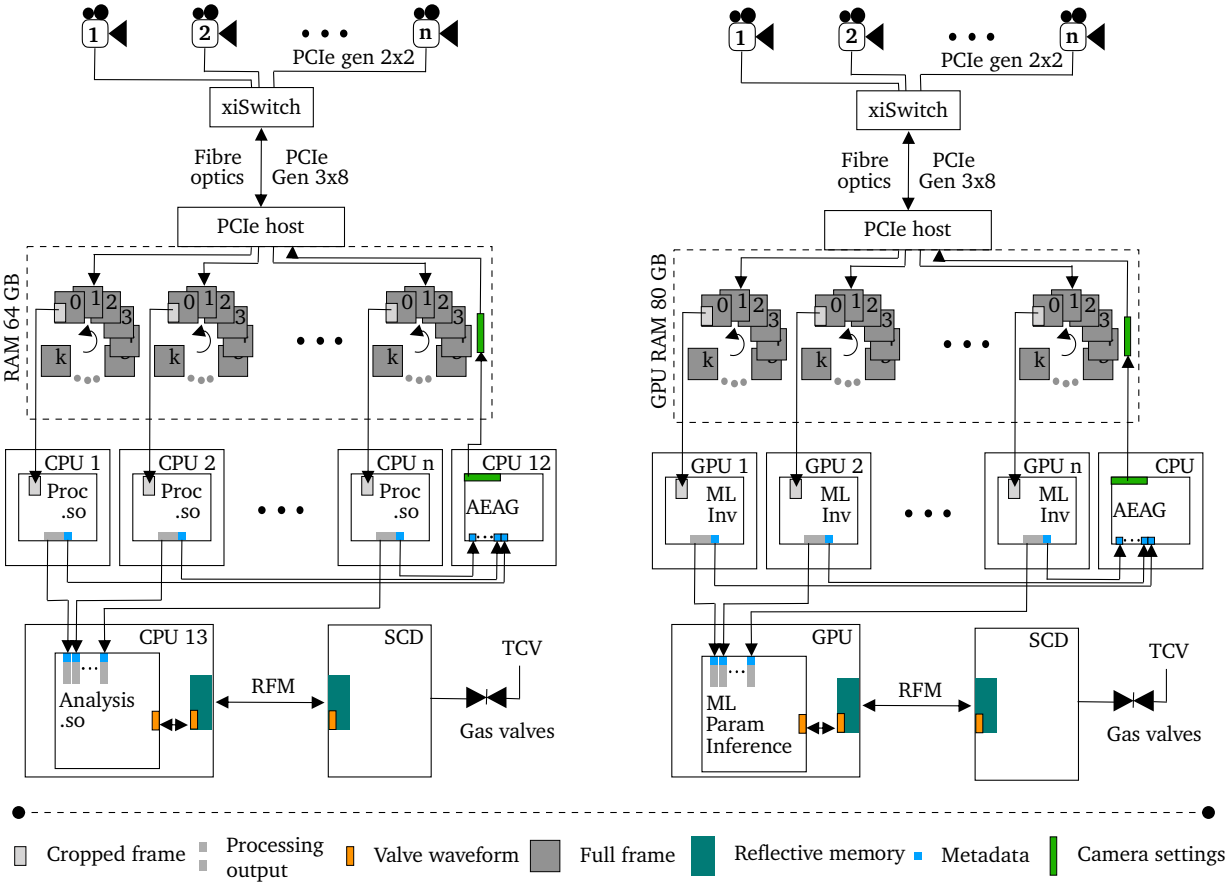


Figure 7.1: *The CPU-based data flow (left) and the GPU-based data flow (right). The cameras (top) measure the plasma emissivity. These data is then streamed to the memory through the PCIe interface and the fibre optics. The CPU/GPU process the acquired images in parallel (middle) after which the processed measurements of the different cameras are combined. The combined data is again processed to compute a control signal, which is transferred through the reflective memory to the control system (bottom). Adapted from [12].*

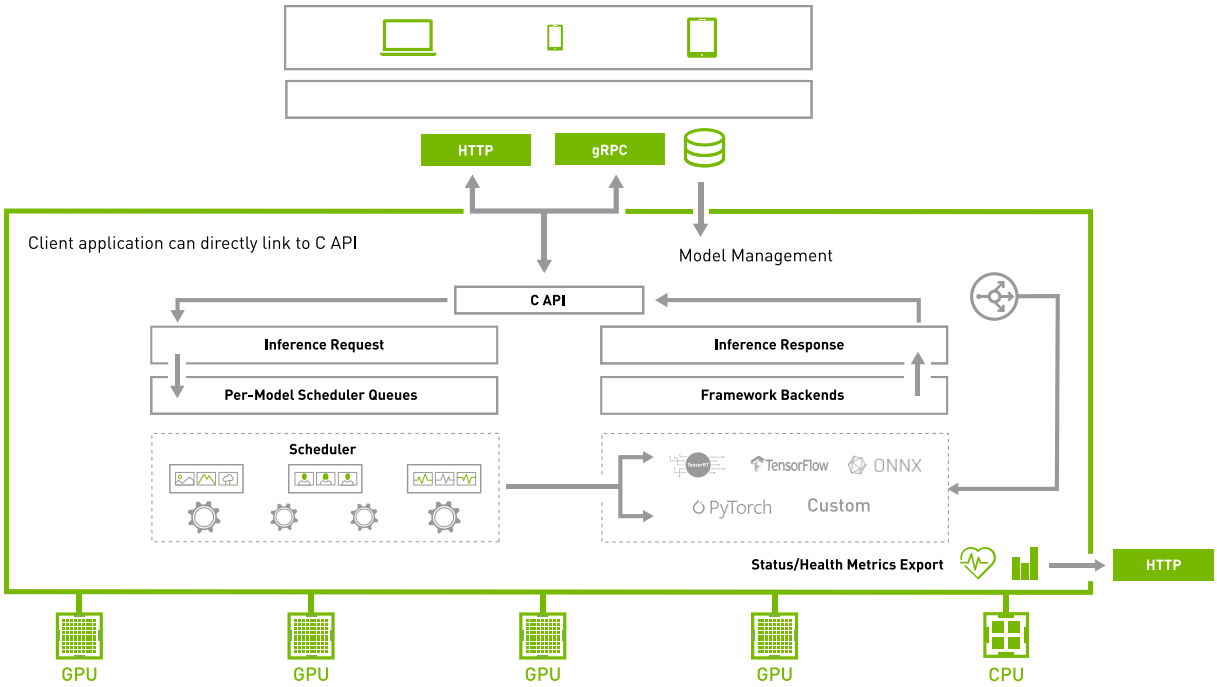


Figure 7.2: *A diagram illustrating the Triton inference server. The server can be called through HTTP or gRPC, both exposed in a python and C++ API. Different models can be added by adding the files to the model repository. Copied from https://developer.nvidia.com/nvidia-triton-inference-server.*

7.2.1 Multi-Instance GPU Virtualization

Newer NVIDIA GPUs allow for multi-instance GPU (MIG) virtualization[2], where one GPU is split up into up to seven instances. Each instance is a fully isolated system, with its own memory and compute cores. The benefit of use MIG is that errors and potential crashes would not affect the other instances. Furthermore, every process can take place in parallel, allowing for a maximum GPU utilization. MIG would also allow for other GPU tasks at TCV to run on the available GPU, without interfering with the real-time reconstruction.

7.2.2 NVIDIA Triton Inference Server

MIG is powerful and flexible, but at the same time needs manual configuration. For a real-time environment such as TCV, we would ideally have the reconstructions models loaded and configured automatically. A solution for this problem is the Triton inference server[3], developed by NVIDIA. The Triton inference server supports concurrent processing, just like MIG. The inference server runs in a separate docker container, where communication to the server is done either through HTTP requests or gPRC; an API is available for both python and C++. The data can be supplied as the body of the request, but it is also possible to pass a pointer to the location of the data on the GPU. New models can be by easily adding them to a folder called the model repository. Triton automatically detects these new models, and serves them directly. The inference server works out of the box for all popular machine learning libraries and export formats. An overview of the Triton inference server is shown in Figure 7.2. Two other notable benefits of Triton are the automatic and easy monitoring of metrics[4], and the model analyser[5]. The model analyser automatically finds the needed allocations, configurations and settings for the model being used combined with the latency and throughput requirements.

[2] See https://www.nvidia.com/en-us/technologies/multi-instance-gpu/
[3] See https://developer.nvidia.com/nvidia-triton-inference-server
[4] See https://github.com/triton-inference-server/server/blob/main/docs/user_guide/metrics.md
[5] See https://github.com/triton-inference-server/model_analyzer

7.3 Practical Approach for Implementation

To incorporate the above-mentioned technologies and the machine-learning algorithm into the MANTIS system, the following incremental approach is recommended:

1. Install the GPU into the computer, together with the required drivers and software.
2. Create an example (e.g. summing two numbers) with the Triton inference server, first using the python API and then the C++ API.
3. Test the RDMA capabilities for the GPU, and create an example using C++ that writes the camera data to a memory location on the GPU. *In parallel with the previous item.*
4. Create an example (summing two numbers) with the Triton inference server, using a memory pointer on the GPU instead of directly sending the values using the API.
5. Create an example with the Triton inference server, using a memory pointer on the GPU to a synthetic camera image. The camera image is processed with the designed machine-learning reconstruction algorithms.
6. Create an example using C++ where camera images are written to the GPU memory directly, and where the Triton inference server uses this camera image to do a simple operation, e.g. doubling the values.
7. Create an example using C++ where pointers to the camera image are used in the Triton inference server. The Triton inference server uses the designed machine-learning reconstruction algorithms.

Using the proposed approach we can incrementally test the different elements of the processing pipeline.

8 Conclusions and Outlook

2D measurements of plasma emissivity can yield plenty of information about the plasma state, such as plasma temperature, plasma density and neutral density. These measurements are used to both control the heat and particle flux towards the target, and to reconstruct 2D plasma parameter profiles. For these reconstructions, emissivities in the poloidal plane are needed. However, 2D imaging camera system measurements, such as from the MANTIS diagnostic at TCV, are line integrated: each pixel contains information from multiple poloidal coordinates. The poloidal emissivity can be recovered with tomographic inversion techniques such as the SIRT algorithm. However, these techniques are generally slow and not suitable for real-time control.

To accelerate the inversion process, we developed two neural network architectures: an informed U-Net architecture, which uses model information by processing a back projected image, and a deep unfolded architecture, adapted from the SIRT algorithm. To effectively train these networks and reduce potential overfitting, we created a representative synthetic dataset. Both networks outperform the SIRT algorithm in terms of accuracy and inference time evaluated on the synthetic dataset. The machine learning based approaches are more successful in reconstructing emission which is not directly in the field of view of the camera. These architectures also outperform the iterative approaches for other geometries and tokamaks, even without any hyperparameter tuning. Only changing the geometry matrix and retraining the network is required. However, it is difficult to find a suitable metric for the reconstruction accuracy of these measured images as no objective ground truth is available. However, the classical methods share this same limitation.

We have also shown that the inclusion of the geometry matrix in the U-Net does not yield great differences for the synthetic dataset. However, it allows for generalizability in the case of out of distribution data, i.e. experimentally obtained camera images, whereas the uninformed U-Net lacks accuracy in this case.

8.1 Outlook

Thus far, we have evaluated the U-Net and unfolded models using the MSE and MAE with respect to the actual inversion or the camera image. The goal of the fast inversion approximations is to use them to infer plasma parameters on TCV based on collisional-radiative modelling. Therefore, an additional useful evaluation is to calculate the outputs for the different inversion methods and use them as inputs for the Bayesian parameter inference code. Another possible evaluation process is to apply the detachment front tracking algorithm used for control and check if the outputs of machine learning based inversions correspond with the SIRT output.

Furthermore, the fast reconstructions allow for feedback control on tokamaks with geometries where the plane of tangency is not directly visible. In this thesis we have shown that the proposed approach also transfers to MAST-U data without any fine-tuning. Nonetheless, this result was obtained on synthetic data, and additional steps in the synthetic dataset generation might be needed to create valid results for experimentally obtained data.

The deep unfolded model predicted larger poloidal emissivities for the experimentally obtained data than both the U-Net and the SIRT algorithm, whereas it performed better on the synthetic dataset. We do not know the exact reason why it predicts a larger emissivity there, but we can say that this indicates that the

synthetic dataset is not fully representative. Thus, further efforts should be made in better analysing the current dataset and the experimental data, and find what the most important missing features are.

In this study the assumption to not model the reflections was made, as reflections would result in a denser geometry matrix and thus a higher computational load. However, current and future reactors such as ITER and DEMO are expected to have a metallic wall. Analysis of the presented approaches for metallic wall devices is thus of interest. We might use geometry matrix without reflections can be used in the network (as it has fewer elements), while using the reflections based geometry matrix for the training set generation. In this case, the reflections would be handled by the neural networks elements: the U-Net post-processing or the proximal operators.

The same architectures can also be applied to other geometries and machines. However, these methods might also have potential for other line-integrated measurements such as bolometry.

Interesting might be to analyse if additional measured data could be incorporated into the network, and perform some kind of sensor fusion. In the shown derivation we assume the distributions to be of the form $p(\cdot|G)$. However, in reality the prior is also dependent on the physical processes at play, so it might be of the form $p(\cdot|G, \mathbf{m})$, where m are additional plasma parameters, such that the distribution of the poloidal emissivity can be expressed as $p(\mathbf{x}|\mathbf{y}, \mathbf{m}, G)$.

Finally, the fast poloidal reconstruction of the emissivities is only the first step in a two-step process for real-time plasma parameter inference. The second step is to infer the plasma state (temperatures and densities) from 2D poloidal emissivities. This is again an inverse problem, related to the collisional-radiative processes which relate the local plasma parameters to local emissivity. Machine learning techniques provide further opportunity in accelerating the inversion of this second step, allowing for the development and application of innovative control techniques for tokamak power exhaust.

Bibliography

[1] B. J. van Ruijven, E. D. Cian, and I. S. Wing. "Amplification of future energy demand growth due to climate change". In: *Nature Communications 2019 10:1* 10 (1 June 2019), pp. 1–12. issn: 2041-1723. doi: 10.1038/s41467-019-10399-3. url: https://www.nature.com/articles/s41467-019-10399-3.

[2] J. H. You et al. "Divertor of the European DEMO: Engineering and technologies for power exhaust". In: *Fusion Engineering and Design* 175 (Feb. 2022), p. 113010. issn: 0920-3796. doi: 10.1016/J.FUSENGDES.2022.113010.

[3] R. Pitts et al. "Physics basis for the first ITER tungsten divertor". In: *Nuclear Materials and Energy* 20 (Aug. 2019), p. 100696. issn: 23521791. doi: 10.1016/j.nme.2019.100696.

[4] A. Kallenbach et al. "Impurity seeding for tokamak power exhaust: from present devices via ITER to DEMO". In: *Plasma Physics and Controlled Fusion* 55 (12 Nov. 2013), p. 124041. issn: 0741-3335. doi: 10.1088/0741-3335/55/12/124041. url: https://iopscience.iop.org/article/10.1088/0741-3335/55/12/124041%20https://iopscience.iop.org/article/10.1088/0741-3335/55/12/124041/meta.

[5] T. Ravensbergen. "Advanced methods in control of the core density and divertor detachment in nuclear fusion devices". Eindhoven University of Technology, DIFFER, Feb. 2021. isbn: 9789038652078.

[6] O. Février et al. "Nitrogen-seeded divertor detachment in TCV L-mode plasmas". In: *Plasma Physics and Controlled Fusion* 62 (3 Feb. 2020), p. 035017. issn: 0741-3335. doi: 10.1088/1361-6587/AB6B00. url: https://iopscience.iop.org/article/10.1088/1361-6587/ab6b00%20https://iopscience.iop.org/article/10.1088/1361-6587/ab6b00/meta.

[7] G. F. Matthews. "Plasma detachment from divertor targets and limiters". In: *Journal of Nuclear Materials* 220-222 (Apr. 1995), pp. 104–116. issn: 0022-3115. doi: 10.1016/0022-3115(94)00450-1.

[8] J. P. Gunn et al. "Surface heat loads on the ITER divertor vertical targets". In: *Nuclear Fusion* 57 (4 Mar. 2017), p. 046025. issn: 0029-5515. doi: 10.1088/1741-4326/AA5E2A. url: https://iopscience.iop.org/article/10.1088/1741-4326/aa5e2a%20https://iopscience.iop.org/article/10.1088/1741-4326/aa5e2a/meta.

[9] J. A. Goetz et al. "High confinement dissipative divertor operation on Alcator C-Mod". In: *Physics of Plasmas* 6 (5 Apr. 1999), p. 1899. issn: 1070-664X. doi: 10.1063/1.873447. url: https://aip.scitation.org/doi/abs/10.1063/1.873447.

[10] M. Bernert. "Analysis of the H-mode density limit in the ASDEX Upgrade tokamak using bolometry". Ludwig-Maximilians-Universität, Max-Planck-Institut für Plasmaphysik, 2013.

[11] O. Février et al. "Analysis of wall-embedded Langmuir probe signals in different conditions on the Tokamak à Configuration Variable". In: *Review of Scientific Instruments* 89 (5 May 2018), p. 053502. issn: 0034-6748. doi: 10.1063/1.5022459. url: https://aip.scitation.org/doi/abs/10.1063/1.5022459.

[12] A. Perek et al. "MANTIS: A real-time quantitative multispectral imaging system for fusion plasmas". In: *Review of Scientific Instruments* 90 (12 Dec. 2019). issn: 0034-6748. doi: 10.1063/1.5115569. url: https://aip.scitation.org/doi/abs/10.1063/1.5115569.

[13] A. Perek et al. "Measurement of the 2D emission profiles of hydrogen and impurity ions in the TCV divertor". In: *Nuclear Materials and Energy* 26 (Mar. 2021), p. 100858. issn: 2352-1791. doi: 10.1016/J.NME.2020.100858.

[14] T. Ravensbergen et al. "Development of a real-time algorithm for detection of the divertor detachment radiation front using multi-spectral imaging". In: *Nuclear Fusion* 60 (6 May 2020), p. 066017. issn: 0029-5515. doi: 10.1088/1741-4326/AB8183. url: https://iopscience.iop.org/article/10.1088/1741-4326/ab8183%20https://iopscience.iop.org/article/10.1088/1741-4326/ab8183/meta.

[15] M. A. Lieberman and A. J. Lichtenberg. “Principles of Plasma Discharges and Materials Processing: Second Edition”. In: *Principles of Plasma Discharges and Materials Processing: Second Edition* (Jan. 2005), pp. 1–757. doi: 10.1002/0471724254. url: https://onlinelibrary.wiley.com/doi/book/10.1002/0471724254.

[16] A. Perek et al. *A spectroscopic inference and SOLPS-ITER comparison of flux-resolved edge plasma parameters in detachment experiments on TCV*. 2021.

[17] K. Verhaegh et al. “Spectroscopic investigations of divertor detachment in TCV”. In: *Nuclear Materials and Energy* 12 (Aug. 2017), pp. 1112–1117. issn: 2352-1791. doi: 10.1016/J.NME.2017.01.004.

[18] C. Bowman et al. “Development and simulation of multi-diagnostic Bayesian analysis for 2D inference of divertor plasma characteristics”. In: *Plasma Physics and Controlled Fusion* 62 (4 2020). issn: 13616587. doi: 10.1088/1361-6587/AB759B.

[19] J. Chai and A. Li. “Deep Learning in Natural Language Processing: A State-of-the-Art Survey”. In: *Proceedings - International Conference on Machine Learning and Cybernetics* 2019-July (July 2019). issn: 21601348. doi: 10.1109/ICMLC48188.2019.8949185.

[20] L. Tai et al. “A Survey of Deep Network Solutions for Learning Control in Robotics: From Reinforcement to Imitation”. In: (Dec. 2016). doi: 10.48550/arxiv.1612.07139. url: https://arxiv.org/abs/1612.07139v4.

[21] M. Längkvist, L. Karlsson, and A. Loutfi. “A review of unsupervised feature learning and deep learning for time-series modeling”. In: *Pattern Recognition Letters* 42 (1 June 2014), pp. 11–24. issn: 0167-8655. doi: 10.1016/J.PATREC.2014.01.008.

[22] L. Guasch et al. “Full-waveform inversion imaging of the human brain”. In: *npj Digital Medicine 2020 3:1* 3 (1 Mar. 2020), pp. 1–12. issn: 2398-6352. doi: 10.1038/s41746-020-0240-8. url: https://www.nature.com/articles/s41746-020-0240-8.

[23] Y. Zhang et al. “Neural network-based image reconstruction in swept-source optical coherence tomography using undersampled spectral data”. In: *Light: Science & Applications 2021 10:1* 10 (1 July 2021), pp. 1–14. issn: 2047-7538. doi: 10.1038/s41377-021-00594-7. url: https://www.nature.com/articles/s41377-021-00594-7.

[24] D. D. Carvalho et al. “Deep neural networks for plasma tomography with applications to JET and COMPASS”. In: *Journal of Instrumentation* 14 (09 Sept. 2019), p. C09011. issn: 1748-0221. doi: 10.1088/1748-0221/14/09/C09011. url: https://iopscience.iop.org/article/10.1088/1748-0221/14/09/C09011%20https://iopscience.iop.org/article/10.1088/1748-0221/14/09/C09011/meta.

[25] S. Silburn et al. “Calcam”. In: (Nov. 2018). doi: 10.5281/ZENODO.1478555. url: https://doi.org/10.5281/zenodo.1478555#.Ya5b2Uxt2fU.mendeley.

[26] J. Nocedal and S. J. Wright. *Line Search Methods*. 2006. doi: 10.1007/978-0-387-40065-5_3.

[27] C. O. Sorzano et al. “A Survey of the Use of Iterative Reconstruction Algorithms in Electron Microscopy”. In: *BioMed Research International* 2017 (2017). issn: 23146141. doi: 10.1155/2017/6482567. url: /pmc/articles/PMC5623807/%20/pmc/articles/PMC5623807/?report=abstract%20https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5623807/.

[28] P. Qiu et al. “A Numerical Study on Travel Time Based Hydraulic Tomography Using the SIRT Algorithm with Cimmino Iteration”. In: *Water 2019, Vol. 11, Page 909* 11 (5 Apr. 2019), p. 909. issn: 2073-4441. doi: 10.3390/W11050909. url: https://www.mdpi.com/2073-4441/11/5/909/htm%20https://www.mdpi.com/2073-4441/11/5/909.

[29] W. Guo and H. Chen. “Improving SIRT Algorithm for Computerized Tomographic Image Reconstruction”. In: *Lecture Notes in Electrical Engineering* 128 LNEE (VOL. 5 2012), pp. 523–528. issn: 18761119. doi: 10.1007/978-3-642-25792-6_79. url: https://link.springer.com/chapter/10.1007/978-3-642-25792-6_79.

[30] M. A. Brzostowski and G. A. McMechan. “3-D tomographic imaging of near-surface seismic velocity and attenuation”. In: *http://dx.doi.org/10.1190/1.1443254* 57 (3 Feb. 2012), pp. 396–403. issn: 00168033. doi: 10.1190/1.1443254. url: https://library.seg.org/doi/abs/10.1190/1.1443254.

[31] R. D. Radcliff and C. A. Balanis. “Reconstruction Algorithms for Geophysical Applications in Noisy Environments”. In: *Proceedings of the IEEE* 67 (7 1979), pp. 1060–1064. issn: 15582256. doi: 10.1109/PROC.1979.11389.

[32] J. Gregor and T. Benson. “Computational analysis and improvement of SIRT”. In: *IEEE Transactions on Medical Imaging* 27 (7 July 2008), pp. 918–924. issn: 02780062. doi: 10.1109/TMI.2008.923696.

[33] A. van der Sluis and H. A. van der Vorst. “SIRT- and CG-type methods for the iterative solution of sparse linear least-squares problems”. In: *Linear Algebra and its Applications* 130 (C Mar. 1990), pp. 257–303. issn: 0024-3795. doi: 10.1016/0024-3795(90)90215-X.

[34] M. Jiang and G. Wang. “Convergence studies on iterative algorithms for image reconstruction”. In: *IEEE Transactions on Medical Imaging* 22 (5 May 2003), pp. 569–579. issn: 02780062. doi: 10.1109/TMI.2003.812253.

[35] N. Parikh and S. Boyd. “Proximal Algorithms”. In: *Found. Trends Optim.* 1 (3 Jan. 2014), pp. 127–239. issn: 2167-3888. doi: 10.1561/2400000003. url: https://doi.org/10.1561/2400000003.

[36] A. Beck and M. Teboulle. “A fast iterative shrinkage-thresholding algorithm with application towavelet-based image deblurring”. In: *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings* (2009), pp. 693–696. issn: 15206149. doi: 10.1109/ICASSP.2009.4959678.

[37] S. W. Smith. *The Scientist and Engineer’s Guide to Digital Signal Processing*. Available at www.dspguide.com. California Technical Publishing, 1997. url: http://www.dspguide.com.

[38] R. Nevatia and K. R. Babu. “Linear feature extraction and description”. In: *Computer Graphics and Image Processing* 13 (3 July 1980), pp. 257–269. issn: 0146-664X. doi: 10.1016/0146-664X(80)90049-0.

[39] Z. Wang, W. Yan, and T. Oates. “Time Series Classification from Scratch with Deep Neural Networks: A Strong Baseline”. In: *Proceedings of the International Joint Conference on Neural Networks* 2017-May (Nov. 2016), pp. 1578–1585. doi: 10.1109/IJCNN.2017.7966039. url: https://arxiv.org/abs/1611.06455v4.

[40] A. Krizhevsky, I. Sutskever, and G. E. Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Advances in Neural Information Processing Systems* 25 (2012). url: http://code.google.com/p/cuda-convnet/.

[41] C. Szegedy et al. “Rethinking the Inception Architecture for Computer Vision”. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 2016-December (Dec. 2015), pp. 2818–2826. issn: 10636919. doi: 10.1109/CVPR.2016.308. url: https://arxiv.org/abs/1512.00567v3.

[42] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.

[43] A. Odena, V. Dumoulin, and C. Olah. “Deconvolution and Checkerboard Artifacts”. In: *Distill* (Oct. 2016). issn: 2476-0757. doi: 10.23915/DISTILL.00003. url: http://distill.pub/2016/deconv-checkerboard.

[44] S. Li et al. “Deep-Learning Inversion of Seismic Data”. In: *IEEE Transactions on Geoscience and Remote Sensing* 58 (3 Jan. 2019), pp. 2135–2149. doi: 10.1109/TGRS.2019.2953473. url: http://arxiv.org/abs/1901.07733%20http://dx.doi.org/10.1109/TGRS.2019.2953473.

[45] X. Hu et al. “RUNet: A robust UNet architecture for image super-resolution”. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops* 2019-June (June 2019), pp. 505–507. issn: 21607516. doi: 10.1109/CVPRW.2019.00073.

[46] V. Monga, Y. Li, and Y. C. Eldar. “Algorithm Unrolling: Interpretable, Efficient Deep Learning for Signal and Image Processing”. In: *IEEE Signal Processing Magazine* 38 (2 Dec. 2019), pp. 18–44. issn: 15580792. doi: 10.48550/arxiv.1912.10557. url: https://arxiv.org/abs/1912.10557v3.

[47] N. Chennakeshava et al. “High resolution plane wave compounding through deep proximal learning”. In: *IEEE International Ultrasonics Symposium, IUS* 2020-September (Sept. 2020). issn: 19485727. doi: 10.1109/IUS46767.2020.9251399.

[48] N. Chennakeshava et al. “Deep Proximal Learning for High-Resolution Plane Wave Compounding”. In: (Dec. 2021). doi: 10.48550/arxiv.2112.12410. url: https://arxiv.org/abs/2112.12410v1.

[49] A. Balatsoukas-Stimming and C. Studer. “Deep Unfolding for Communications Systems: A Survey and Some New Directions”. In: *IEEE Workshop on Signal Processing Systems, SiPS: Design and Implementation* 2019-October (Oct. 2019), pp. 266–271. issn: 15206130. doi: 10.1109/SIPS47522.2019.9020494.

[50] Z. Zhang and Y. Lin. “Data-Driven Seismic Waveform Inversion: A Study on the Robustness and Generalization”. In: *IEEE Transactions on Geoscience and Remote Sensing* 58 (10 Oct. 2020), pp. 6900–6913. issn: 15580644. doi: 10.1109/TGRS.2020.2977635.

[51] M. Mardani et al. “Neural Proximal Gradient Descent for Compressive Imaging”. In: *Advances in Neural Information Processing Systems* 2018-December (June 2018), pp. 9573–9583. issn: 10495258. doi: 10.48550/arxiv.1806.03963. url: https://arxiv.org/abs/1806.03963v1.

[52] X. Wei et al. “Deep Unfolding with Normalizing Flow Priors for Inverse Problems”. In: (July 2021). doi: 10.48550/arxiv.2107.02848. url: https://arxiv.org/abs/2107.02848v2.

[53] C. Mou, Q. Wang, and J. Zhang. “Deep Generalized Unfolding Networks for Image Restoration”. In: (Apr. 2022).

[54] M.-I. Georgescu, R. T. Ionescu, and N. Verga. “Convolutional Neural Networks With Intermediate Loss for 3D Super-Resolution of CT and MRI Scans”. In: *IEEE Access* 8 (2020), pp. 49112–49124. issn: 2169-3536. doi: 10.1109/ACCESS.2020.2980266.

[55] T. Wijkamp. “Characterization of 2D atomic and molecular emission processes in the MAST-U super-X divertor during detachment”. In: *Nuclear Fusion* (2022). To be submitted.

[56] G. Montavon, W. Samek, and K.-R. Müller. “Methods for interpreting and understanding deep neural networks”. In: *Digital Signal Processing* 73 (Feb. 2018), pp. 1–15. issn: 10512004. doi: 10.1016/j.dsp.2017.10.011.

[57] L. Martinelli. “Implementation of high-resolution spectroscopy for ion (and electron) temperature measurements of the TCV divertor plasma”. In: *Review of Scientific Instruments* (2022).

[58] M. Raissi, P. Perdikaris, and G. E. Karniadakis. “Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations”. In: *Journal of Computational Physics* 378 (2019), pp. 686–707.

[59] J. R. Chang et al. “One Network to Solve Them All — Solving Linear Inverse Problems Using Deep Projection Models”. In: IEEE, Oct. 2017, pp. 5889–5898. isbn: 978-1-5386-1032-9. doi: 10.1109/ICCV.2017.627.

[60] J. Zhai et al. “Autoencoder and Its Various Variants”. In: IEEE, Oct. 2018, pp. 415–419. isbn: 978-1-5386-6650-0. doi: 10.1109/SMC.2018.00080.

[61] O. Ronneberger, P. Fischer, and T. Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: (May 2015).

[62] M. Schoukens. “Improved Initialization of State-Space Artificial Neural Networks”. In: *2021 European Control Conference, ECC 2021* (2021), pp. 1913–1918. doi: 10.23919/ECC54610.2021.9655207.