Investigation of the solar atmosphere using machine learning techniques

Masterarbeit der Philosophisch-naturwissenschaftlichen Fakultät der Universität Bern

vorgelegt von

Daniel Zahnd

2024

Leiter der Arbeit

Prof. Dr. Lucia Kleint
Jonas Zbinden

Abstract

As a source of not only all energy needed on earth to enable plants to grow or weather phenomena to form, but also of inspiration and fascination, the sun and its underlying physics has ever been of utmost interest. There are certain solar phenomena such as solar flares or coronal mass ejections, which pose potential threats to technical endeavors on earth and also in space, such as power grids, spacecraft and ultimately humans serving as astronauts in space. Therefore, in analogy to meteorological forecasts on earth, there is a desire to be able to forecast solar flares and coronal mass ejections; this field of research is commonly called space weather. In order to make predictions of solar flares or coronal mass ejections, certain precursors in the solar atmosphere causally related or correlated to these phenomena have to be identified and studied. Aiming for this goal, inversions of so-called atmospheric models of stars are needed to obtain and investigate those physical parameters and to assess, whether some could be identified as precursors or not. Many stellar atmospheres are not invertible in an analytical way, which is to say that inversions of more complex stellar models are only possible by lots of computational power. Therefore, an alternative, in terms of computational power less demanding way of inverting stellar atmospheres is explored in this thesis. In the thesis at hand, normalizing flows are applied to observations of the Swedish Solar Telescope (SST) in combination with the Milne-Eddington model as the sun's atmospheric model.

Contents

Chapter 1

Introduction

For many people, looking up at the night sky is a source of utmost excitation and inspiration. By naked eye, the most prominent objects in the night sky are of course the moon, the planets of our solar system and many stars. The most interesting star for us humans is the Sun, because it drives and powers our everyday life. Without it, there would be no growth of plants and there would ultimately be no life, as we know it, possible on Earth. Apart from mere satisfaction of human curiosity and fascination, understanding stellar and especially solar physics is also key to many viable applications for mankind. The field of space weather is concerned with understanding and predicting solar eruptions, so-called solar flares, which according to [Stix, 2002, p.436] can, but do not have to be correlated with coronal mass ejections.

1.1 The Sun

Among the many characteristics of the Sun, there are at least five phenomenological properties as visualized in fig. 1.1; these include its mass $M_\odot = 1.989 \times 10^{30}\,\mathrm{kg}$, its radius $R_\odot = 696\,340\,\mathrm{km}$, its luminosity

$$L_\odot = 4\pi R_\odot^2 \tilde{\sigma} T_\odot^4 = 3.846 \times 10^{26}\,\mathrm{W}, \tag{1.1}$$

its effective temperature $T_\odot \approx 5800\,\mathrm{K}$ and its distance to Earth $d_{\odot,E} \approx 150 \times 10^6\,\mathrm{km}$. Thereby, $\tilde{\sigma} = 5.670 \times 10^{-8}\,\mathrm{Wm^{-2}K^{-4}}$ is the Stefan-Boltzmann constant. Comparing these characteristics of the Sun to those of Earth with mass $M_E = 5.972 \times 10^{24}\,\mathrm{kg}$, radius $R_E = 6341\,\mathrm{km}$ and distance to the Sun $d_{\odot,E}$, one can see that the Sun is approximately 0.3 million times more massive and roughly 1.3 million times more voluminous than Earth.

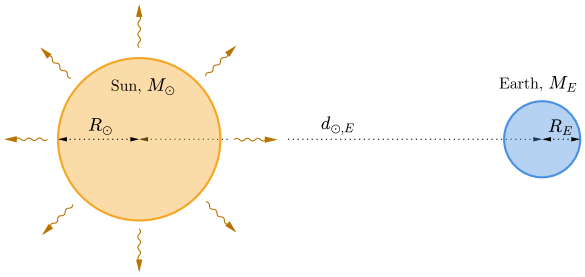


Figure 1.1: Visualization of the basic characteristics of the Sun and Earth.

The outer layers of the Sun can be roughly divided into three regions, the photosphere, the chromosphere and the corona. The photosphere is the innermost of these outer layers. Acoording to [Weigert et al., 2006, p.135], its depth is determined by the optical depth being such, that photons from below the photosphere cannot escape into space anymore; this is to say, that all observable photons from outside the Sun originate at or above the photosphere, which is its namegiving property. Above the photosphere, the chromosphere is located Since the density is very low in this layer, there is not much contribution to the overall emitted energy of the Sun coming from this layer despite its high temperature of up to several 10^5 K. The outermost layer of the solar atmosphere is the corona with even higher temperatures. The identification of the heating mechanism leading to these high temperatures in the solar corona is still one of the major open questions of stellar physics, known as the coronal heating problem. The boundary of the corona to interplanetary space is of a continuous nature.

Those characteristics alone however provide little insight to the formation of Sunspots, solar flares or coronal mass ejections. In order to gain insight to the physics relevant for those phenomena, a detailed theory of radiative transfer is necessary, which then provides a useful tool to extract physical parameters from solar observations of a certain region of the solar atmosphere. Analyzing those parameters for so-called active regions of the Sun - these are regions, where Sunspots, solar flares and coronal mass ejections form - provides insight to the physical processes going on during these phenomena.

1.1.1 Sunspots

Apart from the granulated surface of the Sun, there sometimes are certain dark spots. These dark spots are identified either as Sunspots or

pores. Both phenomena are visible in fig. 1.2. Sunspots consist of a dark

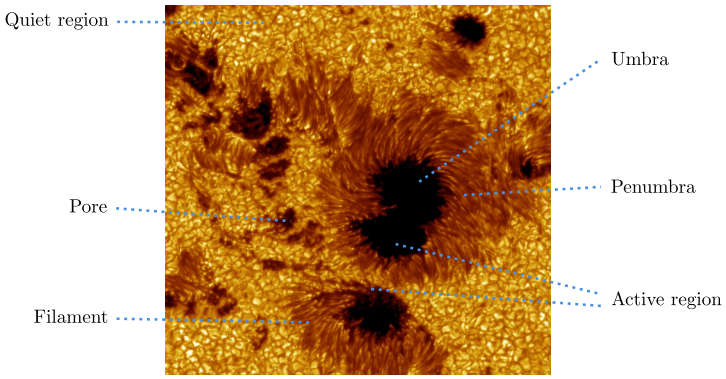


Figure 1.2: Sunspots in the active region 10030, observed on July 15, 2002. Image credit: NASA Goddard Space Flight Center, `https://www.flickr.com/photos/gsfc/5510488494`, accessed on November 07, 2023. Adapted by the author.

region, called the umbra, which is surrounded by a brighter region exhibiting a filament-like structure, known as the penumbra. Pores however just consist of an umbra without a penumbra. The filaments constituting the penumbra are usually approximately radially aligned around the Sunspot.

In fig. 1.3, a schematic representation of a Sunspot is shown. Sunspots form below the photosphere and then rise to the latter due to buoyancy. The relative darkness of the umbra within a Sunspot is due to lower tem-

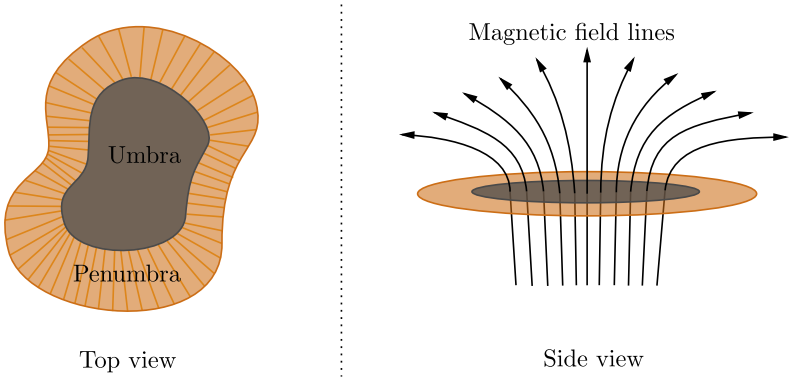


Figure 1.3: Schematic visualization of a Sunspot. Note, that areas on the Sun which show Sunspots and/or pores are called active regions, whereas other realms on the Sun's surface are called quiet regions. Note, that the magnetic field lines as drawn here could also be oriented exactly opposite, i.e. facing towards the Sun's interior.

peratures compared to the undisturbed photosphere. [Weigert et al., 2006,

p.142] suggest, that the explanation for the temperature difference is found in strong magnetic fields penetrating the photosphere in the region of the umbra and partially of the penumbra, as it is seen in the right panel of fig. 1.3. As known from classical electrodynamics, Maxwell's equations state that there are no magnetic monopoles, i.e. $\boldsymbol{\nabla} \cdot \boldsymbol{B} = 0$ for any magnetic field $\boldsymbol{B}$. Hence, [Stix, 2002, p.342] mentions, that if there is a magnetic field penetrating the photosphere from below to above, it must again penetrate from above to below, since magnetic field lines always have to be loops. This gives rise to the so-called bipolar structure/arrangement of Sunspots, which is indeed characteristic to many observed active regions containing Sunspots. However, [Stix, 2002, p.342] states, that a bipolar structure is not necessary to observed Sunspots; this is because sometimes the emerging magnetic flux is more concentrated where it leaves the photosphere than where it reenters; thus leading to only one visible Sunspot formed. A bipolar structure of Sunspots is schematically shown in fig. 1.4. Sunspots are particularly interesting for space

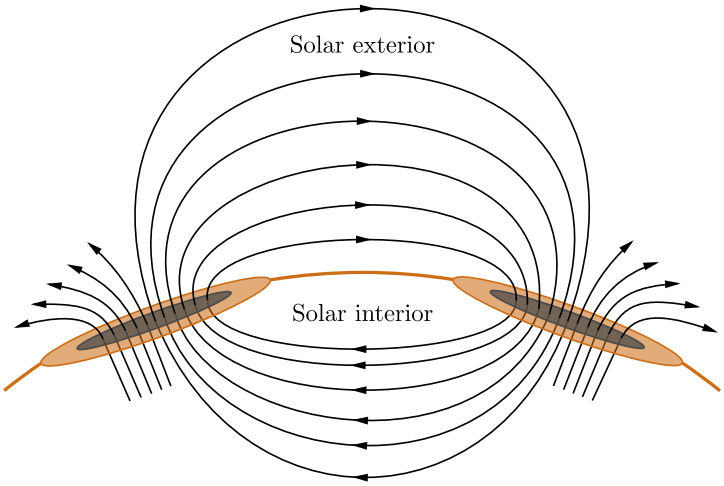


Figure 1.4: Schematic representation of a bipolar Sunspot arrangement. Note, that the magnetic field lines as drawn here could also be oriented exactly opposite.

weather, because they are correlated with eruptive events of the Sun, such as solar flares or coronal mass ejections. Both solar flares and coronal mass ejections belong to the category of solar eruptions, which are are tied to instabilities in the magnetic field configuration within the solar atmosphere.

1.1.2 Solar flares

As [Stix, 2002, p.432] remarks, solar flares originate around Sunspots. A solar flare commonly lasts for about 30 minutes to 1 hour in total, where the first several minutes are dedicated to a fast rise in brightness of the solar flare. The brightness then slowly decreases for the remaining active time of the solar flare. A solar flare essentially is the ejection of a large packet of energy in the form of high-energy photons and some particle radiation in the form of electrons and protons. The electromagnetic radiation caused by a solar flare travels at the speed of light c, since photons are massless.

1.1.3 Coronal mass ejections

The term coronal mass ejection refers to a phenomenon, where the Sun ejects large amounts of mass into interplanetary space. As [Stix, 2002, p.436] points out, the ejected mass does not necessarily have to originate in the solar corona, as the term could misleadingly suggest; the term reflects the fact, that the mass is seen to be ejected by the corona. Often, coronal mass ejection follows a solar flare, but this does not need to be the case. A coronal mass ejection can be described as the sudden ejection of large amounts of energy by the Sun in the form of particle radiation (mass). Since these particles do have a mass, this particle radiation cannot travel at the speed of light.

1.2 Why bother about stellar atmosphere inversions?

The ultimate goal of space weather is to identify the triggers of solar flares and coronal mass ejections, since they pose potential threats to power grids on Earth, spacecraft and possibly astronauts themselves. In order to look for possible precursors causally related to flaring activity or coronal mass ejections of the Sun, the physical parameters describing the conditions in the Sun's atmosphere need to be known, e.g. the magnetic field strength, its direction, the temperature, pressure and so on. Since in-situ measurement is not possible on the Sun because of its extreme environment, those parameters need to be inferred by means of remote sensing techniques. Such techniques involve measuring Stokes profiles, which can be turned into atmospheric parameters by means of an inversion of a stellar atmospheric model. Those models encompass the

equations of radiative transfer; inversions of those models usually cannot be inverted analytically, depending on their degree of complexity. But there are some models, which can be inverted analytically, such as the Milne-Eddington approximation studied in this thesis.

Notation

At this point, a brief remark about the notation convention used in the following material is in order. Scalar quantities are denoted by an normal mathematics letter Q. For vectorial quantities, a bold mathematics letter $\boldsymbol{Q}$ is used. Finally, for tensorial quantities such as a matrix, a non-italic bold mathematics letter $\mathbf{Q}$ is made use of.

Chapter 2

Solar physics

All stars, and herewith also the Sun, are sources of gigantic amounts of energy. This energy is set free by means of atomic fusion processes happening within the star and is conveyed to an observer through electromagnetic or particle radiation. In order to infer properties of radiation source, e.g. a star, the properties of the emitted radiation need to be observed and analyzed using the theoretical framework of physics.

In the case of solar atmosphere research, there are three main tools for investigation available to the physicist; first, the theory of radiative transfer, which is concerned with the propagation of electromagnetic radiation in some medium. Second, there is spectroscopy, which is about how electromagnetic radiation interacts with matter as a function of wavelength. Third, polarimetry is necessary, which addresses how and why electromagnetic radiation is polarized as it is.

2.1 Radiative transfer

Because electromagnetic radiation is the dominant radiation source in solar atmosphere research, radiative transfer is integral to investigating the solar atmosphere.

2.1.1 Quantities in radiative transfer

In the theory of radiative transfer, the definition of several measurable quantities is necessary. If some quantity is a called a spectral quantity, this means that the quantity is measured at a certain wavelength λ or frequency ν; spectral quantity will be denoted by a subscript λ or ν.

The radiant energy E with units $[E] = \mathrm{J}$ is defined as the energy

radiated away from some source. The radiant flux Φ with units $[\Phi] = \mathrm{Js}^{-1} = \mathrm{W}$ however is defined as the radiant energy per unit time and is equal to the spectral flux integrated over all wavelengths. Furthermore, the spectral flux Φ_λ having units $[\Phi_\lambda] = \mathrm{Wm}^{-1}$ is the radiant flux per unit wavelength. Additionally, the radiance I with units $[I] = \mathrm{Wsr}^{-1}\mathrm{m}^{-2}$ is the radiant flux per unit solid angle and per unit projected area; similarly the spectral radiance I_λ with units $\mathrm{Wsr}^{-1}\mathrm{m}^{-3}$ is the radiance per unit wavelength. Finally, the irradiance F with units $[F] = \mathrm{Wm}^{-2}$ is given by the radiant flux per unit projected area; it is also known as the flux density and is equal to the radiance integrated over all solid angles of a hemisphere. Associated to the radiance, there is also the spectral irradiance F_λ, with is nothing but the radiance per unit wavelength and therefore with units $[F_\lambda] = \mathrm{Wm}^{-3}$.

2.1.2 Thermal and thermodynamic equilibrium

The thermal equilibrium (TE) is defined as a state of a system, such that there are no macroscopic flows of heat. That is the case, whenever the temperature T is homogeneously constant throughout the system under consideration.

The thermodynamic equilibrium however is defined as a state of a system, in which there are no macroscopic flows of matter in addition to the thermal equilibrium condition.

Furthermore, there is the concept of local thermal equilibrium (LTE). A local thermal equilibrium is defined as a thermal equilibrium confined to local space, whereby local space refers to a parcel consisting of significantly more than one molecule, but much smaller in extent than the global system.

A system in local thermal equilibrium implies that a parcel within the considered system can locally be described as being in the state of a thermal equilibrium, which is to say, that Planck's law holds for this considered parcel.

2.1.3 Planck's law

The Planck law can be derived considering the physics of thermodynamics under the assumption of a thermal equilibrium for the considered system. It describes the spectral radiance emitted by a blackbody; that is some object, that perfectly absorbs and emits radiation of all wavelengths, and is at a thermal equilibrium with its environment. The Planck

law in both the frequency and wavelength domains is given by

$$B_\nu(T)\,\mathrm{d}\nu = \frac{2h\nu^3}{c^2}\frac{1}{e^{\frac{h\nu}{k_BT}}-1}\,\mathrm{d}\nu, \quad B_\lambda(T)\,\mathrm{d}\lambda = \frac{2hc^2}{\lambda^5}\frac{1}{e^{\frac{hc}{\lambda k_BT}}-1}\,\mathrm{d}\lambda, \quad (2.1)$$

where B_ν and B_λ are spectral radiances with units $[B_\nu] = \mathrm{Wm^{-2}sr^{-1}Hz^{-1}}$ and $[B_\lambda] = \mathrm{Wm^{-3}sr^{-1}}$.

2.1.4 Blackbodies and Kirchhoff's law

A blackbody is a hypothetical object, that absorbs all radiation at all wavelengths completely. As a result, this body will at some point reach a thermal equilibrium with its environment, since absorption of radiation leads to a warming of the body which causes emission of thermal radiation and hence loss of energy and therefore heat. Eventually, absorbed and emitted energy will be balanced and thermal equilibrium is reached; in this state, the blackbody emits and absorbs radiation isotropically and exactly according to the Planck formula $B_\lambda(T)$[1].

Let the emissivity ε_λ and absorptivity A_λ be defined as

$$\varepsilon_\lambda(T) \doteq \frac{\mathcal{E}_\lambda(T)}{B_\lambda(T)}, \qquad A_\lambda \doteq \frac{\mathcal{A}_\lambda(T)}{B_\lambda(T)}, \quad (2.2)$$

where $\mathcal{E}_\lambda(T)$ is the emittive spectral radiance and $\mathcal{A}_\lambda(T)$ is the absorptive spectral radiance of a body at a given temperature T. In thermal equilibrium, the emittive spectral radiance must be equal to the absorptive spectral radiance, for if this would not be the case, macroscopic flow of heat would result. From this consideration, Kirchhoff's law follows as

$$\mathcal{E}_\lambda(T) = \mathcal{A}_\lambda(T) \quad \Leftrightarrow \quad \varepsilon_\lambda(T) = A_\lambda(T), \quad (2.3)$$

which in words states, that the emissivity is equal to the absorptivity at every particular wavelength λ for a system in thermal equilibrium at temperature T.

For a blackbody in thermal equilibrium, the absorptive spectral radiance is maximal, which is to say that $\mathcal{A}_\lambda(T) = B_\lambda(T)$ holds, leading to $A_\lambda(T) = 1$. Since the blackbody is in thermal equilibrium, the absorptive spectral radiance is equal to the emissive spectral radiance and hence $\mathcal{E}_\lambda(T) = B_\lambda(T)$ must hold, wherefrom $\varepsilon_\lambda(T) = 1$ follows. Therefore, a

[1] Using the transformation relation $\nu = \frac{c}{\lambda}$ and $\int_0^\infty B_\lambda\,\mathrm{d}\lambda \overset{!}{=} \int_0^\infty B_\nu\,\mathrm{d}\nu$, one can go back and forth between the formulation of Planck's law in the wavelength or frequency domain.

blackbody in thermal equilibrium can be described mathematically by means of the relation

$$\varepsilon_\lambda(T) = A_\lambda(T) = 1. \tag{2.4}$$

2.1.5 Rayleigh-Jeans and Wien approximations

Consider the Planck law eq. (2.1). The Rayleigh-Jeans law is the Planck law for the limit of low energies, whereas the Wien law is the Planck law for the limit of high energies. For low energies $h\nu/k_BT \to 0$, the Rayleigh-Jeans law

$$B_\nu(T)\,\mathrm{d}\nu = \frac{2\nu^2}{c^2} k_B T\,\mathrm{d}\nu, \quad \frac{h\nu}{k_B T} \to 0 \tag{2.5}$$

is obtained, whereas for high energies $\nu \to \infty$, the Wien law

$$B_\nu(T)\,\mathrm{d}\nu = \frac{2h\nu^3}{c^2} e^{-\frac{h\nu}{k_B T}}\,\mathrm{d}\nu, \quad \nu \to \infty \tag{2.6}$$

follows.

In order to derive the Rayleigh-Jeans law from the Planck law, one has to do the substitution $x = h\nu/k_BT$ in the Planck law and perform a Taylor expansion of the resulting expression around $x_0 = 0$. This corresponds to the case $h\nu/k_BT \to 0$.

Moreover, to arrive at the Wien law, the substitution $y = e^{-h\nu/k_BT}$ in the Planck law has to be made and a Taylor expansion of the resulting expression around $y_0 = 0$ has to be performed. This corresponds to the case $\nu \to \infty$.

2.1.6 Stefan-Boltzmann law

Consider some isotropic spectral radiance I_λ along a ray, flowing through a surface A as depicted in fig. 2.1. If one calculates the flux of energy per wavelength through the surface A, one gets a quantity known as the spectral irradiance F_λ, also known as spectral flux density. This can be done by integrating the normal component $I_{\lambda,n} = I_\lambda \cos(\theta)$ of I_λ over the whole half-sphere, that is over the parameter space $\theta \in [0, \pi/2]$ and $\varphi \in [0, 2\pi]$. Furthermore, the total flux F can be obtained from F_λ by integrating over all wavelengths. Therefore, the irradiance (flux density) F of isotropic electromagnetic radiation can be calculated by means of the integral

$$F = \int_0^{2\pi} \mathrm{d}\varphi \int_0^{\pi/2} \sin(\theta)\cos(\theta)\,\mathrm{d}\theta \int_0^\infty I_\lambda\,\mathrm{d}\lambda = \pi \int_0^\infty I_\lambda\,\mathrm{d}\lambda, \tag{2.7}$$

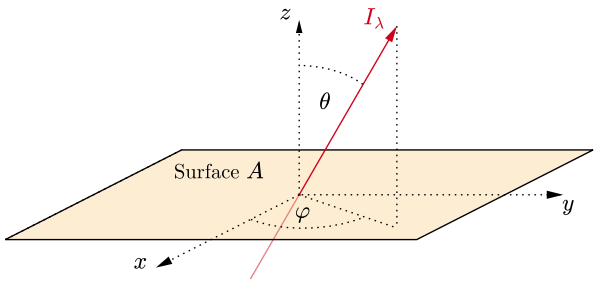


Figure 2.1: Visualization for a derivation of the Stefan-Boltzmann law.

where the factor $\sin(\theta)$ originates in the transformation from cartesian to polar coordinates. If one now assumes, that I_λ describes a system in thermal equilibrium, $I_\lambda = B_\lambda(T)$ follows and hence the Stefan-Boltzmann law

$$F = \tilde{\sigma} T^4 \tag{2.8}$$

for a system in thermal equilibrium.

The Stefan-Boltzmann law is a useful means to determine the effective temperature of stars, such as that of the Sun $T_\odot$. Measuring the solar constant $S_\odot \approx 1361\,\mathrm{Wm}^{-2}$, i.e. how much solar power per unit square the Earth receives on average, one can find out the effective temperature $T_\odot$ of the Sun by means of relating the two quantities as

$$S_\odot = \frac{L_\odot}{4\pi d_{\odot,E}^2} = \tilde{\sigma} T_\odot^4 \left(\frac{R_\odot}{d_{\odot,E}}\right)^2 \quad \Rightarrow \quad T_\odot = \left(\frac{S_\odot}{\tilde{\sigma}}\right)^{1/4} \left(\frac{d_{\odot,E}}{R_\odot}\right)^{1/2}, \tag{2.9}$$

which gives $T_\odot = 5777\,\mathrm{K} \approx 5800\,\mathrm{K}$ as stated in the introduction.

2.1.7 Radiative transfer equation

The radiative transfer equation is usually given in terms of spectral irradiance I_λ. The basic two forms of the radiative transfer equation shall now be briefly derived, following the material in [Rutten, 2015, pp.27-40].

Consider as a first step fig. 2.2. A beam of spectral irradiance I_λ passes through a slab of matter with thickness $\mathrm{d}z$, along a path s with length $\mathrm{d}s$. The question now is, how the spectral irradiance I_λ after the slab can be calculated. In order to quantify the spectral irradiance after passing through the slab of matter, two contributions due to electromagnetic interaction between radiation and matter within the slab have to be considered, namely extinction and emission. Extinction is loss of radiative energy along a path s due to scattering or absorption and it is accounted

for by the extinction coefficient α_λ with units $[\alpha_\lambda] = \mathrm{m}^{-1}$. Emission is gain of radiative energy along a path s due to spontaneous emission of photons caused by decay of excited states in molecules or atoms within the slab of matter; it is quantified by the emission coefficient j_λ having units $[j_\lambda] = \mathrm{Wsr}^{-1}\mathrm{m}^{-4}$.

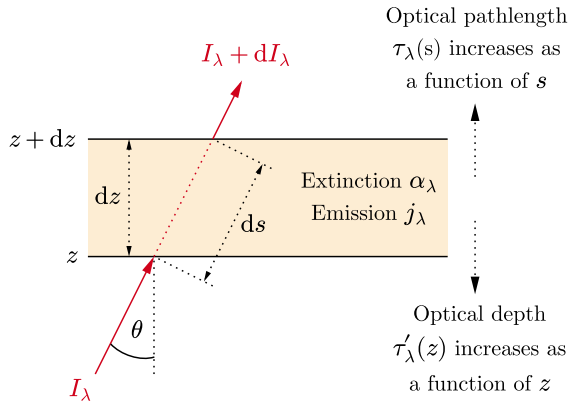


Figure 2.2: Illustration for the derivation of the radiative transfer equation. $[I_\lambda] = \mathrm{W\,sr^{-1}\,m^{-3}}$ denotes the spectral radiance measured along a ray of inclination θ with respect to a horizontal plane in the coordinate system under consideration.

The difference $\mathrm{d}I_\lambda$ in spectral irradiance along the path of propagation must be the sum of the extinction and emission contributions. Considering the units of the extinction and emission coefficients, these contributions can be written as $-\alpha_\lambda I_\lambda \,\mathrm{d}s$ for the loss due to extinction and as $j_\lambda \,\mathrm{d}s$ for the gain due to emission. Herewith, one arrives at

$$\mathrm{d}I_\lambda = j_\lambda \,\mathrm{d}s - \alpha_\lambda I_\lambda \,\mathrm{d}s \quad \Leftrightarrow \quad \frac{\mathrm{d}I_\lambda}{\mathrm{d}s} = j_\lambda - \alpha_\lambda I_\lambda. \tag{2.10}$$

Defining the optical pathlength τ_λ by means of $\mathrm{d}\tau_\lambda \doteq \alpha_\lambda \,\mathrm{d}s$ and furthermore the so-called source function S_λ as $S_\lambda \doteq \frac{j_\lambda}{\alpha_\lambda}$, this equation can also be written as

$$\frac{\mathrm{d}I_\lambda(\tau_\lambda)}{\mathrm{d}\tau_\lambda} = S_\lambda(\tau_\lambda) - I_\lambda(\tau_\lambda), \tag{2.11}$$

which is called the transport equation in differential form. Defining the optical depth τ'_λ as $\mathrm{d}\tau'_\lambda \doteq \alpha_\lambda \,\mathrm{d}z = -\alpha_\lambda \cos(\theta)\,\mathrm{d}s = -\cos(\theta)\,\mathrm{d}\tau_\lambda$, the transport equation can also be written in terms of optical depth rather than optical pathlength, namely

$$\cos(\theta)\frac{\mathrm{d}I_\lambda(\tau'_\lambda)}{\mathrm{d}\tau'_\lambda} = I_\lambda(\tau'_\lambda) - S_\lambda(\tau'_\lambda). \tag{2.12}$$

The change in sign is due to the fact, that the optical depth τ'_λ is measured from the point of view of an observer looking vertically upon the matter, from which the radiation is emerging from, while the optical pathlength τ_λ measures the distance of propagation in the direction of the ray. Adopting the boundary conditions $\tau_\lambda(0) = \tau'_\lambda(0) = 0$, the optical pathlength $\tau_\lambda(S)$ at a position $s = S$ aswell as the optical depth $\tau_\lambda(Z)$ at the position $z = Z$ are e given by integration as

$$\tau_\lambda(S) = \int_{\tau_\lambda(0)}^{\tau_\lambda(S)} \mathrm{d}\tau_\lambda = \int_0^S \alpha_\lambda(s)\,\mathrm{d}s \tag{2.13}$$

and

$$\tau'_\lambda(Z) = \int_{\tau'_\lambda(0)}^{\tau'_\lambda(Z)} \mathrm{d}\tau'_\lambda = \int_0^Z \alpha_\lambda(z)\,\mathrm{d}z. \tag{2.14}$$

Formal solution of the radiative transfer equation

Both of the above formulations for the transfer equation in differential form can also be written in an integral form, as it is demonstrated as follows; consider for that matter fig. 2.3.

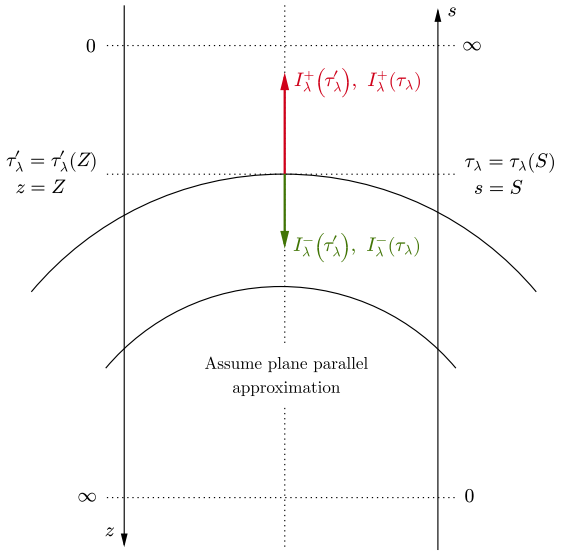


Figure 2.3: Illustration of coordinate systems for optical pathlength τ_λ parametrized by s and optical depth τ'_λ parametrized by z. Note, that the optical pathlength axis does not necessarily need to be parallel to the optical depth axis, the factor $\cos(\theta)$ corrects for the case where these axes are not parallel. The curved lines represent layers of a stellar atmosphere, which are locally assumed as plane parallel.

In order to derive integral forms of the transfer equation with respect to the optical pathlength τ_λ, the derivative

$$\frac{\mathrm{d}}{\mathrm{d}\tau_\lambda}\left[e^{\tau_\lambda} I_\lambda(\tau_\lambda)\right] = e^{\tau_\lambda}\left[I_\lambda(\tau_\lambda) + \frac{\mathrm{d}I_\lambda(\tau_\lambda)}{\mathrm{d}\tau_\lambda}\right] = e^{\tau_\lambda} S_\lambda(\tau_\lambda) \tag{2.15}$$

is first calculated. Finding the outward spectral radiance $I_\lambda^+(\tau_\lambda)$ at a given optical pathlength τ_λ requires integration of eq. (2.15) from 0 to τ_λ, namely

$$\int_0^{\tau_\lambda} \frac{\mathrm{d}}{\mathrm{d}t}\left[e^t I_\lambda^+(t)\right]\,\mathrm{d}t = e^{\tau_\lambda} I_\lambda^+(\tau_\lambda) - I_\lambda^+(0) = \int_0^{\tau_\lambda} e^t S_\lambda(t)\,\mathrm{d}t. \tag{2.16}$$

From this expression it follows by multiplication of the above equation with $e^{-\tau_\lambda}$ and rearrangement of terms, that the outward spectral radiance at a certain optical pathlength τ_λ from the origin of the coordinate system under consideration is given by

$$I_\lambda^+(\tau_\lambda) = I_\lambda^+(0)e^{-\tau_\lambda} + \int_0^{\tau_\lambda} e^{t-\tau_\lambda} S_\lambda(t)\,\mathrm{d}t, \tag{2.17}$$

which is what is commonly called the formal solution of the transfer equation with respect to optical pathlength. If one wants to calculate the inward spectral radiance $I_\lambda^-(\tau_\lambda)$ however, integration of eq. (2.15) is required to be carried from ∞ to τ_λ, such that

$$-\int_{\tau_\lambda}^{\infty} \frac{\mathrm{d}}{\mathrm{d}t}\left[e^t I_\lambda^-(t)\right]\,\mathrm{d}t = e^{\tau_\lambda} I_\lambda^-(\tau_\lambda) = -\int_{\tau_\lambda}^{\infty} e^t S_\lambda(t)\,\mathrm{d}t \tag{2.18}$$

results. Multiplying this equation by $e^{-\tau_\lambda}$ results in the inward spectral radiance at a given optical pathlength τ_λ is calculable as

$$I_\lambda^-(\tau_\lambda) = -\int_{\tau_\lambda}^{\infty} e^{t-\tau_\lambda} S_\lambda(t)\,\mathrm{d}t. \tag{2.19}$$

For both formulations of $I_\lambda^+(\tau_\lambda)$ and $I_\lambda^-(\tau_\lambda)$ the boundary conditions

$$I_\lambda^-(\infty) = 0, \quad e^t S_\lambda(t) \xrightarrow{t\to\infty} 0 \tag{2.20}$$

were used, which mean that at the position $\tau_\lambda = \infty$ there is no radiation directed towards the source moreover that the source function $S_\lambda(t)$ decreases faster with increasing optical pathlength than e^{-t}.

The integral representations of the transfer equation with respect to the optical depth τ'_λ can also be obtained in a similar way; for this purpose, the derivative

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}\tau'_\lambda}\left[\cos(\theta)e^{-\frac{\tau'_\lambda}{\cos(\theta)}}I_\lambda(\tau'_\lambda)\right] &= e^{-\frac{\tau'_\lambda}{\cos(\theta)}}\left[\cos(\theta)\frac{\mathrm{d}I_\lambda(\tau'_\lambda)}{\mathrm{d}\tau'_\lambda} - I_\lambda(\tau'_\lambda)\right] \\ &= -e^{-\frac{\tau'_\lambda}{\cos(\theta)}}S_\lambda(\tau'_\lambda)\end{aligned} \tag{2.21}$$

is needed. Obtaining an expression for the outward spectral radiance $I^+_\lambda(\tau'_\lambda)$ at a given optical depth τ'_λ requires integration of eq. (2.21) from τ'_λ to ∞, namely

$$\begin{aligned}\int_{\tau'_\lambda}^{\infty}\frac{\mathrm{d}}{\mathrm{d}t}\left[\cos(\theta)e^{\frac{t}{\cos(\theta)}}I^+_\lambda(t)\right]\,\mathrm{d}t &= -\cos(\theta)e^{-\frac{\tau'_\lambda}{\cos(\theta)}}I^+_\lambda(\tau'_\lambda) \\ &= -\int_{\tau'_\lambda}^{\infty}e^{-\frac{t}{\cos(\theta)}}S_\lambda(t)\,\mathrm{d}t.\end{aligned} \tag{2.22}$$

Dividing by $\cos(\theta)$ and multiplying by $e^{\frac{\tau'_\lambda}{\cos(\theta)}}$ yields

$$I^+_\lambda(\tau'_\lambda,\theta) = \frac{1}{\cos(\theta)}\int_{\tau'_\lambda}^{\infty}e^{\frac{\tau'_\lambda - t}{\cos(\theta)}}S_\lambda(t)\,\mathrm{d}t \tag{2.23}$$

as the expression for the outward spectral radiance at a given optical depth τ'_λ. Moreover, if the inward spectral radiance $I^-_\lambda(\tau'_\lambda)$ is to be calculated, integration of eq. (2.21) is required from τ'_λ to zero. Written down, this amounts to

$$\begin{aligned}-\int_0^{\tau'_\lambda}\frac{\mathrm{d}}{\mathrm{d}t}\left[e^{-\cos(\theta)\frac{t}{\cos(\theta)}}I^-_\lambda(t)\right] &= -\cos(\theta)e^{-\frac{\tau'_\lambda}{\cos(\theta)}}I^-_\lambda(\tau'_\lambda) \\ &= \int_0^{\tau'_\lambda}e^{-\frac{t}{\cos(\theta)}}S_\lambda(t)\,\mathrm{d}t.\end{aligned} \tag{2.24}$$

Finally, dividing again by $-\cos(\theta)$ and multiplying by $e^{\frac{\tau'_\lambda}{\cos(\theta)}}$ leads to the inward spectral radiance at a given optical depth τ'_λ as

$$I^-_\lambda(\tau'_\lambda,\theta) = -\frac{1}{\cos(\theta)}\int_0^{\tau'_\lambda}e^{\frac{\tau'_\lambda - t}{\cos(\theta)}}S_\lambda(t)\,\mathrm{d}t. \tag{2.25}$$

Again, for both expressions $I^+_\lambda(\tau'_\lambda,\theta)$ and $I^-_\lambda(\tau'_\lambda,\theta)$ boundary conditions had to be applied, namely

$$I^-_\lambda(0,\theta) = 0, \quad e^{-t}S_\lambda(t) \xrightarrow{t\to\infty} 0. \tag{2.26}$$

These conditions mean, that at the location $\tau'_\lambda = 0$, there is no radiation directed towards the source; furthermore the source function $S_\lambda(t)$ is required not grow faster than e^t as optical depth τ'_λ increases.

2.1.8 Solutions for a homogeneous medium

In the case, where radiative transfer through a homogeneous material is considered, the extinction coefficient α_λ aswell as the emission coefficient j_λ are constant everywhere in the material. Therefore, the source function does not depend on any spatial parameter either.

Taking the optical pathlength τ_λ as the spatial parameter, the outward spectral radiance $I_\lambda^+(\tau_\lambda)$ can be calculated using eq. (2.17) and a constant source function $S_\lambda(\tau_\lambda) = \text{const}$. Since in that case the source function does not integrate with respect to τ_λ, performing the integration yields

$$\begin{aligned} I_\lambda^+(\tau_\lambda) &= I_\lambda^+(0)e^{-\tau_\lambda} + S_\lambda \int_0^{\tau_\lambda} e^{t-\tau_\lambda}\,\mathrm{d}t \\ &= I_\lambda^+(0)e^{-\tau_\lambda} + S_\lambda \left(1 - e^{-\tau_\lambda}\right), \end{aligned} \tag{2.27}$$

which is the solution of the radiative transfer equation with respect to optical pathlength τ_λ in the case of a homogeneous medium. There are two approximations for this solution, namely for an optically thick or thin material.

The first approximation for an *optically thick* material determined by the requirement $\tau_\lambda \gg 1$, which is to say, that the optical pathlength through the medium is very long. In this case, the exponential factors $e^{-\tau_\lambda}$ approach zero, as $\lim_{\tau_\lambda \to \infty} e^{-\tau_\lambda} = 0$ holds. This is to say, that the solution to the radiative transfer equation in the optically thick case is given by

$$I_\lambda^+(\tau_\lambda) \approx S_\lambda, \quad \tau_\lambda \gg 1, \quad S_\lambda = \text{const.} \tag{2.28}$$

Note, that the outward spectral radiance in this case does not vary with optical pathlength τ_λ.

The second approximation for an *optically thin* material is given by the condition $\tau_\lambda \ll 1$. Using a Taylor series expansion of $e^{-\tau_\lambda}$ up to first order in τ_λ for $\tau_\lambda \ll 1$ yields $e^{-\tau_\lambda} \approx 1 - \tau_\lambda$, in the case where the optical pathlength τ_λ is indeed very short. Inserting this expression into eq. (2.27), the solution to the radiative transfer equation in the optically thin case is accounted for by

$$I_\lambda^+(\tau_\lambda) \approx I_\lambda^+(0) + \left[S_\lambda - I_\lambda^+(0)\right]\tau_\lambda \quad \tau_\lambda \ll 1, \quad S_\lambda = \text{const.} \tag{2.29}$$

Note, that the outward spectral radiance in this case varies linearly with optical pathlength τ_λ.

2.1.9 The Eddington-Barbier approximation

The Eddington-Barbier approximation is a more realistic simplification of the radiative tranfer equation than the assumption of a homogeneous medium, for it requires a source function S_λ that varies as a power series in τ'_λ rather than being constant throughout the medium. In order to derive the exact form of this approximation, optical depth τ'_λ is used rather than optical pathlength τ_λ, because in this way one arrives at an expression for the emergent spectral radiance, which is an actually measurable quantity, rather than obtaining an expression for the spectral radiance at an invisible layer within the medium. The optical depth τ'_λ at a coordinate of $z = Z$ is calculated by means of integrating the extinction coefficient $\alpha_\lambda(z)$ as $\tau'_\lambda(Z) = \int_0^Z \alpha_\lambda(z)\,\mathrm{d}z$, where the position $z = 0$ refers to the outermost layer of the considered medium - e.g. the position of an observer and $z = \infty$ denotes the innermost layer of said medium; consider fig. 2.3 for a visual clarification of that matter.

Now, the outward (emergent) spectral radiance at a position $z = Z$ and a local angle of θ with respect to the normal vector of the plane parallel medium layers is accounted for by eq. (2.23). Assuming that a considered source function $S_\lambda(\tau'_\lambda)$ satisfies the necessary requirements to be written as a power series, one can write down such an expansion in powers of τ'_λ as

$$S_\lambda(\tau'_\lambda) = \sum_{n=0}^{\infty} a_n {\tau'_\lambda}^n \tag{2.30}$$

and plug this expression into eq. (2.23). Carrying out this substitution yields

$$\begin{aligned} I^+_\lambda(\tau'_\lambda, \theta) &= \frac{1}{\cos(\theta)} \int_{\tau'_\lambda}^{\infty} e^{\frac{\tau'_\lambda - t}{\cos(\theta)}} S_\lambda(t)\,\mathrm{d}t \\ &= e^{\frac{\tau'_\lambda}{\cos(\theta)}} \sum_{n=0}^{\infty} a_n [\cos(\theta)]^n \int_{\frac{\tau'_\lambda}{\cos(\theta)}}^{\infty} e^{-x} x^n\,\mathrm{d}x. \end{aligned} \tag{2.31}$$

Using the mathematical identity $\int_0^\infty e^{-x} x^n\,\mathrm{d}x = n!$ for $n \in \mathbb{N}$, an evaluation of the above expression at $\tau'_\lambda = 0$ brings forth

$$\begin{aligned} I^+_\lambda(\tau'_\lambda = 0, \theta) &= \sum_{n=0}^{\infty} a_n \left[\cos(\theta)\right]^n \int_0^{\infty} e^{-x} x^n\,\mathrm{d}x \\ &= \sum_{n=0}^{\infty} a_n \left[\cos(\theta)\right]^n n! \approx a_0 + a_1 \cos(\theta). \end{aligned} \tag{2.32}$$

Recalling the above given power series form for the source function $S_\lambda(\tau'_\lambda)$, an evaluation at $\tau'_\lambda = \cos(\theta)$ yields

$$\begin{aligned} S_\lambda(\tau'_\lambda = \cos(\theta)) = \sum_{n=0}^{\infty} a_n \left[\cos(\theta)\right]^n &= a_0 + a_1 \cos(\theta) + \mathcal{O}[\cos^2(\theta)] \\ &\approx a_0 + a_1 \cos(\theta) \end{aligned} \tag{2.33}$$

and therefore one obtains a relation for the outward spectral radiance $I^+_\lambda(\tau'_\lambda = 0, \theta)$ in the so-called Eddington-Barbier approximation, namely

$$I^+_\lambda(\tau'_\lambda = 0, \theta) \approx S_\lambda(\tau'_\lambda = \cos(\theta)). \tag{2.34}$$

If a ray propagates in a vertical direction with respect to the plane parallel layers of the medium against an observer, $\cos(\theta = 0) = 1$ and therefore

$$I^+_\lambda(\tau'_\lambda = 0, \theta = 0) \approx S_\lambda(\tau'_\lambda = 1) \tag{2.35}$$

hold. If the source function S_λ is given exactly as an affine function in τ'_λ, namely $S_\lambda(\tau'_\lambda) = a_0 + a_1\tau'_\lambda$, then the above two relations do not only hold approximately, but exactly, as it follows directly from eq. (2.32) and eq. (2.33).

2.2 Spectral line formation

In order to provide a qualitative understanding of spectral line formation, one can use the Eddington-Barbier relation with $\theta = 0$, that is

$$I^+_\lambda(\tau'_\lambda = 0) \approx S_\lambda(\tau'_\lambda = 1), \tag{2.36}$$

which is an approximative solution to the radiative transfer equation

$$\cos(\theta)\frac{\mathrm{d}I_\lambda(\tau'_\lambda)}{\mathrm{d}\tau'_\lambda} = I_\lambda(\tau'_\lambda) - S_\lambda(\tau'_\lambda). \tag{2.37}$$

Recall, that $S_\lambda = \frac{j_\lambda}{\alpha_\lambda}$ with j_λ the emission and α_λ the extinction coefficient is the source function and of optical depth τ'_λ is defined as $\tau'_\lambda(Z) = \int_0^Z \alpha_\lambda(z)\,\mathrm{d}z$.

Electrons in atoms and molecules as described by the Schrödinger equation have discrete energy states; these states encompass discrete rotational, vibrational and orbital energy states. Note, that electromagnetic radiation interacts with atoms atoms and molecules in such a way, that those energy states are either excited, deexcited or even ionized.

Furthermore, electromagnetic radiation can also scatter in matter. Excitation of discrete atomic or molecular energy states by photons of suitable wavelengths, combined with ionization and scattering effects, defines the extinction coefficient α_λ, while deexcitation, combined with recombination effects determines the emission coefficient j_λ. Hereby, excitation is to mean, that an electron of energy E within an atom or molecule transitions to a quantum state of higher energy $E + \Delta E$ by means of absorbing a photon of corresponding wavelength $\lambda_\gamma = \frac{hc}{\Delta E}$; hence deexcitation refers to a transition of an electron within an atom or molecule in energy state $E + \Delta E$ to the energy state E by means of emitting a photon of wavelength E_γ. Consider for an illustration of these processes fig. 2.4.

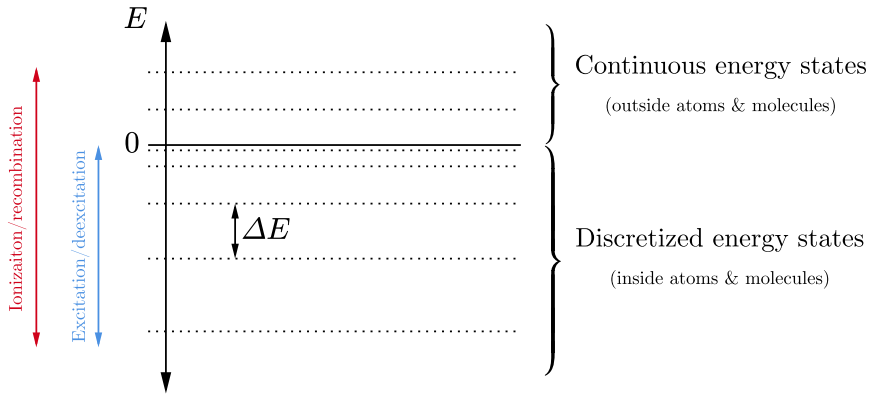


Figure 2.4: Illustration of discretized and continuous energy states in matter. Note, that the dotted lines for negative energies represent possible energy states within an atom or molecule (bound states), whereas there is actually a continuum of dotted lines for positive energies.

Consider now fig. 2.5. Consider a hypothetical atom, which is such that it contains only one electron and which has only two distinct energy states available to the electron. These energy states are separated by the energy ΔE corresponding to the wavelength $\lambda_\gamma = hc/\Delta E$. Assume further, that there is some hypothetical medium consisting of only those hypothetical atoms along with a electromagnetic radiation source; and that scattering is negligible. At precisely the wavelength λ_γ, this hypothetical medium will perfectly absorb all photons, as long as there are still some energy state changes available. This is to say, that there will also be some spike in the curve of α_λ, the extinction coefficient, as it is shown in the top left panel of fig. 2.5 - a spectral line is formed. Now, as such, the spectral line will be just a delta function; in reality however different

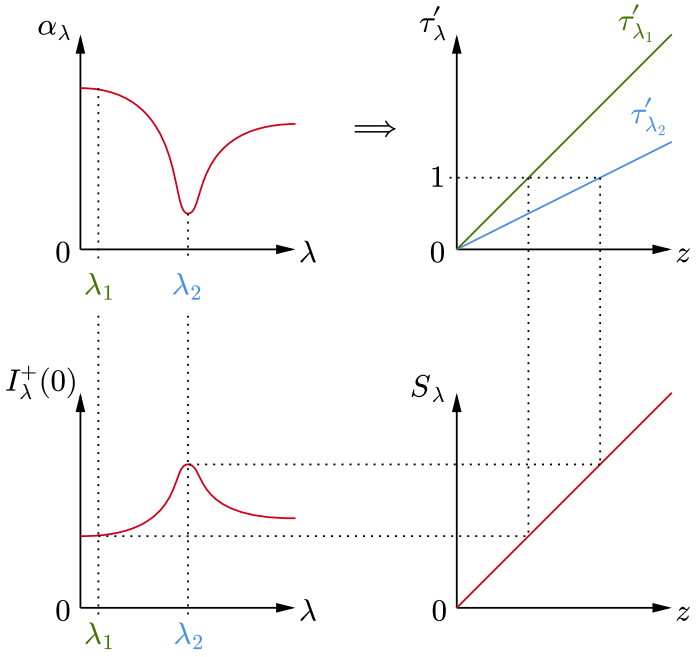


Figure 2.5: Illustration of a spectral line formation using the Eddington-Barbier relation as an explanatory tool. Note, that in this picture one assumes, that the extinction coefficient α_λ is independent of depth z, and furthermore, that the source function $S_\lambda = S$ is independent of wavelength and furthermore linear in z. This means, that the Eddington-Barbier relation does not only hold approximatively, but exactly. Furthermore, the optical depth is then just given as $\tau'_\lambda(Z) = \int_0^Z \alpha_\lambda \, \mathrm{d}z = \alpha_\lambda Z$. Figure inspired by [Rutten, 2015, p.38].

broadening mechanisms such as Doppler or Lorentz broadening of spectral lines apply, furthermore there are many more than just two energy levels available to electrons in a real medium. Therefore, in fig. 2.5 the extinction coefficient α_λ is plotted for some arbitrary, non-hypothetical medium. One can see, that the absorption line is centered as λ_2. Assuming, that the extinction coefficient is not dependent upon height z and taking into account the definition of the optical depth $\tau'_\lambda(Z) = \int_0^Z \alpha_\lambda \, \mathrm{d}z$, one deduces $\tau'_{\lambda_1}(z) = \alpha_{\lambda_1} z$ and $\tau'_{\lambda_2}(z) = \alpha_{\lambda_2} z$. Since $\alpha_{\lambda_1} \gg \alpha_{\lambda_2}$ clearly τ'_{λ_1} must be a steeper function of z than τ'_{λ_2}. This is to say, that the medium is more opaque at wavelength λ_2 than at wavelength λ_1. Taking into account now the Eddington-Barbier relation, which states, that the source function S_λ at optical depth $\tau'_\lambda = 1$ is equal to the outward spectral radiance $I^+_\lambda(0)$ one arrives at a spectral line in emission for an observer measuring $I^+_\lambda(0)$. If however, α_λ would have a spike upwards, the spectral line would appear in absorption for an observer at $\tau'_\lambda = 0$. A

study of spectral line properties and an analysis, which parameters and how they influence spectral line formation can lead to insights about the medium, in which they form. Using the theory of radiative transfer and especially simplified atmospheric models of stars for example, one can infer physical properties of an atmosphere, simply by investigating one or more spectral lines formed in the atmosphere under consideration.

2.3 Polarimetry

The subject of polarimetry deals with the polarization properties of electromagnetic waves and how they are affected by the properties of the media they are propagating in. In particular, the presence of a magnetic field in the propagating medium of electromagnetic waves leads to the so-called Zeeman splitting of spectral lines, from which the presence and magnitude of a magnetic field can be deduced. Polarimetry furthermore links observable quantities like the Stokes parameters to the parameters of spectral lines, thus providing insight to the medium, where those spectral lines formed. In the context of stellar atmospheres, polarimetry provides the link between the observable Stokes parameters and the parameters defining the stellar atmosphere, such as the magnetic field strength, inclination and azimuth.

2.3.1 Zeeman effect

The Zeeman effect for strong external magnetic fields allows inference about the external magnetic field strength. The presence of such an external magnetic field leads to the splitting of energy levels in atoms and molecules. This is to say, that also one spectral line splits into several spectral lines. According to [del Toro Iniesta, 2003, p.123], the Zeeman wavelength splitting λ_B can be quantified by means of

$$\lambda_B = \frac{\lambda^2 \nu_L}{c}, \quad \nu_L = \frac{e|\boldsymbol{B}|}{4\pi mc}, \tag{2.38}$$

where ν_L is the Larmor frequency, e is the elementary charge, m is the particle mass, λ is the wavelength of a spectral line in an inertial frame and $\boldsymbol{B}$ is the external magnetic field giving rise to the Zeeman splitting. These formulae can be derived by a rigorous calculation involving perturbation theory in quantum mechanics.

2.3.2 The Stokes parameters and Stokes vector

Consider a monochromatic electromagnetic wave $\boldsymbol{E}$. In general, $\boldsymbol{E}$ is a function of position $\boldsymbol{r} \in \mathbb{R}^3$ in euclidean space and time t, such that $\boldsymbol{E} = \boldsymbol{E}(\boldsymbol{r}, t)$. Without loss of generality, the propagation direction of the monochromatic electromagnetic wave can be defined in z-direction, hence [Stix, 2002, p.121] gives a monochromatic wave or arbitrary polarization state as

$$\boldsymbol{E}(\boldsymbol{r}, t) = \begin{pmatrix} E_x \cos(kz - \omega t) \\ E_y \cos(kz - \omega t + \xi) \\ 0 \end{pmatrix}, \qquad k = \frac{2\pi}{\lambda}. \tag{2.39}$$

The quantity ξ is the phase difference between the amplitude of the electromagnetic wave in x-direction and y-direction, which together with E_x and E_y therefore accounts for the polarization state of the wave. As a side note, notice that the dispersion relation $\omega = \omega(k)$ for a wave propagating in vacuum can be given as $\omega(k) = 2\pi\nu = ck$. For the above specified electromagnetic wave, the Stokes parameters are defined as

$$\boldsymbol{I} = \begin{pmatrix} I \\ Q \\ U \\ V \end{pmatrix} = \begin{pmatrix} E_x^2 + E_y^2 \\ E_x^2 - E_y^2 \\ 2E_x E_y \cos(\xi) \\ 2E_x E_y \sin(\xi) \end{pmatrix}, \tag{2.40}$$

as given in [Stix, 2002, p.121]. Hereby, the identity $I^2 = Q^2 + U^2 + V^2$ holds, as it is shown by the calculation

$$\begin{aligned} Q^2 + U^2 + V^2 &= E_x^4 - 2E_x^2E_y^2 + E_y^4 + 4E_x^2E_y^2\left[\cos^2(\xi) + \sin^2(\xi)\right] \\ &= E_x^4 + 2E_x^2E_y^2 + E_y^4 = (E_x^2 + E_y^2)^2 = I^2. \end{aligned} \tag{2.41}$$

Since these parameters were defined for a monochromatic electromagnetic wave, the Stokes parameters are in general functions of wavelength; the wavelength dependency of $\boldsymbol{I} = \boldsymbol{I}(\lambda)$ shall henceforth be indicated by the subscript λ, such that the Stokes parameters are denoted by

$$\boldsymbol{I}(\lambda) = \boldsymbol{I}_\lambda = (I_\lambda, Q_\lambda, U_\lambda, V_\lambda)^\top, \tag{2.42}$$

which is called the Stokes vector, whereby also $I_\lambda^2 = Q_\lambda^2 + U_\lambda^2 + V_\lambda^2$ holds. In fig. 2.6, a schematic representation of the definition of the Stokes parameters (vector) is given. Stokes Q_λ and U_λ are a measure of linear polarization, while Stokes V_λ accounts for the amount of circular polarization present in the electromagnetic waves under consideration.

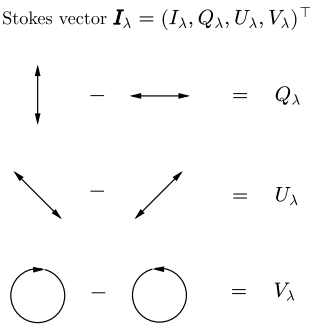


Figure 2.6: Schematic representation of the general Stokes vector $\boldsymbol{I}_\lambda$ containing the Stokes parameters I_λ, Q_λ, U_λ and V_λ. The observer is looking into negative z-direction with respect to the electromagnetic wave as defined by eq. (2.39). Figure inspired by [Degl'Innocenti, 2005, p.17].

2.3.3 The generalized radiative transfer equation

In the preceding material, the radiative transfer equation was already derived and solved for the spectral radiance $I_\lambda(\tau'_\lambda, \theta)$ as a function of the continuum optical depth τ'_λ and local inclination θ of an electromagnetic radiation ray. This radiative transfer equation was given by

$$\cos(\theta)\frac{\mathrm{d}I_\lambda(\tau'_\lambda)}{\mathrm{d}\tau'_\lambda} = I_\lambda(\tau'_\lambda) - S_\lambda(\tau'_\lambda). \tag{2.43}$$

This equation can be generalized to account not only for changes of the spectral radiance (intensity) I_λ, but for changes of every Stokes parameter $\boldsymbol{I}_\lambda = (I_\lambda, Q_\lambda, U_\lambda, V_\lambda)^\top$. This can be done by means of introducing a propagation matrix $\mathbf{K}_\lambda(\tau'_\lambda) = \mathbb{1} + \mathbf{N}_\lambda(\tau'_\lambda)$, where $\mathbf{N}_\lambda(\tau'_\lambda)$ accounts for extinction and emission properties of the medium giving rise to spectral line formation; and where $\mathbb{1}$ is the unity matrix. Using the wavelength-dependent propagation matrix $\mathbf{K}_\lambda$, the generalized radiative transfer equation for the Stokes vector can be written as

$$\frac{\mathrm{d}\boldsymbol{I}_\lambda(\tau'_\lambda)}{\mathrm{d}\tau'_\lambda} = \mathbf{K}_\lambda(\tau'_\lambda)\left[\boldsymbol{I}_\lambda(\tau'_\lambda) - \boldsymbol{S}_\lambda(\tau'_\lambda)\right], \tag{2.44}$$

as given by [del Toro Iniesta, 2003, p.150]. Note that the factor of $\cos(\theta)$ is absorbed into the propagation matrix $\mathbf{K}_\lambda$ in the above equation. In the above formulation of the generalized radiative transfer equation also the source function $S_\lambda(\tau'_\lambda)$ was generalized to the source function vector $\boldsymbol{S}_\lambda(\tau'_\lambda) = S_\lambda(\tau'_\lambda)(1,0,0,0)^\top$. It is evident from this form of the equation, that the unity matrix $\mathbb{1}$ of the propagation matrix takes care of the extinction and emission in the continuum, while the matrix $\mathbf{N}_\lambda$ takes into account extinction and emission pertaining to spectral lines.

Formal solution of the generalized radiative transfer equation

In order to derive a formal solution to the generalized radiative transfer equation following the treatment in [del Toro Iniesta, 2003, pp.152-157], the parameter $\kappa \doteq \tau'_\lambda$ shall be introduced to temporarily simplify the notation, hence yielding the generalized radiative transfer equation to be

$$\frac{\mathrm{d}\boldsymbol{I}_\lambda(\kappa)}{\mathrm{d}\kappa} = \mathbf{K}_\lambda(\kappa)\left[\boldsymbol{I}_\lambda(\kappa) - \boldsymbol{S}_\lambda(\kappa)\right]. \tag{2.45}$$

Consider now for the moment only the homogeneous differential equation

$$\frac{\mathrm{d}\boldsymbol{I}_{\lambda,h}(\kappa)}{\mathrm{d}\kappa} = \mathbf{K}_\lambda(\kappa)\boldsymbol{I}_{\lambda,h}(\kappa). \tag{2.46}$$

As it is done in [del Toro Iniesta, 2003, p.152], a so-called evolution operator $\mathbf{P}(\kappa,\kappa')$ can be defined, such that it gives the transformation $\boldsymbol{I}_{\lambda,h}(\kappa') \rightarrow \boldsymbol{I}_{\lambda,h}(\kappa)$ of the solution $\boldsymbol{I}_{\lambda,h}(\kappa')$ to the homogeneous equation from position κ' to κ, i.e.

$$\boldsymbol{I}_{\lambda,h}(\kappa) = \mathbf{P}(\kappa,\kappa')\boldsymbol{I}_{\lambda,h}(\kappa'). \tag{2.47}$$

This operator is linear, because the differential equation governing $\boldsymbol{I}_{\lambda,h}(\kappa)$ is also linear. Furthermore, because of the way it was defined, it has the properties

$$\mathbf{P}(\kappa,\kappa) = \mathbb{1}, \qquad \mathbf{P}(\kappa,\kappa')\mathbf{P}(\kappa',\kappa'') = \mathbf{P}(\kappa,\kappa''). \tag{2.48}$$

Differentiating eq. (2.47) with respect to κ, namely

$$\frac{\mathrm{d}\boldsymbol{I}_{\lambda,h}(\kappa)}{\mathrm{d}\kappa} = \frac{\mathrm{d}\mathbf{P}(\kappa,\kappa')}{\mathrm{d}\kappa}\boldsymbol{I}_{\lambda,h}(\kappa') = \mathbf{K}_\lambda(\kappa)\mathbf{P}(\kappa,\kappa')\boldsymbol{I}_{\lambda,h}(\kappa'), \tag{2.49}$$

the identity

$$\frac{\mathrm{d}\mathbf{P}(\kappa,\kappa')}{\mathrm{d}\kappa} = \mathbf{K}_\lambda(\kappa)\mathbf{P}(\kappa,\kappa') \tag{2.50}$$

results. Furthermore differentiating eq. (2.47) with respect to κ' yields

$$\frac{\mathrm{d}\boldsymbol{I}_{\lambda,h}(\kappa)}{\mathrm{d}\kappa'} = \mathbf{P}(\kappa,\kappa')\frac{\mathrm{d}\boldsymbol{I}_{\lambda,h}(\kappa')}{\mathrm{d}\kappa'} + \frac{\mathrm{d}\mathbf{P}(\kappa,\kappa')}{\mathrm{d}\kappa'}\boldsymbol{I}_{\lambda,h}(\kappa') = 0. \tag{2.51}$$

By means of rearranging this equation and inserting the definition of the homogeneous equation (2.46) the equation

$$\begin{aligned}\frac{\mathrm{d}\mathbf{P}(\kappa,\kappa')}{\mathrm{d}\kappa'}\boldsymbol{I}_{\lambda,h}(\kappa') &= -\mathbf{P}(\kappa,\kappa')\frac{\mathrm{d}\boldsymbol{I}_{\lambda,h}(\kappa')}{\mathrm{d}\kappa'} \\ &= -\mathbf{P}(\kappa,\kappa')\mathbf{K}_\lambda(\kappa')\boldsymbol{I}_{\lambda,h}(\kappa')\end{aligned} \tag{2.52}$$

obtains, wherefrom the identity

$$\frac{\mathrm{d}\mathbf{P}(\kappa,\kappa')}{\mathrm{d}\kappa'} = -\mathbf{P}(\kappa,\kappa')\mathbf{K}_\lambda(\kappa') \tag{2.53}$$

is deduced.

Now, the generalized radiative transfer equation eq. (2.45) is multiplied with the operator $\mathbf{P}(\kappa',\kappa)$ from the left, thus obtaining

$$\begin{aligned}\mathbf{P}(\kappa',\kappa)\frac{\mathrm{d}\boldsymbol{I}_\lambda(\kappa)}{\mathrm{d}\kappa} &= \mathbf{P}(\kappa',\kappa)\mathbf{K}_\lambda(\kappa)\left[\boldsymbol{I}_\lambda(\kappa) - \boldsymbol{S}_\lambda(\kappa)\right] \\ &= \frac{\mathrm{d}}{\mathrm{d}\kappa}\left[\mathbf{P}(\kappa',\kappa)\boldsymbol{I}_\lambda(\kappa)\right] - \frac{\mathrm{d}\mathbf{P}(\kappa',\kappa)}{\mathrm{d}\kappa}\boldsymbol{I}_\lambda(\kappa) \\ &= -\frac{\mathrm{d}\mathbf{P}(\kappa',\kappa)}{\mathrm{d}\kappa}\left[\boldsymbol{I}_\lambda(\kappa) - \boldsymbol{S}_\lambda(\kappa)\right],\end{aligned} \tag{2.54}$$

where the identity eq. (2.53) was employed in the last step. Rearranging the above equation yields

$$\frac{\mathrm{d}}{\mathrm{d}\kappa}\left[\mathbf{P}(\kappa',\kappa)\boldsymbol{I}_\lambda(\kappa)\right] = \frac{\mathrm{d}\mathbf{P}(\kappa',\kappa)}{\mathrm{d}\kappa}\boldsymbol{S}_\lambda(\kappa) = -\mathbf{P}(\kappa',\kappa)\mathbf{K}_\lambda(\kappa)\boldsymbol{S}_\lambda(\kappa), \tag{2.55}$$

where again identity eq. (2.53) was made use of. One has now an integratable equation with respect to κ. Integrating the left hand side of the above equation from κ_0 to κ_1 gives

$$\begin{aligned}\int_{\kappa_0}^{\kappa_1}\frac{\mathrm{d}}{\mathrm{d}\kappa}\left[\mathbf{P}(\kappa',\kappa)\boldsymbol{I}_\lambda(\kappa)\right]\,\mathrm{d}\kappa &= \mathbf{P}(\kappa',\kappa_1)\boldsymbol{I}_\lambda(\kappa_1) - \mathbf{P}(\kappa',\kappa_0)\boldsymbol{I}_\lambda(\kappa_0) \\ &= -\int_{\kappa_0}^{\kappa_1}\mathbf{P}(\kappa',\kappa)\mathbf{K}_\lambda(\kappa)\boldsymbol{S}_\lambda(\kappa)\,\mathrm{d}\kappa.\end{aligned} \tag{2.56}$$

Through rearrangement of the equation and multiplication with $\mathbf{P}(\kappa_1,\kappa')$, the expressions

$$\begin{aligned}\mathbf{P}(\kappa_1,\kappa')\mathbf{P}(\kappa',\kappa_1)\boldsymbol{I}_\lambda(\kappa_1) &= \mathbf{P}(\kappa_1,\kappa')\mathbf{P}(\kappa',\kappa_0)\boldsymbol{I}_\lambda(\kappa_0) \\ &\quad - \int_{\kappa_0}^{\kappa_1}\mathbf{P}(\kappa_1,\kappa')\mathbf{P}(\kappa',\kappa)\mathbf{K}_\lambda(\kappa)\boldsymbol{S}_\lambda(\kappa)\,\mathrm{d}\kappa,\end{aligned} \tag{2.57}$$

results, which can due to $\mathbf{P}(\kappa,\kappa')\mathbf{P}(\kappa',\kappa) = \mathbf{P}(\kappa,\kappa) = \mathbb{1}$ and $\mathbf{P}(\kappa'',\kappa')\mathbf{P}(\kappa',\kappa) = \mathbf{P}(\kappa'',\kappa)$ further simplified to a first formal solution to the generalized radiative transfer equation, namely

$$\boldsymbol{I}_\lambda(\kappa_1) = \mathbf{P}(\kappa_1,\kappa_0)\boldsymbol{I}_\lambda(\kappa_0) - \int_{\kappa_0}^{\kappa_1}\mathbf{P}(\kappa_1,\kappa)\mathbf{K}_\lambda(\kappa)\boldsymbol{S}_\lambda(\kappa)\,\mathrm{d}\kappa. \tag{2.58}$$

Resubstituting $\tau'_\lambda = \kappa$ and hence $\mathrm{d}\kappa = \mathrm{d}\tau'_\lambda$ and identifying $\kappa_0 = \tau'_{\lambda,0} \to \infty$ and $\kappa_1 = \tau'_{\lambda,1} \to \tau'_\lambda$, one gets the general formal solution

$$\boldsymbol{I}_\lambda(\tau'_\lambda) = \mathbf{P}\left(\tau'_\lambda, \infty\right) \boldsymbol{I}_\lambda(\infty) - \int_\infty^{\tau'_\lambda} \mathbf{P}\left(\tau'_\lambda, t\right) \mathbf{K}_\lambda(t) \boldsymbol{S}_\lambda(t)\, \mathrm{d}t \tag{2.59}$$

to the generalized radiative transfer equation. Considering fig. 2.3, the optical depth $\tau'_\lambda \to \infty$ can be understood to be located at the limit, where no photon can ever escape, e.g. at the center of a star, whose radiative transfer is described by eq. (2.59). Further taking a closer look at fig. 2.3, the optical depth $\tau'_\lambda = 0$ can be understood as the position of an observer located outside or right at boundary of the radiative transfer system, that is for example the position of an observer looking at a stellar atmosphere from outside the atmosphere. In this case, the measurable Stokes vector at the position of an observer outside the radiative transfer system is given as

$$\boldsymbol{I}_\lambda(\tau'_\lambda = 0) = \int_0^\infty \mathbf{P}\left(0, t\right) \mathbf{K}_\lambda(t) \boldsymbol{S}_\lambda(t)\, \mathrm{d}t. \tag{2.60}$$

This result is obtained from eq. (2.59), where [del Toro Iniesta, 2003, p.153] gives the limit of the evolution operator $\mathbf{P}(0,t)$ for $t \to \infty$ to be

$$\lim_{t\to\infty} \mathbf{P}(0,t) = 0. \tag{2.61}$$

This result makes sense insofar, as this limit can be represented as $\boldsymbol{I}_{\lambda,h}(0) = \mathbf{P}(0,\infty)\boldsymbol{I}_{\lambda,h}(\infty)$ by means of using the definition eq. (2.47) of the evolution operator; it should account for the change in $\boldsymbol{I}_{\lambda,h}$ from $\tau'_\lambda \to \infty$ to $\tau'_\lambda = 0$. Since the solution $\boldsymbol{I}_{\lambda,h}$ to the homogeneous radiative transfer equation eq. (2.46) does not contain any source term, the homogeneous solution must change to zero as one goes from optical depth infinity to optical depth zero. This is because in this case, a ray travels a theoretically infinite distance through the radiative transfer system without any sources along the way, therefore extincting all photons on the way to optical depth zero.

2.3.4 Absorption, emission and dispersion profile functions

Absorption can be defined as the removal of energy from an electromagnetic field by interaction with matter [del Toro Iniesta, 2003, p.87]. Emission however, is the opposite of absorption, namely contribution of

energy to an electromagnetic field by means of photons emerging from electronic transitions in atoms and molecules. There is a third effect due to the interaction of matter with electromagnetic fields, namely dispersion; [del Toro Iniesta, 2003, p.87] defines it as the dephasing of the electric field components as the radiation propagates through matter.

Spectral lines due to absorption and emission properties of matter can be accounted for by so-called profile functions. These functions are not just delta functions, as the discrete energy levels in atoms and molecules would suggest, but are always broadened by different processes. Among these processes are the so-called pressure broadening (Lorentz broadening) and the thermal broadening (Doppler broadening). Doppler broadening is caused by the thermal velocities of atoms and molecules in matter, which move in arbitrary directions in general. These effects are accounted for by distinct profile functions; these two profile functions can however be combined to account for both pressure and thermal broadening by means of the mathematical operation of a convolution; thus yielding the Voigt profile function $H(u, a)$ as

$$H(u,a) = \frac{a}{\pi} \int_{-\infty}^{\infty} e^{-y^2} \frac{1}{(u-y)^2 + a^2} \, \mathrm{d}y, \tag{2.62}$$

as given by [del Toro Iniesta, 2003, p.100]. Furthermore, the dispersion effects are accounted for by the Faraday-Voigt function $F(u, a)$ given by

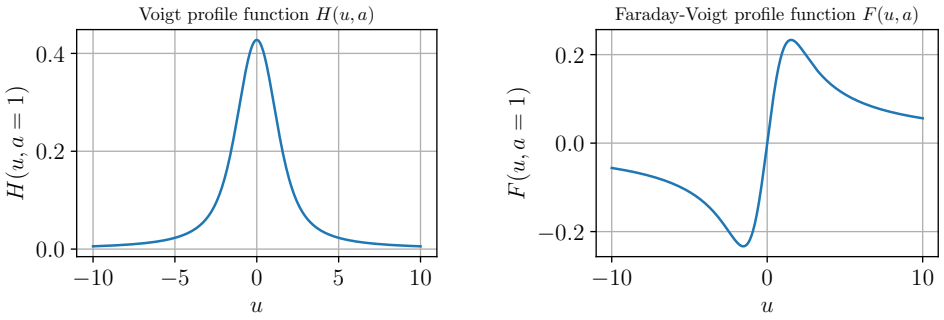
$$F(u,a) = \frac{1}{\pi} \int_{-\infty}^{\infty} e^{-y^2} \frac{u-y}{(u-y)^2 + a^2} \, \mathrm{d}y. \tag{2.63}$$

The Voigt and the Faraday-Voigt profile functions are visualized in fig. 2.7.

2.3.5 The propagation matrix

The propagation matrix $\mathbf{K}_\lambda(\tau'_\lambda)$ accounts for absorption and dispersion effects of electromagnetic waves in matter within the framework of the generalized radiative transfer equation. In particular, it accounts for spectral lines in a radiation continuum. Explicitly, [Stix, 2002, p.154] gives the propagation matrix to be defined as

$$\mathbf{K}_\lambda = \begin{pmatrix} \eta_I & \eta_Q & \eta_U & \eta_V \\ \eta_Q & \eta_I & \rho_V & -\rho_U \\ \eta_U & -\rho_V & \eta_I & \rho_Q \\ \eta_V & \rho_U & -\rho_Q & \eta_I \end{pmatrix}. \tag{2.64}$$



(a) Voigt profile function for $a = 1$.

(b) Faraday-Voigt profile function for $a = 1$.

Figure 2.7: Voigt and Faraday-Voigt profile functions for $a = 1$.

The individual entries of this matrix [del Toro Iniesta, 2003, p.116] accounts for by

$$\begin{aligned}
\eta_I &= 1 + \frac{\eta_0}{2}\left[\phi_0 \sin^2(\theta) + \frac{1}{2}(\phi_{+1} + \phi_{-1})(1 + \cos^2(\theta))\right], \\
\eta_Q &= \frac{\eta_0}{2}\left[\phi_0 - \frac{1}{2}(\phi_{+1} + \phi_{-1})\right] \sin^2(\theta)\cos(2\varphi), \\
\eta_U &= \frac{\eta_0}{2}\left[\phi_0 - \frac{1}{2}(\phi_{+1} + \phi_{-1})\right] \sin^2(\theta)\sin(2\varphi), \\
\eta_V &= \frac{\eta_0}{2}(\phi_{-1} - \phi_{+1})\cos(\theta), \\
\rho_Q &= \frac{\eta_0}{2}\left[\psi_0 - \frac{1}{2}(\psi_{+1} + \psi_{-1})\right] \sin^2(\theta)\cos(2\varphi), \\
\rho_U &= \frac{\eta_0}{2}\left[\psi_0 - \frac{1}{2}(\psi_{+1} + \psi_{-1})\right] \sin^2(\theta)\sin(2\varphi), \\
\rho_V &= \frac{\eta_0}{2}(\psi_{-1} - \psi_{+1})\cos(\theta).
\end{aligned} \tag{2.65}$$

Hereby, the angles θ and φ are called inclination and azimuth and are defined as seen in fig. 2.8. The ranges for those angles are $\theta \in [0, \pi]$ and $\varphi \in [0, \pi]$. In the case of solar observations, $\theta = 0$ means that a vector field $\boldsymbol{V}$ under consideration is oriented towards an observer on-looking the Sun, whereas $\varphi =$ does either mean, that the vectorial quantity $\boldsymbol{V}$ faces either perfectly west or east on the solar surface. $\theta = \pi$ then does mean, that $\boldsymbol{V}$ points exactly away from the observer.

The quantities ϕ_α and ψ_α for $\alpha \in \{0, -1, 1\}$ are left to be defined;

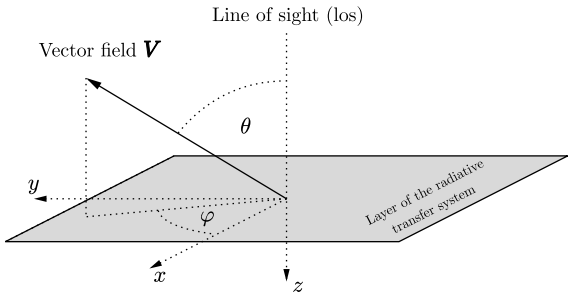


Figure 2.8: Visualization of the definitions for the inclination and azimuth angles θ and φ.

[del Toro Iniesta, 2003, p.123] defines these parameters as

$$\begin{aligned} \phi_\alpha(u,a) &= \frac{1}{\sqrt{\pi}} H(u + \alpha u_B - u_{los}, a), \\ \psi_\alpha(u,a) &= \frac{1}{\sqrt{\pi}} F(u + \alpha u_B - u_{los}, a). \end{aligned} \tag{2.66}$$

Herewith, u, u_B and u_{los} aswell as a are dimensionless quantities determining the the Voigt and Faraday-Voigt functions. In these functions u is related to the wavelength λ and is to be regarded as a fixed value for a certain spectral line under consideration. [Uitenbroek, 2020, p.21] gives the parameter u_{los} as

$$u_{los} = \lambda \frac{\boldsymbol{u} \cdot \boldsymbol{n}}{c \Delta\lambda_D}, \tag{2.67}$$

where $\boldsymbol{n}$ defines the line of sight vector and $\boldsymbol{u}$ the velocity of the medium constituents relative to an inertial observer outside the radiative transfer system. This is a dimensionless number related to the actual line of sight velocity v_{los} as $v_{los} = c u_{los}$. Furthermore, $\Delta\lambda_D$ refers to the Doppler broadening of a spectral line due to the thermal velocity

$$v_{th} = \sqrt{\frac{2 k_B T}{m}} \tag{2.68}$$

of the particles of mass m in the medium; it is given by the expression

$$\Delta\lambda_D = \frac{v_{th}\lambda}{c} = \frac{\lambda}{c}\sqrt{\frac{2 k_B T}{m}}. \tag{2.69}$$

Since the thermal velocity of the particles is responsible for the Doppler broadening, one can also call the thermal velocity v_{th} the Doppler velocity $v_{dop} \doteq v_{th}$. Last but not least, u_B is a factor originating in the

Zeeman effect; therefore it connects the radiative transfer equation with an external magnetic field $\boldsymbol{B}$ via the relation

$$u_B = g_L \frac{e\lambda^2 |\boldsymbol{B}|}{4\pi mc\Delta\lambda_D}, \tag{2.70}$$

where g_L is the Landé factor. In total therefore, the propagation matrix $\mathbf{K}_\lambda$ for a certain wavelength λ can be fully determined by means of the seven independent parameters $\{|\boldsymbol{B}|, \theta, \varphi, v_{los}, v_{dop}, a, \eta_0\}$.

2.3.6 The Milne-Eddington approximation

The Milne-Eddington approximation of radiative transfer allows for the construction of a relatively simple stellar atmosphere model $\boldsymbol{y} = \boldsymbol{M}(\boldsymbol{x})$, where $\boldsymbol{x}$ would be the atmospheric parameters, such as magnetic field strength, inclination and so forth, and where $\boldsymbol{y}$ would be the predicted observational data, namely the Stokes vectors $\boldsymbol{I}_\lambda$ for a certain wavelength interval $\lambda \in [\lambda_-, \lambda_+]$ giving so-called Stokes profiles. A Milne-Eddington model for a stellar atmosphere can even be analytically inverted, which means that there is a mathematically exact way to calculate atmospheric parameters $\boldsymbol{x}$ based on observational data $\boldsymbol{y}$ by means of mathematically inverting the model, i.e. $\boldsymbol{x} = \boldsymbol{M}^{-1}(\boldsymbol{y})$. In the case of real and not synthetic observational data $\boldsymbol{y}$, the best fit to the data given the model $\boldsymbol{y} = \boldsymbol{M}(\boldsymbol{x})$ is to be found by iteratively changing the parameters; that is by means of synthesizing observations, comparing them to the real observations and adjusting the parameters, such that a better fit to the real observations is expected in the next iteration.

The simplest possible solution for the generalized radiative transfer equation eq. (2.44) is the so-called Milne-Eddington solution. Its derivation relies on the Milne-Eddington approximation, which basically consists of three fundamental assumptions given in [Degl'Innocenti, 2005, p.411] profoundly simplifying the radiative transfer problem:

(1) The radiative transfer system is assumed to be of a plane parallel nature and in a local thermal equilibrium.

(2) All parameters relevant to spectral line formation are required to be independent of all measures of height within the system.

(3) The source function is required to be affinely dependent upon the continuum optical depth.

The second assumption is to say, that the propagation matrix $\mathbf{K}_\lambda$ is isotropically constant, i.e.

$$\mathbf{K}_\lambda = \mathbf{K}_{\lambda,0}. \tag{2.71}$$

The third ingredient to the Milne-Eddington model means, that the source function vector can be written as

$$\boldsymbol{S}_\lambda(\tau'_\lambda) = \boldsymbol{S}_{\lambda,0} + \tau'_\lambda \boldsymbol{S}_{\lambda,1}, \tag{2.72}$$

where $\boldsymbol{S}_{\lambda,0} = S_{\lambda,0}(1,0,0,0)^\top$ and $\boldsymbol{S}_{\lambda,1} = S_{\lambda,1}(1,0,0,0)^\top$ are constant vectors. Using the first assumption of the Milne-Eddington model, one can furthermore make the identifications $S_{\lambda,0} = B_\lambda(T)$ and $S_{\lambda,1} = B_\lambda(T)\beta$ with $\beta \in \mathbb{R}$ a constant. This is because local thermal equilibrium implies, that the source function is locally equal to the Planck function.

In the Milne-Eddington case, the homogeneous generalized radiative transfer equation eq. (2.46) reads as

$$\frac{\mathrm{d}\boldsymbol{I}_{\lambda,h}(\tau'_\lambda)}{\mathrm{d}\tau'_\lambda} = \mathbf{K}_{\lambda,0}\boldsymbol{I}_{\lambda,h}(\tau'_\lambda). \tag{2.73}$$

According to standard mathematical literature, the matrix exponential $e^{\mathbf{A}}$ for a matrix $\mathbf{A}$ is defined as

$$e^{\mathbf{A}} = \mathbb{1} + \mathbf{A} + \frac{1}{2!}\mathbf{A}^2 + \frac{1}{3!}\mathbf{A}^3 + \cdots = \sum_{k=1}^{\infty} \frac{\mathbf{A}^k}{k!}, \tag{2.74}$$

Analogously to the scalar case, a solution to the generalized homogeneous radiative transfer equation can be given as

$$\boldsymbol{I}_\lambda(\tau'_\lambda = 0) = e^{-\mathbf{K}_{\lambda,0}\tau'_\lambda}\boldsymbol{I}_\lambda(\tau'_\lambda). \tag{2.75}$$

Comparing this to the definition eq. (2.47) of the evolution operator, one finds

$$\mathbf{P}(0,\tau'_\lambda) = e^{-\mathbf{K}_{\lambda,0}\tau'_\lambda} \tag{2.76}$$

in the Milne-Eddington case. One can now calculate the solution $\boldsymbol{I}_\lambda(\tau'_\lambda = 0,\theta)$ using the special case eq. (2.60) of the formal solution to the generalized radiative transfer equation, namely

$$\boldsymbol{I}_\lambda(\tau'_\lambda = 0,\theta) = \int_0^\infty \mathbf{P}\,(0,t)\,\mathbf{K}_\lambda(t)\boldsymbol{S}_\lambda(t)\,\mathrm{d}t. \tag{2.77}$$

First, note that the derivative of $e^{-\mathbf{K}_{\lambda,0}\tau'_\lambda}$ is given by

$$\frac{\mathrm{d}}{\mathrm{d}\tau'_\lambda} e^{-\mathbf{K}_{\lambda,0}\tau'_\lambda} = -e^{-\mathbf{K}_{\lambda,0}\tau'_\lambda}\mathbf{K}_{\lambda,0}, \tag{2.78}$$

because

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}x}e^{x\mathbf{A}} &= \frac{\mathrm{d}}{\mathrm{d}x}\left(\mathbb{1} + x\mathbf{A} + \frac{1}{2!}x^2\mathbf{A}^2 + \frac{1}{3!}x^3\mathbf{A}^3 + \dots\right) \\ &= \mathbf{A} + x\mathbf{A}^2 + \frac{1}{2!}x^2\mathbf{A}^3 + \frac{1}{3!}x^3\mathbf{A}^4 + \dots \\ &= \left(\mathbb{1} + x\mathbf{A} + \frac{1}{2!}x^2\mathbf{A}^2 + \frac{1}{3!}x^3\mathbf{A}^3 + \dots\right)\mathbf{A} = e^{x\mathbf{A}}\mathbf{A},\end{aligned} \tag{2.79}$$

hence consequently also $\frac{\mathrm{d}}{\mathrm{d}x}e^{-x\mathbf{A}} = -e^{x\mathbf{A}}\mathbf{A}$ holds for any matrix $\mathbf{A}$ and real parameter x. Therefore, $-e^{-\mathbf{K}_{\lambda,0}\tau'_\lambda}\mathbf{K}_{\lambda,0}^{-1}$ is an antiderivative of $-e^{-\mathbf{K}_{\lambda,0}\tau'_\lambda}$. Now, one proceeds by calculating

$$\begin{aligned}\boldsymbol{I}_\lambda(\tau'_\lambda = 0) &= \int_0^\infty \mathbf{P}\,(0,t)\,\mathbf{K}_\lambda(t)\boldsymbol{S}_\lambda(t)\,\mathrm{d}t \\ &= \int_0^\infty \underbrace{e^{-\mathbf{K}_{\lambda,0}t}\mathbf{K}_{\lambda,0}}_{\doteq\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{F}(t)}\underbrace{(\boldsymbol{S}_{\lambda,0} + t\boldsymbol{S}_{\lambda,1})}_{\doteq\boldsymbol{G}(t)}\,\mathrm{d}t \\ &= [\mathbf{F}(t)\boldsymbol{G}(t)]_0^\infty - \int_0^\infty \mathbf{F}(t)\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{G}(t)\,\mathrm{d}t \\ &= \left[-e^{-\mathbf{K}_{\lambda,0}t}\mathbf{K}_{\lambda,0}^{-1}\mathbf{K}_{\lambda,0}(\boldsymbol{S}_{\lambda,0} + t\boldsymbol{S}_{\lambda,1})\right]_0^\infty \\ &\quad + \int_0^\infty e^{-\mathbf{K}_{\lambda,0}t}\mathbf{K}_{\lambda,0}^{-1}\mathbf{K}_{\lambda,0}\boldsymbol{S}_{\lambda,1}\,\mathrm{d}t \\ &= \boldsymbol{S}_{\lambda,0} + \int_0^\infty e^{-\mathbf{K}_{\lambda,0}t}\boldsymbol{S}_{\lambda,1}\,\mathrm{d}t \\ &= \boldsymbol{S}_{\lambda,0} + \left[e^{-\mathbf{K}_{\lambda,0}t}\mathbf{K}_{\lambda,0}^{-1}\boldsymbol{S}_{\lambda,1}\right]_0^\infty \\ &= \boldsymbol{S}_{\lambda,0} - \mathbf{K}_{\lambda,0}^{-1}\boldsymbol{S}_{\lambda,1}.\end{aligned} \tag{2.80}$$

This is the fundamental result of the Milne-Eddington approximation and is sometimes also called the Unno-Rachkovsky solution, as stated by [del Toro Iniesta, 2003, p.160]. It is important to note here, that λ refers to the central wavelength of the considered spectral line(s), which means, that the parameters defining the Milne-Eddington atmosphere are considered to be constant for the wavelength interval(s) around the spectral line(s) under examination. Identifying therefore $\boldsymbol{S}_{\lambda,0} = S_{\lambda,0}(1,0,0,0)^\top = S_0(1,0,0,0)^\top$ and $\boldsymbol{S}_{\lambda,1} = S_{\lambda,1}(1,0,0,0)^\top = S_1(1,0,0,0)^\top$ and taking into account, that the propagation matrix $\mathbf{K}_{\lambda,0}$ depends upon the seven parameters as specified in section 2.3.5, the Stokes values $\boldsymbol{I}_\lambda(\tau'_\lambda = 0)$ around the central wavelength λ of a spectral line are determined fully by the set of nine parameters $|\boldsymbol{B}|$, θ, φ, v_{los}, v_{dop}, a, η_0, S_0, S_1. Note, that S_0

is related to the Planck function as $B_\lambda(T) = S_0$; therefore, one can also assign a temperature T to the system under investigation once one has a value for S_0 and knows which wavelength λ to consider.

The parameter vector $\boldsymbol{x}$ of the Milne-Eddington model is hence given as

$$\boldsymbol{x} = (|\boldsymbol{B}|, \theta, \varphi, v_{los}, v_{dop}, a, \eta_0, S_0, S_1)^\top \in \mathbb{R}^9. \tag{2.81}$$

Let a number $D \in \mathbb{N}$ of wavelengths be sampled from a given wavelength interval $[\lambda_{min}, \ldots, \lambda, \ldots, \lambda_{max}]$ centered around λ, the central wavelength of a spectral line. Therefore, the observations vector $\boldsymbol{y}$ of the Milne-Eddington model is accounted for by

$$\boldsymbol{y} = (\boldsymbol{I}_{\lambda_{min}}, \ldots, \boldsymbol{I}_\lambda, \ldots, \boldsymbol{I}_{\lambda_{max}})^\top \in \mathbb{R}^D. \tag{2.82}$$

Thus, the full Milne-Eddington model can be written as

$$\boldsymbol{M} : \mathbb{R}^9 \to \mathbb{R}^D, \quad \boldsymbol{x} = \begin{pmatrix} |\boldsymbol{B}| \\ \theta \\ \varphi \\ v_{los} \\ v_{dop} \\ a \\ \eta_0 \\ S_0 \\ S_1 \end{pmatrix} \mapsto \boldsymbol{y} = \boldsymbol{M}(\boldsymbol{x}) = \begin{pmatrix} \boldsymbol{I}_{\lambda_{min}} \\ \vdots \\ \boldsymbol{I}_\lambda \\ \vdots \\ \boldsymbol{I}_{\lambda_{max}} \end{pmatrix}, \tag{2.83}$$

where eq. (2.80) is identified to define the model $\boldsymbol{y} = \boldsymbol{M}(\boldsymbol{x})$.

2.4 Stellar atmosphere models

2.4.1 Stellar atmosphere inversions

Consider some observation $\boldsymbol{y}$ of a stellar atmosphere. Assume, that there is a model $\boldsymbol{M}$ with associated parameters $\boldsymbol{x}$, such that the model is given by the relationship $\boldsymbol{M} = \boldsymbol{M}(\boldsymbol{x}) = \boldsymbol{y}$.

So if one knows the parameter values $\boldsymbol{x}_0$ of the stellar atmosphere model $\boldsymbol{M}$, then observations $\boldsymbol{y}_0$ can be generated based on the parameter values $\boldsymbol{x}_0$ by means of the relation $\boldsymbol{y}_0 = \boldsymbol{M}(\boldsymbol{x}_0)$; this is called the forward model. However, if one only has observations $\boldsymbol{y}_0$ and the stellar model $\boldsymbol{M}$, one would like to find out the model parameters $\boldsymbol{x}_0$ associated to these observations; this is called the backward model and can be achieved by means of applying the relation $\boldsymbol{x}_0 = \boldsymbol{M}^{-1}(\boldsymbol{y}_0)$. As one

can see, this requires a so-called inversion of the atmosphere model $\boldsymbol{M}$, which can lead to degeneracy in the model parameters $\boldsymbol{x}_0$, meaning that more than one configuration of $\boldsymbol{x}_0$ can lead to the same observations $\boldsymbol{y}_0$. A normalizing flow helps to disentangle this degeneracy by means of introducing a probability density function for each model parameter, such that the more likely parameter configurations leading to some observations $\boldsymbol{y}_0$ can be distinguished from less likely parameter configurations leading to the same observations.

2.4.2 Solving a stellar atmosphere inversion problem

Consider a stellar model $\boldsymbol{M}(\boldsymbol{x})$. Given observations $\boldsymbol{y}_0$, one would like to infer the stellar parameters $\boldsymbol{x}_0$ associated to these observations by applying the relation

$$\boldsymbol{x}_0 = \boldsymbol{M}^{-1}(\boldsymbol{y}_0). \tag{2.84}$$

Commonly, such inversion problems are carried out using a gradient search minimization algorithm which basically functions as follows. First, a function which should be minimized is defined; in our case

$$\boldsymbol{F}(\boldsymbol{x}) \doteq \boldsymbol{M}^{-1}(\boldsymbol{y}_0) - \boldsymbol{x}. \tag{2.85}$$

By computing the gradient of $\boldsymbol{F}(\boldsymbol{x}) = (F_1(\boldsymbol{x}), \ldots, F_n(\boldsymbol{x}))^\top$, $n \in \mathbb{N}$ with respect to the parameters $\boldsymbol{x}$, that is for every $F_i \in \boldsymbol{F} = (F_1, \ldots, F_n)^\top$ the vector $\boldsymbol{\nabla}_{\boldsymbol{x}} F_i(\boldsymbol{x})$ is calculated, one approaches a minimum of $\|\boldsymbol{F}\|$, if $\boldsymbol{x}$ is iteratively changed in the direction of negative gradient of $\boldsymbol{F}(\boldsymbol{x})$.

But these types of algorithms are computationally costly, if one wants to infer some sort of distribution of model parameters based on more than just one observation; this is because in the case of multiple observations, as it is almost always the case in real-life applications, a possible solution to the minimization problem needs to be valid for all observations. Furthermore, this procedure just outputs one single vector as a solution. Here the normalizing flows technique comes into play, which provides a computationally more efficient way to do inversions and furthermore allows for the possibility to construct a probability density of possible solutions to the inversion problem, because the a trained normalizing flow outputs not just one single vector as gradient descent does, but a whole bunch of solution vectors. This means, that for each component of these solution vectors one can construct a probability density just by interpolating between the points of a normalized histogram for each component.

2.4.3 Calculation of synthesized datasets using a stellar atmosphere model

In the case, where data $\boldsymbol{x} \in \mathbb{R}^d$ is conditioned by context $\boldsymbol{y} \in \mathbb{R}^D$, it can be of interest to find a mathematical model implementing a transformation $\boldsymbol{y} = \boldsymbol{M}(\boldsymbol{x})$. In order to be able to train a normalizing flow implementing a transformation

$$\boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})} : \mathbb{R}^d \to \mathbb{R}^d, \quad \boldsymbol{x} \mapsto \boldsymbol{z} = \boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{x}), \tag{2.86}$$

if can be a commodity to know the true posterior distribution $p(\boldsymbol{x}|\boldsymbol{y})$, thus enabling a comparison to the posterior distribution $p_{\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{x}|\boldsymbol{y})$ learned by the normalizing flow; this can be done by application of the Bayes theorem. Note, that $\boldsymbol{\phi}(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ are weights depending upon training data $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ associated to the implementation of the function $\boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi}}$ by a neural network. As to calculate the true posterior distribution by means of the model $\boldsymbol{M}(\boldsymbol{x}) = \boldsymbol{y}$ linking together the so-called context $\boldsymbol{y}$ with the input $\boldsymbol{x}$ to the normalizing flow, one can proceed as follows.

First, $\boldsymbol{x}$ is assumed to follow a specific probability density $p(\boldsymbol{x})$, for example a uniform distribution. Next, one can sample $L \in \mathbb{N}$ datapoints $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_L\}$ from the prior $p(\boldsymbol{x})$. Using the model $\boldsymbol{M}$, for every $\boldsymbol{x}_k$ the corresponding value $\boldsymbol{y}_k$ can be calculated as $\boldsymbol{y}_k = \boldsymbol{M}(\boldsymbol{x}_k)$, leading to the likelihood distribution $p(\boldsymbol{y}|\boldsymbol{x})$. Using the Bayes theorem and the law of total probability, one can calculate $p(\boldsymbol{x}_k|\boldsymbol{y}_k)$ as

$$p(\boldsymbol{x}_k|\boldsymbol{y}_k) = \frac{p(\boldsymbol{y}_k|\boldsymbol{x}_k)p(\boldsymbol{x}_k)}{\sum_{j=1}^{L} p(\boldsymbol{y}_k|\boldsymbol{x}_j)p(\boldsymbol{x}_j)}, \tag{2.87}$$

leading to the probability distribution $p(\boldsymbol{x}|\boldsymbol{y})$ for $k \in \{1, \ldots, L\}$ and $L \to \infty$.

Chapter 3

Artificial intelligence

The field of artificial intelligence (AI) is relatively recent field of research dedicated to mimic the process of learning information. Hereby, inspiration for first principles of a sub-field of artificial, the so-called deep learning, comes from the brain structure of mammals.

As artificial intelligence tries to mimic the process of learning, it remains to be answered what learning actually is. In order to define how a learning process is constituted, another quantity has to be defined first, namely information. [Mitchell, 1997, p.2] defines learning such that a computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E. If the so-called information entropy, which will be defined at a later stage, serves as a performance measure P, the process of learning can also be described by the statement, that any learning process decreases the information entropy of some system under consideration.

3.1 Information theory

Following [Goodfellow et al., 2016, p.73], the fundamental intuition of information theory is the consideration, that the occurrence of an unlikely event should convey more information than the occurrence of a likely event. The quantification of information in the field of information theory therefore relies on three key requirements, as given by [Goodfellow et al., 2016, p.73]:

(1) Likely events should have low information content. Guaranteed events are required to have no information content.

(2) Unlikely events should have high information content.

(3) Independent events should have additive information; i.e. learning about two independent events of the same likelihood should convey twice as much information as learning about the occurrence of only one of these events.

3.1.1 Self-information

The above mentioned requirements are satisfied by the so-called self-information $\mathcal{I}[p(x)]$ as defined by [Goodfellow et al., 2016, p.73], namely

$$\mathcal{I}[p(x)] = -\log\left[p(x)\right], \tag{3.1}$$

with X being a random variable and $p(x)$ the probability density function thereof.

By this definition, information $\mathcal{I}[p(x = U)]$ of an unlikely event $x = U$ will be higher than information $\mathcal{I}[p(x = L)]$ of a likely event $x = L$; this can be demonstrated by assessing the self-information contents of both events. Because U is said to be an unlikely event, $p(U) \approx 0$ will hold and therefore $\mathcal{I}[p(x = U)] = -\log[p(U)] = \log[1/p(U)] \gg 1$. Since L is a likely event, the probability $p(L)$ of L occurring is estimated by $p(L) \approx 1$ and the self-information $\mathcal{I}[p(x = L)]$ will therefore be very low, because $\mathcal{I}[p(x = L)] = -\log[p(L)] = \log[1/p(L)] \approx 0$ holds. Moreover, information about two independent events $x = A$ and $x = B$ is additive, as the calculation

$$\begin{aligned} \mathcal{I}[p(A \cap B)] &= -\log[p(A \cap B)] = -\log[p(A)p(B)] \\ &= -(\log[p(A)] + \log[p(B)]) = \mathcal{I}[p(A)] + \mathcal{I}[p(B)] \end{aligned} \tag{3.2}$$

shows it to be the case.

3.1.2 Information entropy

One can describe the information content of a probability density function by the so-called information entropy given by [Goodfellow et al., 2016, p.74] as

$$H_p[p(x)] = E_{x\sim p}(\mathcal{I}[p(x)]) = -E_{x\sim p}\left(\log[p(x)]\right), \tag{3.3}$$

where $p(x)$ is a probability density function of a random variable X. If the random variable X is of a discrete nature, the information entropy is called Shannon entropy, whereas if X is continuous, the information entropy goes by the name of differential entropy.

Information entropy defined in this way is a measure of the amount of uncertainty in a probability distribution. The information entropy will

be very low, if $p(x)$ describes mostly very likely events, whereas it will be very high, if $p(x)$ describes mainly very unlikely events. This can also be stated in the following way: Probability distributions that are nearly deterministic (delta-function shaped) have a low entropy, whereas distributions that are closer to a uniform distribution have high entropy.

The field of artificial intelligence and especially its sub-field machine learning are concerned with learning the available self-information from the data. That is to say, the information entropy initially present within the machine learning entity is iteratively reduced by means of training it on a dataset, thus resulting in learned information on the part of the trained artificial intelligence.

3.2 Overview of artificial intelligence

According to [Amini, 2023, p.7], the field of artificial intelligence can be broadly described as any computational technique, that enables a computer to engage with information in a reasonable way, such as to mimic how a human would process the available data or information. Machine learning is more specialized sub-field of artificial intelligence, which is concerned with any routine, that gives a computer the ability to learn a task without being explicitly programmed for that particular task. Deep learning, which is ultimately about reflecting and rebuilding a simple representation the human brain working principle, is an even more specialized sub-field of artificial intelligence, which is itself a sub-field of machine learning.

The foundational tool of deep learning is the so-called neural network, also known as deep neural network. The basic building block of a neural network is the single layer perceptron (SLP). By combining several single layer perceptrons to a multi layer perceptron (MLP), a deep neural network can be constructed. Given an input vector $\boldsymbol{x} = (x_1, \ldots, x_d)^\top \in \mathbb{R}^d$ and an output vector $\bar{\boldsymbol{x}} = (\bar{x}_1, \ldots, \bar{x}_w)^\top \in \mathbb{R}^w$ with $d, w \in \mathbb{N}$, the working principle of single layer perceptron is shown in fig. 3.1, similarly for a multi layer preceptron in fig. 3.2.

3.3 Neural networks

3.3.1 Single layer perceptron

In fig. 3.1, a single layer perceptron is visualized. The quantity $\boldsymbol{x} = (x_1, \ldots, x_d)^\top \in \mathbb{R}^d$ with $d \in \mathbb{N}$ is an arbitrary input vector. The single

layer perceptron generates a single output $\bar{x}$. First, all input vector components x_i are multiplied with initially arbitrary weights $\phi_i \in \mathbb{R}$ for all $i \in \{1, \ldots, d\}$ and subsequently summed up to some sum $\tilde{s} = \sum_{i=1}^{d} x_i \phi_i$; then, an arbitrary bias value $b \in \mathbb{R}$ is added to the sum $\tilde{s}$ such as to arrive at a new sum $s = b + \tilde{s}$. Finally, an output scalar $\bar{x} = a(s)$ is generated by means of pushing the beforehand attained sum s through a so-called activation function $a : \mathbb{R} \to \mathbb{R}, s \mapsto a(x)$, which is nothing more than some arbitrary nonlinear function, which introduces nonlinearities to the otherwise linear operations; this enables the perceptron to learn nonlinear correlations. All quantities involving the letters b and ϕ

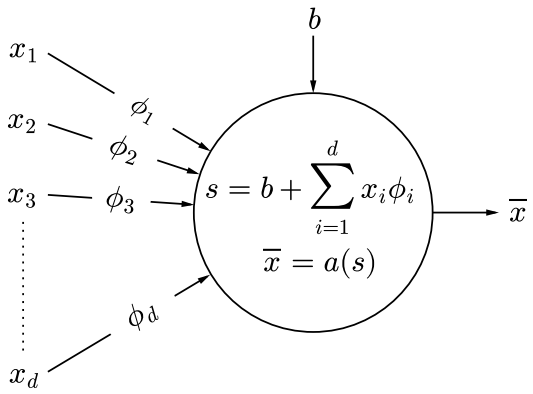


Figure 3.1: Working principle of a single layer perceptron; hereby, a represents a nonlinear activation function $a : \mathbb{R} \to \mathbb{R}$, $s \mapsto a(s)$.

compose the total vector $\boldsymbol{\phi}$ of weights for the perceptron.

3.3.2 Multi layer perceptron

A multi layer perceptron is the simplest form of a neural network. In fig. 3.2, such a multi layer perceptron is visualized. Again, $\boldsymbol{x} = (x_1, \ldots, x_d)^\top$ as an element of $\mathbb{R}^d$ with $d \in \mathbb{N}$ is an arbitrary input vector. Compared to the single layer perceptron, the multi layer perceptron however generates a vectorial output $\bar{\boldsymbol{x}} = (\bar{x}_1, \ldots, \bar{x}_w)^\top \in \mathbb{R}^w$ with $w \in \mathbb{N}$. Now, all input components x_i are connected to more than one perceptron; thus all perceptrons directly connected to the input components compose a so-called network layer, the first layer. There can be one or more network layers; generally there are $n \in \mathbb{N}$ layers in a neural network. At each perceptron of a certain fixed network layer, a single output is generated as with the single layer perceptron. All of these single outputs together

compose a new vector, which serves as the input vector for the next network layer composed of perceptrons. This general working principle is better explained visually, consider therefore fig. 3.2. Again, all quantities

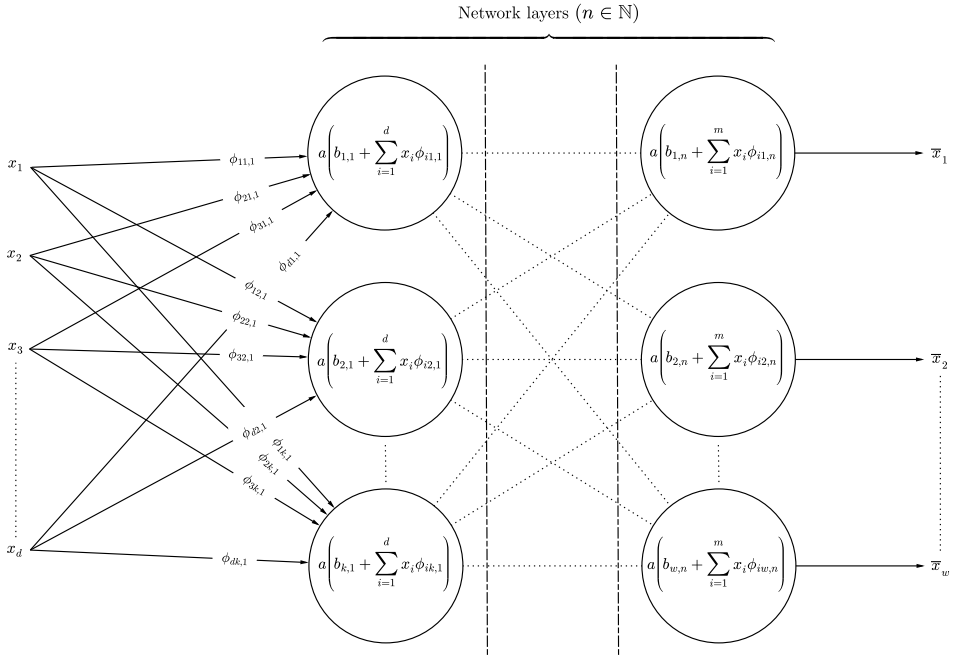


Figure 3.2: Working principle of a multi layer perceptron; hereby, a represents a nonlinear activation function $a : \mathbb{R} \to \mathbb{R}$, $s \mapsto a(s)$.

with letters b or ϕ compose the total weights vector $\boldsymbol{\phi}$ of the multi layer perceptron, which is the simplest possible neural network involving more than one perceptron.

3.3.3 Training process for neural networks

In order for a neural network to learn anything, it has to be conditioned by data, whose interpretation is known. This process is generally referred to as training of a neural network and requires a mathematical framework that allows for optimization. During the training process of a neural network, one would like to achieve the best possible performance of the neural network in interpreting the training data correctly, that is to say, that the input data $\boldsymbol{x}$ maps to the known output data $\bar{\boldsymbol{x}}$ with minimal discrepancies.

The mathematical framework for such an optimization relies on the definition of a loss function $\mathcal{L}_{\boldsymbol{\phi}}$. Such a loss function can be implemented

in many different ways; for example on can just take the norm of the difference between input data $\boldsymbol{x}$ and output data $\bar{\boldsymbol{x}}$ as a loss function, i.e. $\mathcal{L}_{\boldsymbol{\phi}(\hat{\mathbf{x}})} = \|\boldsymbol{x} - \bar{\boldsymbol{x}}\|$, where $\hat{\mathbf{x}}$ is a matrix of training data and $\boldsymbol{\phi}(\hat{\mathbf{x}})$ the associated weights of the neural network described by this loss function. However, the loss function depends on the task at hand and needs to be chosen accordingly, such that it represents a suitable measure on how well a neural network learns from the training data. Once one has defined a loss function $\mathcal{L}_{\boldsymbol{\phi}}$, the key to training a neural network resides within something called gradient descent. The idea behind gradient descent is quite simple; one just calculates the gradient of the loss function $\mathcal{L}_{\boldsymbol{\phi}}$ with respect to the weights $\boldsymbol{\phi}$, i.e. $\boldsymbol{\nabla}_{\boldsymbol{\phi}}\mathcal{L}_{\boldsymbol{\phi}}$ and updates all weights such that one moves into the direction of negative gradient. Iteratively repeating this process, eventually a local or global minimum of the loss function is found. In terms of mathematics, this procedure translates as follows. Consider some starting point $\boldsymbol{\phi}_i$ and the loss function at this point, i.e. $\mathcal{L}_{\boldsymbol{\phi}_i}$. The gradient $\boldsymbol{\nabla}\mathcal{L}_{\boldsymbol{\phi}}$ of the loss function with respect to the weights $\boldsymbol{\phi}$ indicates, in which direction of $\boldsymbol{\phi}$ the loss function increases or decreases. The negative gradient points into the direction of a local/global minimum of $\mathcal{L}_{\boldsymbol{\phi}}$, whereas the positive gradient points towards a local/global maximum. Hence, $\boldsymbol{\phi}_{i+1} = \boldsymbol{\phi}_i - \gamma\boldsymbol{\nabla}_{\boldsymbol{\phi}}\mathcal{L}_{\boldsymbol{\phi}_i}$ with $i \in \mathbb{N}$ and $\gamma \in \mathbb{R}$ a suitable step size provides a rule for eventually finding a global or local minimum of the loss function. One can therefore define an iterative procedure as

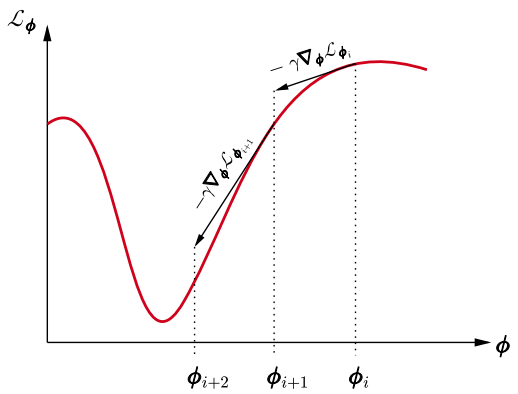


Figure 3.3: Visualization of a gradient descent routine. Note, that the horizontal axis is in reality multidimensional, whereas it is pictured here as just being one-dimensional. The vertical axis represents the value of the loss function $\mathcal{L}_{\boldsymbol{\phi}}$ in dependence of the weights $\boldsymbol{\phi}$.

follows:

(1) Take a start value $\boldsymbol{\phi}_0$ and calculate the value of the loss function

$\mathcal{L}_{\boldsymbol{\phi}_0}$ at this point.

(2) Update the weights $\boldsymbol{\phi}_i$ according to the rule

$$\boldsymbol{\phi}_{i+1} = \boldsymbol{\phi}_i - \gamma \boldsymbol{\nabla}_{\boldsymbol{\phi}} \mathcal{L}_{\boldsymbol{\phi}_i} \tag{3.4}$$

for $i \in \mathbb{N}$ individual steps and $\gamma \in \mathbb{R}$ a suitable step size as long as $\epsilon \leq |\mathcal{L}_{\boldsymbol{\phi}_{i+1}} - \mathcal{L}_{\boldsymbol{\phi}_i}|$ holds for $\epsilon \in \mathbb{R}$ an accuracy threshold.

(3) Terminate, if $\epsilon > |\mathcal{L}_{\boldsymbol{\phi}_{i+1}} - \mathcal{L}_{\boldsymbol{\phi}_i}|$.

This iterative process is found visualized in fig. 3.3; note, that this is a simplified picture, as the horizontal axis would in reality be multidimensional.

3.3.4 Learning curves

It is the goal of any entity of machine learning to learn from data in such a way, that it enables the machine learning entity to make as accurate as possible predictions on prior unseen data. [Goodfellow et al., 2016, p.110] speak in this context of generalization, by which they mean the ability of a machine learning model to perform well on previously unobserved data. In order for a machine learning model to do that, it has to learn model parameters and mathematical operations to predict an outcome, given some input. It is crucial to a good performance of a machine learning model on prior unseen data, that the model learns those parameters and mathematical operations such that there is neither overfitting, underfitting or underrepresentation of training and testing data.

Overfitting is the case, where a model has learned too many parameters, such that it performs extremely well on the training data, but increasingly bad on testing data - that is to say, on prior unseen data to the model. The contrary - underfitting - means, that a model has not learned enough parameters to be able to generalize to prior unseen data.

Apart from overfitting, underfitting or underrepresentation issues, one can certainly ask if it matters, which particular model Ansatz one chooses for a particular task. In fact, [Goodfellow et al., 2016, p.116] note that the so-called no free lunch theorem states, that every machine learning classification algorithm generalized equally well (or bad), if averaged over all possible data generating probability distributions. But this is not to say, that every machine learning algorithm performs equally well or bad on a particular task - this is not the case. Hence, one must choose the most appropriate machine learning model for the task at hand. Most

appropriate in this context is to say, that the chosen model should generalize better than any other available model Ansatz.

Once one has chosen a machine learning model for a particular task, the question of how to ensure, that the model is trained on data in such a way, that it generalizes best to prior unobserved data. Methods to take care of this are generally known as regularization, as [Goodfellow et al., 2016, p.120] explain.

However, besides regularization, tuning the so-called hyperparameters of a machine learning algorithm can improve its generalization properties. Hyperparameters are just settings of the learning algorithm for a machine learning entity. In order to find optimal hyperparameters, one can either do a so-called grid search algorithms can be applied; or one can just inspect the learning curves for different settings of hyperparameters.

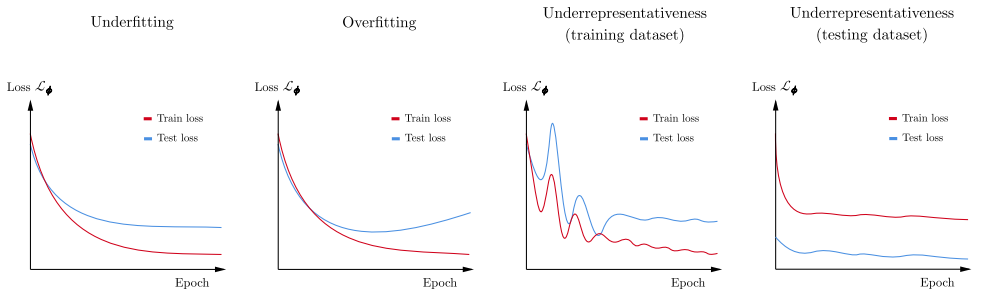


Figure 3.4: Examples for the four basic cases of non-ideal model performance. Sketches inspired by Jason Brownlee, `https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/`, accessed on December 18, 2023.

The concepts of underfitting, overfitting or underrepresentation of either training or testing data for the learning process of a machine learning model reflect in the learning curves as sketched in fig. 3.4.

As underfitting means, that the model has not learned enough parameters to generalize to unseen data, it reflect in a learning curve such that the train loss is notoriously lower than the test loss and that there is an approximately constant gap between the train and test loss curves after several epochs. This situation is shown on the far left panel of fig. 3.4.

If however the test loss steadily increases as the train loss continuously decreases, it is very likely that there is overfitting in the model; that is to say, that the model has learned too many model parameters, such that it performs very well on the training data, but increasingly worse on the test data. The situation of overfitting is shown in the second from left

panel of fig. 3.4.

Furthermore, there can also be two cases of underrepresentativeness of either the training dataset or the testing dataset, both of which are visualized in the two panels to the right in fig. 3.4. If the training dataset is not representative of the testing dataset and hence generally on prior unseen data, one would expect that the test loss is higher than the train loss. This is indeed the case for an underrepresentative training dataset; additionally, there can be large fluctuations in the learning curves, since the gradient descent algorithm jumps over local or global minima of the loss function more easily, if the training data has ≪gaps≫. On the other hand, if the testing dataset is underrepresentative, one would expect that the test loss is notoriously lower than the train loss - because the average performance of a model trained on a diverse and therefore representative amount of data on a smaller, underrepresentative dataset will be better. That again is the case as shown in the far right panel of fig. 3.4, where again fluctuations are to be expected due to the mentioned reason.

Another factor that can lead to a behaviour a shown in the second to right panel of fig. 3.4 is a learning rate, which is chosen to be too high. If the learning rate is set to too large a rate, the gradient descent process optimizing the model parameters can ≪jump≫ over minima, leading to an oscillatory behaviour of the train and test loss.

An ideal learning curve is shown in fig. 3.5. It features a continu-

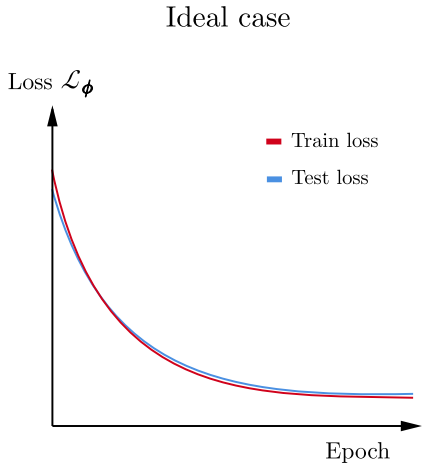


Figure 3.5: Example of an ideal learning curve.

ously decreasing train and test loss, which eventually seem to converge to a constant value after several epochs and where furthermore the gap between the train and test losses is very small. An ideal learning curve

indicates, that the hyperparameter setting for a machine learning model is optimal or at least well-chosen.

Analyzing learning curves visually using these tools can lead to finding optimal hyperparameters for machine learning model and especially for neural networks. Inspecting learning curves can also help to identify possible issues concerning the data itself, for example if the data is underrepresentative of the general diversity of data or not.

3.4 Deep generative models

Generative models - as their name already suggests - generate data. These models learn how to do so by extracting defining features of the data they are trained on, such as the data's probability density. Consider a matrix of training data $\hat{\mathbf{x}}$, where the individual columns follow a probability distribution $\boldsymbol{x} \sim p_{\boldsymbol{x}}(\boldsymbol{x})$. According to [Raschka, 2022, p.593], a generative model in deep learning is then defined such, that it learns parameters $\boldsymbol{\phi}(\hat{\mathbf{x}})$ to construct a probability density function $p_{\boldsymbol{\phi}(\hat{\mathbf{x}})}(\boldsymbol{x}) \approx p_{\boldsymbol{x}}(\boldsymbol{x})$. This is to say, that one can then draw samples from the learned probability density $p_{\boldsymbol{\phi}(\hat{\mathbf{x}})}(\boldsymbol{x})$, which essentially are generated data imitating the training data.

One can distinguish at least three basic architectures of generative models, namely variational autoencoder networks (VAE's), generative adversarial networks (GAN's) and flows. These types of generative models are visualized in fig. 3.6, fig. 3.7 and fig. 3.8. The parallelograms used to visualize the working principles of these models represent dimensionality reduction/increase of an input variable compared to an output variable; if the parallelogram narrows towards the output, it means dimensionality reduction, whereas if it broadens towards the output, increase in dimensionality is visualized.

3.4.1 VAE's

Variational autoencoders work as sketched in fig. 3.6. A variational autoencoder consists of an encoder and a decoder. The encoder implements a conditional probability distribution $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ for a latent variable $\boldsymbol{z}$, based on an input variable $\boldsymbol{x}$, where $\boldsymbol{\phi}$ are the weights of the neural network constituting the encoder. The decoder however models the probability $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ of the output variable $\boldsymbol{x}$ based on the latent variable $\boldsymbol{z}$, where again $\boldsymbol{\theta}$ are weights associated to the neural network of the decoder. The variable $\boldsymbol{x}$ represents the input variable with probability

density $\boldsymbol{x} \sim p_{\boldsymbol{x}}(\boldsymbol{x})$, whereas the variable $\boldsymbol{z}$ is a latent variable and is chosen to fit the probability density of a standard normal distribution, i.e. $\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z}) = \mathcal{N}(0,1)^d$, where $d \in \mathbb{N}$ is the dimension of the latent variable.

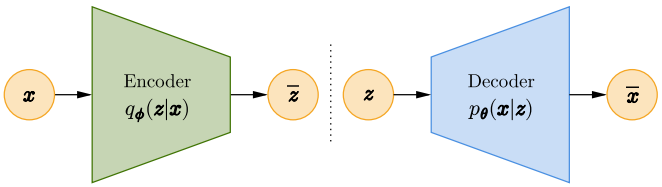


Figure 3.6: Working principle of a variational autoencoder (VAE). Figure inspired by [Weng, 2018].

A variational autoencoder is trained in such a way, that the encoder learns how to transform an input $\boldsymbol{x}$ such that the output $\bar{\boldsymbol{z}}$ follows the chosen latent probability density. Furthermore, the decoder has to learn, how to transform a variable $\boldsymbol{z}$ following a standard normal distribution to an output $\bar{\boldsymbol{x}}$, such that the output variable follows the probability density of the original input variable $\boldsymbol{x}$.

A VAE in principle can be said to learn the most important information from the training data $\hat{\mathbf{x}}$, as it has to pass this information from the input variable $\boldsymbol{x}$ through a low-dimensional bottleneck to $\bar{\boldsymbol{z}}$, from which again the original input $\bar{\boldsymbol{x}}$ has to be reconstructed as best as possible, such that $\boldsymbol{x} \approx \bar{\boldsymbol{x}}$. Once the VAE is trained sufficiently, one can just use the decoder to generate outputs $\bar{\boldsymbol{x}}$ based on $\boldsymbol{z}$ sampled from a multidimensional standard normal distribution.

A VAE could be used for stellar atmosphere inversion purposes, but it has several disadvantages in this regard compared to normalizing flows. First of all, VAE's have to be trained using an evidence lower bound (ELBO), because the exact likelihood of $\boldsymbol{x}$ given $\boldsymbol{z}$ is not tractable. A normalizing flow however allows for exact likelihood estimation of $\boldsymbol{x}$ given $\boldsymbol{z}$ because of its architecture. Furthermore, again because of its inherent architecture, a normalizing flow allows to perform exact inversions of data, since it is basically built of invertible functions. This is not the case for VAE's; yes, encoder and decoder are sort of inverse to one another, but it cannot be an exact inversion in the mathematical sense, since the VAE relies on a smaller dimension of the latent variable and the input/output variable.

3.4.2 GAN's

Generative adversarial networks function as sketched in fig. 3.7. A generative adversarial network consists of a generator and a discriminator. The generator is basically implements a function $\bar{\boldsymbol{x}} = \boldsymbol{G}_{\boldsymbol{\phi}}(\boldsymbol{z})$, where $\bar{\boldsymbol{x}}$ is the output of the generator, $\boldsymbol{\phi}$ are the weights associated to the neural network implementing the generator function and $\boldsymbol{z}$ is a latent variable, which is chosen to follow a standard normal distribution, i.e. $\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z}) = \mathcal{N}(0,1)^d$ with $d \in \mathbb{N}$ again the dimensionality of the latent space. The purpose of the generator is to generate fake data $\bar{x}$ based on samples $\boldsymbol{z}$ of the latent probability density $p_{\boldsymbol{z}}(\boldsymbol{z})$. The discriminator however takes an input $\boldsymbol{x}$, such as the output of the generator, and outputs a value 0 or 1, where 0 stands for «input is fake» and 1 represents the situation «input is real». The task of the discriminator is therefore to decide, if an input $\boldsymbol{x}$ is fake or real.

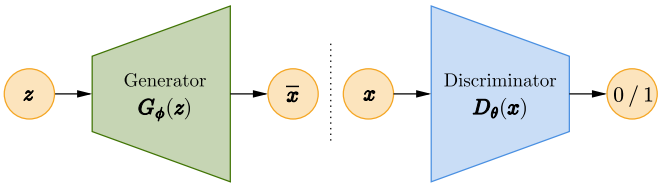


Figure 3.7: Working principle of a generative adversarial network (GAN). Figure inspired by [Weng, 2018].

A generative adversarial network is trained in such a way, that the generator learns how to generate the best possible real-looking data $\bar{\boldsymbol{x}}$, whereas the discriminator learns as good as possible to distinguish between real data contained in the training set $\hat{\mathbf{x}}$ and fake data $\bar{\boldsymbol{x}}$ generated by the generator. Herein lies the adversarity of the model; the generator and the discriminator are actually competing against each other; the generator learns how to create the best possible fake data, whereas the discriminator learns, how to best distinguish between real and fake data - in this way, both the generator and the discriminator are optimized for their respective task. The loss function for training therefore has to be defined accordingly in a game-theoretical manner, where both the discriminator and the generator try to optimize their outcomes. Once trained, one can just use the generator to create fake data $\bar{\boldsymbol{x}} = \boldsymbol{G}_{\boldsymbol{\phi}}(\boldsymbol{z})$ from samples $\boldsymbol{z}$ of $p_{\boldsymbol{z}}(\boldsymbol{z})$ at will.

Despite the many interesting applications of GAN's, they cannot really be used for stellar atmosphere inversion purposes, since they do not

suit the problem at hand, since stellar inversion problems do not involve classification between real and fake data.

3.4.3 Flows

Flow-based generative models work as sketched in fig. 3.8. A flow just consists of a bijective function $\bar{\boldsymbol{z}} = \boldsymbol{f}_{\boldsymbol{\phi}}(\boldsymbol{x})$, where $\boldsymbol{\phi}$ refers to the weights of the neural network(s) used to implement this function. One of the main differences between flows and GAN's on the one hand and VAE's on the other hand is the fact, that flows are bijective in architectural nature, while GAN's and VAE's are not. This is to say that they are perfectly adapted to an inversion problem, such as a stellar atmosphere inversion task. Because of the bijective nature of flows, there is no dimensionality reduction or increase from input/output to latent space.

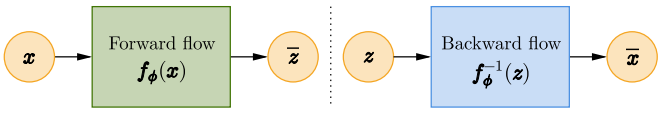


Figure 3.8: Working principle of a flow-based generative model. Figure inspired by [Weng, 2018].

Flows can be trained by exact likelihood calculation; that is to say, the loss function can be mathematically derived and implemented using the change of variable theorem from probability calculus in combination with the specific bijective transformations constituting the flow. A flow is called a normalizing flow, if the latent probability density function is chosen as a standard normal distribution - hence an arbitrary probability density $p_{\boldsymbol{x}}(\boldsymbol{x})$ of input data is mapped into the density of a standard normal distribution. In that way, the input data can be said to be normalized to the latent space. As flows and especially normalizing flows are best suited to stellar atmosphere inversions compared to the other available generative models, this technique will be the tool of choice in the material to follow.

3.5 Introduction to normalizing flows

Normalizing flows are useful to learn complex data distributions from available samples, be they experimentally obtained or of a synthesized nature.

The goal of a normalizing flow is to learn a certain probability density function $p_{\boldsymbol{x}}(\boldsymbol{x})$ of some quantity represented by a potentially multivariate random variable $\boldsymbol{x}$. Given samples from $p_{\boldsymbol{x}}(\boldsymbol{x})$ obtained by either experimental or synthetical procedure, the normalizing flow can learn the probability density function by means of finding a diffeomorphism, which maps samples $\boldsymbol{z}$ from a standard normal distribution to $p_{\boldsymbol{x}}(\boldsymbol{x})$. Since a diffeomorphism is invertible by definition, a trained normalizing flow can then be used to draw samples from $p_{\boldsymbol{x}}(\boldsymbol{x})$ at will; this can be done by sampling from a standard normal distribution and subsequent application of the inverse learned diffeomorphism to the generated samples. This will result in a set of samples from $p_{\boldsymbol{x}}(\boldsymbol{x})$.

This technique can be used for various purposes; for example, for inversion problems in (stellar) atmospheric physics, as it is being explored in the subsequent material.

3.5.1 Why use normalizing flows for inversions?

If one has an analytical model $\boldsymbol{y} = \boldsymbol{M}(\boldsymbol{x})$ of a stellar atmosphere, it can usually be inverted too, such that $\boldsymbol{x} = \boldsymbol{M}^{-1}(\boldsymbol{y})$. This raises the question, why one should even bother using normalizing flows to learn such inversions.

First of all, only simple models of radiative transfer, such as the Milne-Eddington atmosphere can be inverted analytically. More complex models require resource-demanding calculations by means of Monte Carlo methods to do inversions. The advantage of a normalizing flow resides in the fact, that inversions done by it would be comparatively much faster, once the normalizing flow is trained.

A second incentive to use normalizing flow for atmospheric inversions is, that a normalizing flow not only provides one with one solution to the inversion problem for each atmospheric parameter, but rather with a probability distribution for each parameter. This allows for error estimation and the determination of the most likely solution to an inversion problem, if there is more than one solution possible.

3.6 Change of variable formula in probability theory

The change of variable formula can be considered as the heart of the normalizing flow technique, since it guides the implementation of a loss function to train a flow. Because of its importance for normalizing flows,

a full proof of this theorem will therefore be given for the univariate case, which will be generalized to the multivariate case subsequently.

Theorem 3.6.1. *Let x and z be continuous random variables with associated probability density functions $p_x(x)$ and $p_z(z)$. Furthermore, let z be the transformation of x by an invertible and strictly increasing continuously differentiable function $f(x)$, such that $z = f(x)$ and $x = f^{-1}(z)$ hold; in this case the change of variable formula from probability theory applies as*

$$p_x(x) = p_z(f(x))\frac{\mathrm{d}f(x)}{\mathrm{d}x}. \tag{3.5}$$

Proof. First, let with a and b be defined real numbers with $a < b$. By definition of the transformation $f(x)$ let with c and d be defined the real numbers $c = f(a)$ and $d = f(b)$, hence

$$P(a \leq x \leq b) = P(c \leq z \leq d) \tag{3.6}$$

must hold, where this quantity is defined as $P(a \leq x \leq b) \doteq \int_a^b p_x(x)\,\mathrm{d}x$ and therefore represents the probability mass for an experiment outcome of x to be between a and b. Since x and z are continuous random variables with associated valid probability density functions, one can write

$$P(c \leq z \leq d) = \int_c^d p_z(z)\,\mathrm{d}z = \int_{f(a)}^{f(b)} f_z(z)\,\mathrm{d}z. \tag{3.7}$$

By the substitution rule for integrals one can substitute in the above integral $z(x) = f(x)$ and

$$\frac{\mathrm{d}z(x)}{\mathrm{d}x} = \frac{\mathrm{d}f(x)}{\mathrm{d}x} \quad \Leftrightarrow \quad \mathrm{d}z = \mathrm{d}z(y) = \frac{\mathrm{d}f(x)}{\mathrm{d}x}\,\mathrm{d}x. \tag{3.8}$$

Therefore, the relation

$$\begin{aligned} P(c \leq z \leq d) &= \int_{f(a)}^{f(b)} p_z(z)\,\mathrm{d}z \\ &= \int_a^b p_z(f(x))\frac{\mathrm{d}f(x)}{\mathrm{d}x}\,\mathrm{d}x = P(a \leq x \leq b) \end{aligned} \tag{3.9}$$

obtains, thus

$$p_x(x) = p_z(f(x))\frac{\mathrm{d}f(x)}{\mathrm{d}x} \tag{3.10}$$

must hold, where ${}^{\mathrm{d}f(x)}/_{\mathrm{d}x}$ acts as a normalizing factor for the new probability density $p_x(x)$. □

Let now $\boldsymbol{x}, \boldsymbol{z} \in \mathbb{R}^d$, $d \in \mathbb{N}$ be continuous multivariate random variables with associated probability density functions $p_{\boldsymbol{x}}(\boldsymbol{x})$ and $p_{\boldsymbol{z}}(\boldsymbol{z})$. Furthermore, let

$$\boldsymbol{f} : \mathbb{R}^d \to \mathbb{R}^d, \quad \boldsymbol{x} \mapsto \boldsymbol{z} = \boldsymbol{f}(\boldsymbol{x}) \tag{3.11}$$

be a diffeomorphism transforming $\boldsymbol{x}$ to $\boldsymbol{z}$. In this case, the formula generalizes to the multivariate case as

$$\begin{aligned} p_{\boldsymbol{x}}(\boldsymbol{x}) &= p_{\boldsymbol{z}}[\boldsymbol{f}(\boldsymbol{x})] \left| \det \left(\frac{\partial \boldsymbol{f}(\boldsymbol{x})}{\partial \boldsymbol{x}} \right) \right| \\ &= p_{\boldsymbol{z}}[\boldsymbol{f}(\boldsymbol{x})] \left| \det \begin{pmatrix} \frac{\partial f_1(\boldsymbol{x})}{\partial x_1} & \cdots & \frac{\partial f_1(\boldsymbol{x})}{\partial x_d} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_d(\boldsymbol{x})}{\partial x_1} & \cdots & \frac{\partial f_d(\boldsymbol{x})}{\partial x_d} \end{pmatrix} \right| . \end{aligned} \tag{3.12}$$

3.7 General principle of a normalizing flow

3.7.1 Conceptual formulation of a normalizing flow

Consider some vector $\boldsymbol{x} \in \mathbb{R}^d$ with d, $q \in \mathbb{N}$ and a latent vector $\boldsymbol{z} \in \mathbb{R}^d$. The probability densities for these vectors shall be given by

$$\boldsymbol{x} \sim p_{\boldsymbol{x}}(\boldsymbol{x}), \quad \boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z}) = \mathcal{N}(0,1)^d. \tag{3.13}$$

The aim of a normalizing flow is to learn a diffeomorphism

$$\boldsymbol{f}_{\boldsymbol{\phi}(\hat{\mathbf{x}})} : \mathbb{R}^d \to \mathbb{R}^d, \quad \boldsymbol{x} \mapsto \boldsymbol{z} = \boldsymbol{f}_{\boldsymbol{\phi}(\hat{\mathbf{x}})}(\boldsymbol{x}) \tag{3.14}$$

transforming the probability density function $p_{\boldsymbol{x}}$ to $p_{\boldsymbol{z}}$. The conceptual principle of a normalizing flow is visualized in fig. 3.9. Hereby, $\boldsymbol{\phi}(\hat{\mathbf{x}})$ are weights depending on training data $\hat{\mathbf{x}} \in \mathbb{R}^{d \times q}$, which are associated to a deep neural network implementing the diffeomorphic function $\boldsymbol{f}_{\boldsymbol{\phi}(\hat{\mathbf{x}})}$. Given such a function, using the change of variable formula the probability density $p_{\boldsymbol{x}}(\boldsymbol{x})$ can be calculated as

$$p_{\boldsymbol{x}}(\boldsymbol{x}) = p_{\boldsymbol{z}}[\boldsymbol{f}_{\boldsymbol{\phi}(\hat{\mathbf{x}})}(\boldsymbol{x})] \left| \det \left(\frac{\partial \boldsymbol{f}_{\boldsymbol{\phi}(\hat{\mathbf{x}})}(\boldsymbol{x})}{\partial \boldsymbol{x}} \right) \right| . \tag{3.15}$$

Knowing the transformation $\boldsymbol{z} = \boldsymbol{f}_{\boldsymbol{\phi}(\hat{\mathbf{x}})}(\boldsymbol{x})$ and given that $\boldsymbol{z} \sim \mathcal{N}(0,1)^d$, one can easily calculate samples from the distribution $p_{\boldsymbol{x}}(\boldsymbol{x})$ by means of sampling $\boldsymbol{z}$ from a d-dimensional standard normal distribution and obtaining the corresponding $\boldsymbol{x}$ plugging $\boldsymbol{z}$ into the inverse function $\boldsymbol{f}^{-1}_{\boldsymbol{\phi}(\hat{\mathbf{x}})}$, such that $\boldsymbol{x} = \boldsymbol{f}^{-1}_{\boldsymbol{\phi}(\hat{\mathbf{x}})}(\boldsymbol{z})$.

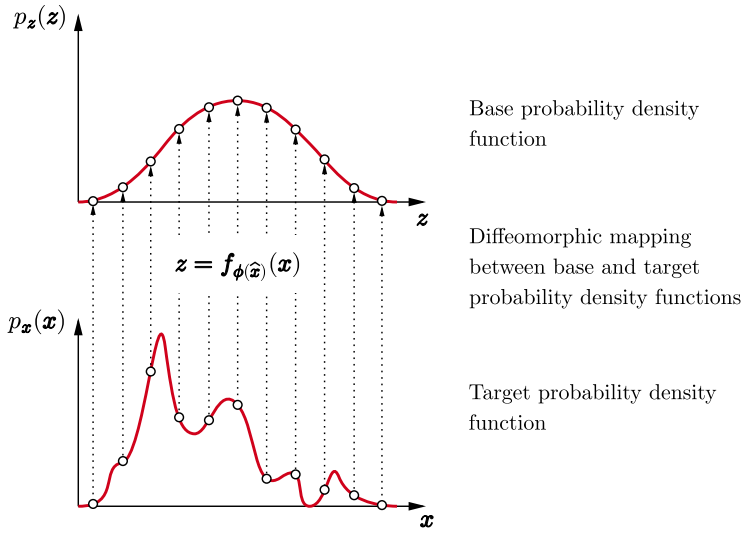


Figure 3.9: Conceptual principle of a normalizing flow $\boldsymbol{z} = \boldsymbol{f}_{\boldsymbol{\phi}(\hat{\mathbf{x}})}(\boldsymbol{x})$ transforming a target probability density function $p_{\boldsymbol{x}}(\boldsymbol{x})$ to a base probability density function $p_{\boldsymbol{z}}(\boldsymbol{z})$.

3.7.2 Requirements for a normalizing flow

Consider a function

$$\boldsymbol{f} : \mathbb{R}^d \to \mathbb{R}^d, \quad \boldsymbol{x} \mapsto \boldsymbol{z} = \boldsymbol{f}(\boldsymbol{x}) \tag{3.16}$$

composed of several functions $\boldsymbol{f}_{(k)}$, $k \in \{1, \ldots, n\}$, $n \in \mathbb{N}$, such that

$$\boldsymbol{z} = \boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{f}_{(n)} \circ \cdots \circ \boldsymbol{f}_{(1)}(\boldsymbol{x}). \tag{3.17}$$

In this case, the transformation functions $\boldsymbol{f}_{(k)}$ aswell as the function $\boldsymbol{f}$ as a whole should satisfy several conditions to serve purposefully as a tool to sample from a probability density $p_{\boldsymbol{x}}(\boldsymbol{x})$ given samples $\boldsymbol{z}$ obtained from a standard normal distribution $p_{\boldsymbol{z}}(\boldsymbol{z})$. According to [Kobyzev et al., 2021], all functions $\boldsymbol{f}_{(k)}$ should satisfy the following conditions:

(1) All transformation functions $\boldsymbol{f}_{(k)}$ need to be invertible and differentiable. In order to transform one probability density into another density, it is necessary to calculate the Jacobian of the transformation function, as stated by the change of variable theorem. Furthermore, the transformation functions should be invertible, because one would like to sample from a standard normal distribution, plug the obtained values $\boldsymbol{z}$ into the inverse transformation function and thereby obtaining samples $\boldsymbol{x}$. Both of these conditions - invertibility and differentiabiliby - are satisfied by diffeomorphic functions.

(2) The transformation functions $\boldsymbol{f}_{(k)}$ need to be expressive enough to model real data distributions. The normalizing flow composed of these transformation functions needs to learn a real data distribution; hence enough flexibility of the transformation functions is required in order to allow the neural networks composing the flow to learn actual and potentially complicated probability distributions.

(3) The transformation functions $\boldsymbol{f}_{(k)}$ should be computationally efficient. That is, the Jacobian determinant should be calculable in an as efficient as possible way; furthermore also the application of the trained forward and inverse normalizing flow should consume as little as possible time. In order to train and apply the normalizing flow, i.e. the transformation functions composing the normalizing flow, many Jacobian determinants need to be calculated, as required by the change of variable formula. Therefore, transformation functions with a simple Jacobians allowing for quick and easy calculations of the corresponding determinants are preferable.

3.7.3 Normalizing flows with conditioning data

Consider a model $\boldsymbol{M}(\boldsymbol{x}) = \boldsymbol{y}$, that produces observations $\boldsymbol{y} \in \mathbb{R}^D$, $D \in \mathbb{N}$ based on model parameters $\boldsymbol{x} \in \mathbb{R}^d$, d, $q \in \mathbb{N}$. Furthermore, consider a latent vector $\boldsymbol{z} \in \mathbb{R}^d$. The probability densities for these vectors are given by

$$\boldsymbol{x} \sim p_{\boldsymbol{x}}(\boldsymbol{x}), \quad \boldsymbol{y} \sim p_{\boldsymbol{y}}(\boldsymbol{y}), \quad \boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z}) = \mathcal{N}(0,1)^d. \tag{3.18}$$

The aim of the normalizing flow is to model the probability density $p_{\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{x}|\boldsymbol{y})$, namely the probability density function of $\boldsymbol{x}$, given a certain observation $\boldsymbol{y}$, thereby defining an inversion of the model $\boldsymbol{M}(\boldsymbol{x}) = \boldsymbol{y}$, see fig. 3.10 for a visualization of a normalizing flow with conditioning data; which from now on will be called a conditional normalizing flow. The multidimensional quantity $\boldsymbol{\phi}(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ is a set of parameters learned by a neural network, which is used to implement the normalizing flow.

If there is no conditioning data $\boldsymbol{y}$ available, the normalizing flow can directly be applied to learn the probability distribution $p_{\boldsymbol{\phi}(\hat{\mathbf{x}})}(\boldsymbol{x})$ based on standard normally distributed data $z \sim p_{\boldsymbol{z}}(\boldsymbol{z}) = \mathcal{N}(0,1)^d$. In this case, the conditioning data $\boldsymbol{y}$, i.e. the symbols $\boldsymbol{y}$ and $\hat{\mathbf{y}}$, can be omitted in all formulae pertaining to the current chapter of this document.

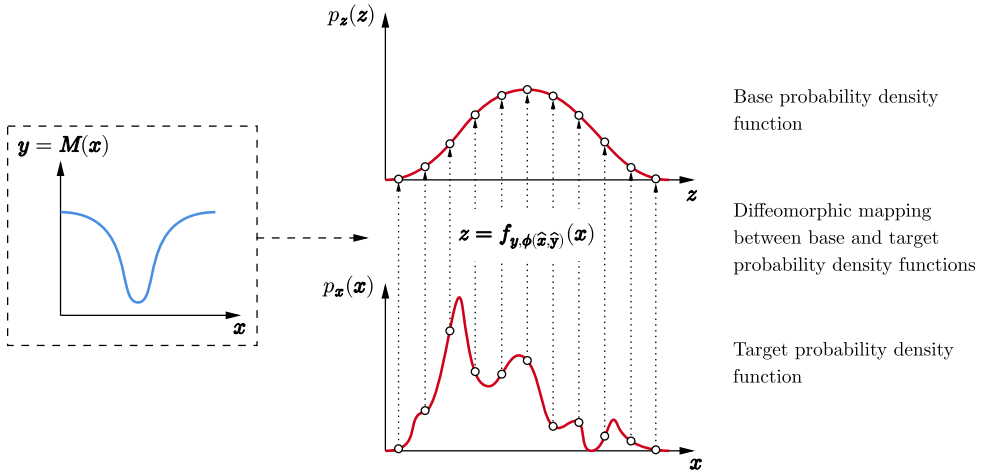


Figure 3.10: Conceptual principle of a conditional normalizing flow $\boldsymbol{z} = \boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{x})$, transforming a target probability density function $p_{\boldsymbol{x}}(\boldsymbol{x})$ to a base probability density function $p_{\boldsymbol{z}}(\boldsymbol{z})$ conditional on $\boldsymbol{y} = \boldsymbol{M}(\boldsymbol{x})$, where $\boldsymbol{M}$ is a model that relates the data $\boldsymbol{x}$ to the conditioning data $\boldsymbol{y}$.

3.7.4 Coupling layer normalizing flows

Consider a diffeomorphism

$$\boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})} : \mathbb{R}^d \to \mathbb{R}^d, \quad \boldsymbol{x} \mapsto \boldsymbol{z} = \boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{x}), \tag{3.19}$$

where $\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})$ are weights depending on training data $\hat{\mathbf{x}} \in \mathbb{R}^{d\times q}$ and $\hat{\mathbf{y}} \in \mathbb{R}^{D\times q}$, which are associated to a deep neural network implementing the diffeomorphic function $\boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}$. The probability density for $\boldsymbol{x}$ as calculated via $\boldsymbol{x} = \boldsymbol{f}^{-1}_{\boldsymbol{y},\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{z})$ therefore becomes conditional on the context $\boldsymbol{y}$, because the network parameters $\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})$ themselves depend on the context $\hat{\mathbf{y}}$ used to train the network. Hence for $p_{\boldsymbol{x}}(\boldsymbol{x})$, the notation $p_{\boldsymbol{x}}(\boldsymbol{x}) = p_{\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{x}|\boldsymbol{y})$ is introduced.

With the change of variable theorem, one obtains

$$\begin{aligned} 1 = \int_{\mathbb{R}^d} p_{\boldsymbol{z}}(\boldsymbol{z})\,\mathrm{d}\boldsymbol{z} &= \int_{\mathbb{R}^d} p_{\boldsymbol{z}}[\boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{x})] \left|\det\left(\frac{\partial \boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{x})}{\partial \boldsymbol{x}}\right)\right| \mathrm{d}\boldsymbol{x} \\ &= \int_{\mathbb{R}^d} p_{\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{x}|\boldsymbol{y})\,\mathrm{d}\boldsymbol{x}, \end{aligned} \tag{3.20}$$

from which it follows, that the probability density $p_{\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{x}|\boldsymbol{y})$ can be expressed by means of the diffeomorphism $\boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}$, the Jacobian $\frac{\partial \boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{x})}{\partial \boldsymbol{x}}$ and the probability density $p_{\boldsymbol{z}}(\boldsymbol{z})$, namely

$$p_{\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{x}|\boldsymbol{y}) = p_{\boldsymbol{z}}[\boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{x})] \left|\det\left(\frac{\partial \boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{x})}{\partial \boldsymbol{x}}\right)\right|. \tag{3.21}$$

This identity is to be considered as the key to implementing a loss function for the normalizing flow technique.

Since a concatenation of diffeomorphisms is again a diffeomorphism, the function $\boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}$ can be composed of many diffeomorphisms. Let $\boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi},(1)}, \ldots, \boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi},n}$ with $n \in \mathbb{N}$ be a number of diffeomorphisms, such that

$$\boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})} = \boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi},(n)} \circ \cdots \circ \boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi},(1)}. \tag{3.22}$$

Introducing the notation $\boldsymbol{x} \doteq \boldsymbol{z}_{(0)}$, $\boldsymbol{z} \doteq \boldsymbol{z}_{(n)}$ and $\boldsymbol{z}_{(k)} \doteq \boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi},(k)}(\boldsymbol{z}_{(k-1)})$ with $k \in \{1, \ldots, n\}$, the function $\boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}$ can be written as

$$\boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{z}_{(0)}) = \boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi},(n)} \circ \cdots \circ \boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi},(1)}(\boldsymbol{z}_{(0)}). \tag{3.23}$$

In fig. 3.11, an overview of the working principle of a coupling layer normalizing flow is visualized. Because $\boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{x})$ is a concatenation

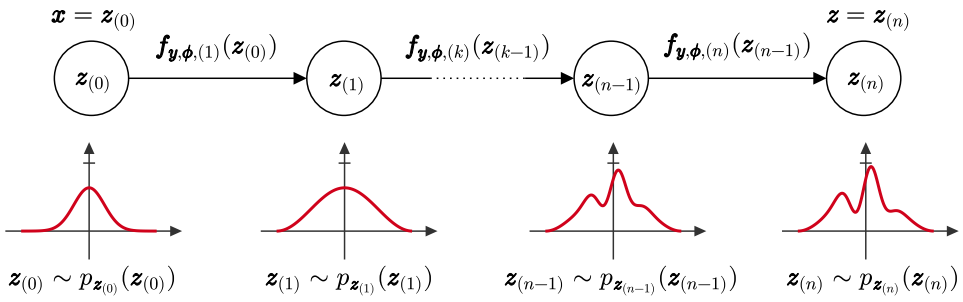


Figure 3.11: Visualization of the working principle of coupling layer normalizing flows.

of functions $\boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi},(k)}(\boldsymbol{z}_{(k-1)})$ with $k \in \{1, \ldots, n\}$, the determinant of the Jacobian for $\boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{x})$ with respect to $\boldsymbol{x}$ is given by

$$\det\left(\frac{\partial \boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{x})}{\partial \boldsymbol{x}}\right) = \prod_{k=1}^{n} \det\left(\frac{\partial \boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi},(k)}(\boldsymbol{z}_{(k-1)})}{\partial \boldsymbol{z}_{(k-1)}}\right), \tag{3.24}$$

where the Jacobians on the right-hand side take the form

$$\frac{\partial \boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi},(k)}(\boldsymbol{z}_{(k-1)})}{\partial \boldsymbol{z}_{(k-1)}} = \begin{pmatrix} \frac{\partial f_{\boldsymbol{y},\boldsymbol{\phi},(k),1}(\boldsymbol{z}_{(k-1)})}{\partial z_{(k-1),1}} & \cdots & \frac{\partial f_{\boldsymbol{y},\boldsymbol{\phi},(k),1}(\boldsymbol{z}_{(k-1)})}{\partial z_{(k-1),d}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_{\boldsymbol{y},\boldsymbol{\phi},(k),d}(\boldsymbol{z}_{(k-1)})}{\partial z_{(k-1),1}} & \cdots & \frac{\partial f_{\boldsymbol{y},\boldsymbol{\phi},(k),d}(\boldsymbol{z}_{(k-1)})}{\partial z_{(k-1),d}} \end{pmatrix} \tag{3.25}$$

with $k \in \{1, \ldots, n\}$. Every function $\boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi},(k)}(\boldsymbol{z}_{(k-1)})$ for $k \in \{1, \ldots, n\}$ represents one of the n so-called coupling layers constituting the total

flow function $\boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}$. Every composite function $\boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi},(k)}$ is implemented by application of neural networks depending on $\boldsymbol{x}$ and $\boldsymbol{y}$, such that each composite function is fully invertible and differentiable. The total set of parameters learned by the neural networks is then identified with $\boldsymbol{\phi}(\hat{\mathbf{x}}, \hat{\mathbf{y}})$, hence defining the function $\boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}$.

3.8 Affine coupling layer normalizing flow

Affine coupling layer normalizing flows[1] are one of the most simple types of normalizing flows. They basically rely on a concatenation of many affine transformations, where the scale and shift parameters are implemented as neural networks. Although quite easy to implement, these type of flows lack flexibility, as noted by [Durkan et al., 2019, p.3]. However, they provide good insight to the working principle of a normalizing flow and are sufficiently flexible for some less complex applications as shown in chapter 4.

3.8.1 Construction of an affine coupling layer normalizing flow

Consider a diffeomorphism

$$\boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})} = \boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi},(n)} \circ \cdots \circ \boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi},(1)} : \mathbb{R}^d \to \mathbb{R}^d, \quad \boldsymbol{x} \mapsto \boldsymbol{z} = \boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{x}), \tag{3.26}$$

where all quantities are given as defined in section 3.7.4.

In the case of $\boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi},(k)}$ being so-called affine coupling layers, the functions can explicitly written by means of two arbitrary functions

$$\boldsymbol{\mu}_{(k)} : \mathbb{R}^{m+D} \to \mathbb{R}^{d-m}, \quad \boldsymbol{\sigma}_{(k)} : \mathbb{R}^{m+D} \to \mathbb{R}^{d-m} \tag{3.27}$$

with $m \in \{1, \ldots, d\}$ and $k \in \{1, \ldots, n\}$ implemented as deep neural networks. That is to say, that the neural networks $\boldsymbol{\mu}_{(k)}$ and $\boldsymbol{\sigma}_{(k)}$ take a vector of dimension $m + D$ as an input and return a vector of dimension $d - m$ as an output; there can be an arbitrary sequence of network layers and activation functions in between input and output. Every mapping $\boldsymbol{z}_{(k)} = \boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi},(k)}(\boldsymbol{z}_{(k-1)})$ for $k \in \{1, \ldots, n\}$ is now defined as an affine

[1] A basic implementation of an affine coupling layer normalizing flow in the `PyTorch` framework, alongside with two example applications, is given by the author at `https://github.com/danielzahnd/master-thesis/tree/main/code/nf-affine-coupling-layer-normalizing-flow`. This code can in principle be applied to any other set of data.

transformation, such that

$$f_{\boldsymbol{y},\boldsymbol{\phi},(k),l}(\boldsymbol{z}_{(k-1)}) = z_{(k-1),l}, \tag{3.28}$$
$$f^{-1}_{\boldsymbol{y},\boldsymbol{\phi},(k-1),l}(\boldsymbol{z}_{(k)}) = z_{(k),l} \tag{3.29}$$

for $l \in \{1, \dots, m\}$ and

$$f_{\boldsymbol{y},\boldsymbol{\phi},(k),l}(\boldsymbol{z}_{(k-1)}) = \mu_{(k),l}(\boldsymbol{z}_{(k-1),1:m}, \boldsymbol{y}) + e^{\sigma_{(k),l}(\boldsymbol{z}_{(k-1),1:m}, \boldsymbol{y})} \cdot z_{(k-1),l}, \tag{3.30}$$
$$f^{-1}_{\boldsymbol{y},\boldsymbol{\phi},(k-1),l}(\boldsymbol{z}_{(k)}) = [z_{(k),l} - \mu_{(k),l}(\boldsymbol{z}_{(k),1:m}, \boldsymbol{y})] \cdot e^{-\sigma_{(k),l}(\boldsymbol{z}_{(k),1:m}, \boldsymbol{y})} \tag{3.31}$$

for $l \in \{m+1, \dots, d\}$, where $\boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi},(k)} = (f_{\boldsymbol{y},\boldsymbol{\phi},(k),1}, \dots, f_{\boldsymbol{y},\boldsymbol{\phi},(k),d})^\top$ and $\boldsymbol{z}_{(k),1:m} = (z_{(k),1}, \dots, z_{(k),m})^\top$.

It is a convenient property of such affine functions $\boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi},(k)}(\boldsymbol{z}_{(k-1)})$ for $k \in \{1, \dots, n\}$, that the Jacobians with respect to $\boldsymbol{z}_{(k-1)}$ are triangular, that is

$$\frac{\partial \boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi},(k)}(\boldsymbol{z}_{(k-1)})}{\partial \boldsymbol{z}_{(k-1)}} = \begin{pmatrix} \mathbb{1}_m & \mathbb{0}_m \\ \frac{\partial \boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi},(k),m+1:d}(\boldsymbol{z}_{(k-1)})}{\partial \boldsymbol{z}_{(k-1),m+1:d}} & \operatorname{diag}\left(e^{\sigma_{(k),m+1:d}(\boldsymbol{z}_{(k-1),1:m}, \boldsymbol{y})}\right) \end{pmatrix}, \tag{3.32}$$

where $\mathbb{1}_m$ is a unity matrix and $\mathbb{0}_m$ is a zero matrix of dimension $m \times m$. The lower left quantity in the above matrix is again a Jacobian and takes the form

$$\frac{\partial \boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi},(k),m+1:d}(\boldsymbol{z}_{(k-1)})}{\partial \boldsymbol{z}_{(k-1),m+1:d}} = \begin{pmatrix} \frac{\partial f_{\boldsymbol{y},\boldsymbol{\phi},(k),m+1}(\boldsymbol{z}_{(k-1)})}{\partial z_{(k-1),m+1}} & \cdots & \frac{\partial f_{\boldsymbol{y},\boldsymbol{\phi},(k),m+1}(\boldsymbol{z}_{(k-1)})}{\partial z_{(k-1),d}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_{\boldsymbol{y},\boldsymbol{\phi},(k),d}(\boldsymbol{z}_{(k-1)})}{\partial z_{(k-1),m+1}} & \cdots & \frac{\partial f_{\boldsymbol{y},\boldsymbol{\phi},(k),d}(\boldsymbol{z}_{(k-1)})}{\partial z_{(k-1),d}} \end{pmatrix}; \tag{3.33}$$

the lower right quantity however is a diagonal matrix with the elements $e^{\sigma_{(k),l}(\boldsymbol{z}_{(k-1),1:m}, \boldsymbol{y})}$ for $l \in \{m+1, d\}$ on the diagonal. Therefore, the Jacobian $\partial \boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi},(k)}(\boldsymbol{z}_{(k-1)}) / \partial \boldsymbol{z}_{(k-1)}$ is triangular and hence its determinant is given by the product of the diagonal elements, that is

$$\det\left(\frac{\partial \boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi},(k)}(\boldsymbol{z}_{(k-1)})}{\partial \boldsymbol{z}_{(k-1)}}\right) = \prod_{j=m+1}^{d} e^{\sigma_{(k),j}(\boldsymbol{z}_{(k-1),1:m}, \boldsymbol{y})}. \tag{3.34}$$

Taking the logarithm of this expression results in a transformation of the product to a sum, such that

$$\log\left[\det\left(\frac{\partial \boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi},(k)}(\boldsymbol{z}_{(k-1)})}{\partial \boldsymbol{z}_{(k-1)}}\right)\right] = \sum_{j=m+1}^{d} \sigma_{(k),j}(\boldsymbol{z}_{(k-1),1:m}, \boldsymbol{y}) \tag{3.35}$$

holds. Recall, that the complete normalizing flow is given as a concatenation of affine coupling functions (layers), such that

$$\det\left(\frac{\partial \boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{x})}{\partial \boldsymbol{x}}\right) = \prod_{k=1}^{n} \det\left(\frac{\partial \boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi},(k)}(\boldsymbol{z}_{(k-1)})}{\partial \boldsymbol{z}_{(k-1)}}\right). \tag{3.36}$$

Taking the logarithm of this expression and inserting the previous result eq. (3.35), this leads to

$$\begin{aligned}\log\left[\det\left(\frac{\partial \boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{x})}{\partial \boldsymbol{x}}\right)\right] &= \sum_{k=1}^{n} \log\left[\det\left(\frac{\partial \boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi},(k)}(\boldsymbol{z}_{(k-1)})}{\partial \boldsymbol{z}_{(k-1)}}\right)\right] \\ &= \sum_{k=1}^{n}\sum_{j=m+1}^{d} \sigma_{(k),j}(\boldsymbol{z}_{(k-1),1:m}, \boldsymbol{y}).\end{aligned} \tag{3.37}$$

3.8.2 Training process for affine coupling layer normalizing flows

Recalling the diffeomorphism $\boldsymbol{z} = \boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{x})$ and the change of variable formula for the multivariate case, the probability density $p_{\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{x}|\boldsymbol{y})$ can be written as found in eq. (3.21). Taking the natural logarithm of expression eq. (3.21) and inserting the result eq. (3.37) into it, one obtains

$$\begin{aligned}\log\left[p_{\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{x}|\boldsymbol{y})\right] &= \log\left(p_{\boldsymbol{z}}[\boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{x})]\right) \\ &\quad + \left|\sum_{k=1}^{n}\sum_{j=m+1}^{d} \sigma_{(k),j}(\boldsymbol{z}_{(k-1),1:m}, \boldsymbol{y})\right|.\end{aligned} \tag{3.38}$$

Note, that the natural logarithm $\log(a)$ is strictly monotonic increasing for $a > 0$; furthermore, $-\infty < \log(a) \leq 0$ for $0 < a \leq 1$. Therefore, maximizing the so-called likelihood $p_{\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{x}|\boldsymbol{y})$ is equivalent to maximizing $\log\left[p_{\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{x}|\boldsymbol{y})\right]$, which is named the log-likelihood. Maximizing $p_{\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{x}|\boldsymbol{y})$ corresponds to finding optimal network parameters $\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})$ for the deep neural networks $\boldsymbol{\mu}_{(k)}$ and $\boldsymbol{\sigma}_{(k)}$ for $k \in \{1,\ldots,n\}$ constituting the flow function $\boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}$, such that the most suitable model parameters $\boldsymbol{x}$ given some context $\boldsymbol{y}$ are found. In the case of a stellar atmosphere inversion, $p_{\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{x}|\boldsymbol{y})$ represents a probability distribution for the solutions $\boldsymbol{x}$ of an inversion $\boldsymbol{x} = \boldsymbol{M}^{-1}(\boldsymbol{y})$, where $\boldsymbol{M}$ is the atmosphere model and $\boldsymbol{y}$ are observations.

That is to say, that the normalizing flow, i.e. the neural networks constituting it, are trained by means of minimizing the negative log-likelihood $-\log\left[p_{\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{x}|\boldsymbol{y})\right]$, which is equivalent to maximizing positive log-likelihood and hence also to maximizing the positive likelihood.

Therefore, the loss function to minimize by any gradient-descent and backpropagation algorithm employed can be defined as

$$\begin{aligned}\mathcal{L}_{\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{x},\boldsymbol{y}) &= -\log\left(p_{\boldsymbol{z}}[\boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{x})]\right) - \log\left(\left|\det\left(\frac{\partial \boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{x})}{\partial \boldsymbol{x}}\right)\right|\right) \\ &= -\log\left(p_{\boldsymbol{z}}[\boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{x})]\right) - \left|\sum_{k=1}^{n}\sum_{j=m+1}^{d}\sigma_{(k),j}(\boldsymbol{z}_{(k-1),1:m},\boldsymbol{y})\right|.\end{aligned} \tag{3.39}$$

This loss function is minimized with respect to the neural network parameters $\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})$.

3.9 Piecewise rational quadratic spline normalizing flow

Affine coupling layer normalizing flows as describe in detail above are simple and relatively easy to implement, but they lack flexibility. As [Durkan et al., 2019, p.3] note, affine coupling layer flows may struggle to model especially multimodal density functions using just affine transformations. Recall, that as a base density, usually a standard normal distribution is taken. If there are now multiple modes present in the target distribution of the data to model, the flow needs to learn how to transform just one mode present in the base distribution to the multiple modes in the multimodal target distribution, thus presenting the flow with the need to learn potentially highly nonlinear transformation. This is where affine coupling layer normalizing flows are at their limit.

Hence, [Durkan et al., 2019, p.4] introduce piecewise rational quadratic spline normalizing flows, which are more flexible in nature, but mathematically still relatively simple and furthermore fully invertible and differentiable, thus diffeomorphic.

This type of normalizing flow is therefore the flow of choice for the more complex applications as presented and discussed in chapter 5. These kind of flows were however not implemented from scratch by the author, but the already existing software [Durkan et al., 2020] was made use of; and the theoretical considerations below follow the material as presented by [Durkan et al., 2019].

3.9.1 Construction of a piecewise rational quadratic spline normalizing flow

Consider a diffeomorphism

$$\boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})} = \boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi},(n)} \circ \cdots \circ \boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi},(1)} : \mathbb{R}^d \to \mathbb{R}^d, \quad \boldsymbol{x} \mapsto \boldsymbol{z} = \boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{x}). \tag{3.40}$$

In the case of this function being implemented by piecewise rational quadratic spline coupling layers, these individual layers can be defined as follows.

Again, $\boldsymbol{x} \sim p_{\boldsymbol{x}}(\boldsymbol{x})$ and $\boldsymbol{y} \sim p_{\boldsymbol{y}}(\boldsymbol{y})$ hold, whereas the modelled probability density of $\boldsymbol{x}$ conditional on $\boldsymbol{y}$ as learned by a normalizing flow is denoted as $\boldsymbol{x} \sim p_{\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{x}|\boldsymbol{y})$, where $\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})$ are weights associated to the neural network(s) used to implement the flow. The quantities $\hat{\mathbf{x}} \in \mathbb{R}^{d\times q}$ and $\hat{\mathbf{y}} \in \mathbb{R}^{D\times q}$ are just training data.

Consider an input vector $\boldsymbol{x} \in \mathbb{R}^d$ and a conditioning vector $\boldsymbol{y} \in \mathbb{R}^D$. Let now be a vector $\boldsymbol{s}$ defined, such that it contains the first m entries of the input vector $\boldsymbol{x}$ and the whole conditioning vector $\boldsymbol{y}$, that is to say, $\boldsymbol{s} \doteq (x_1, \ldots, x_m, y_1, \ldots, y_D) \in \mathbb{R}^{m+D}$. In order to define a single layer $f_{\boldsymbol{y},\boldsymbol{\phi},(k)}$ with $k \in \{1, \ldots, n\}$ of the flow, one requires $(d-m)(3K-1)$ parameters per layer k These parameters are given for each layer k by means of a neural network

$$\boldsymbol{\mu}_{(k)} : \mathbb{R}^{m+D} \to \mathbb{R}^{(d-m)(3K-1)}, \quad \boldsymbol{s}_{(k-1)} \mapsto \boldsymbol{\phi}_{(k)} = \boldsymbol{\mu}_{(k)}(\boldsymbol{s}_{(k-1)}) \tag{3.41}$$

whereby $m \in \{1, \ldots, d\}$, $k \in \{1, \ldots, n\}$ and $K \in \mathbb{N}$. The integer K is the number of spline knots of choice. The precise nature of the mapping $\boldsymbol{\mu}_{(k)}$ is that of an arbitrary neural network, which therefore cannot be

explicitly written, but can be accounted for by

$$\boldsymbol{\phi}_{(k)} = \boldsymbol{\mu}_{(k)}(\boldsymbol{s}_{(k-1)}) = \begin{pmatrix} \phi^w_{(k),m+1,1} \\ \vdots \\ \phi^w_{(k),i,j} \\ \vdots \\ \phi^w_{(k),d,K} \\ \phi^h_{(k),m+1,1} \\ \vdots \\ \phi^h_{(k),i,j} \\ \vdots \\ \phi^h_{(k),d,K} \\ \phi^\delta_{(k),m+1,1} \\ \vdots \\ \phi^\delta_{(k),i,j} \\ \vdots \\ \phi^\delta_{(k),d,K-1} \end{pmatrix} \in \mathbb{R}^{(d-m)(3K-1)} \tag{3.42}$$

in the general case. Let furthermore $\boldsymbol{\phi}_{(k),(i)}$ with $i \in \{m+1, \ldots, d\}$ be defined as

$$\boldsymbol{\phi}_{(k),(i)} \doteq (\phi^w_{(k),i,1}, \ldots, \phi^w_{(k),i,K}, \phi^h_{(k),i,1}, \ldots, \phi^h_{(k),i,K}, \phi^\delta_{(k),i,1}, \ldots, \phi^\delta_{(k),i,K-1})^\top. \tag{3.43}$$

The letter w denotes the with of bins, h the height of bins and δ the derivatives at the internal points.

The last ingredients to be finally able to define the individual transformations $\boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi},(k)}$ are $d-m$ actual quadratic functions

$$g_{\boldsymbol{\phi}_{(k),(i)}} : \mathbb{R} \to \mathbb{R} \tag{3.44}$$

for each coupling layer k. Now, each coupling layer k is defined as

$$\boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi},(k)} : \mathbb{R}^d \to \mathbb{R}^d, \quad \boldsymbol{z}_{(k-1)} \mapsto \boldsymbol{z}_{(k)} = \begin{pmatrix} z_{(k-1),1} \\ \vdots \\ z_{(k-1),m} \\ g_{\boldsymbol{\phi}_{(k),(m+1)}}(z_{(k-1),m+1}) \\ \vdots \\ g_{\boldsymbol{\phi}_{(k),(d)}}(z_{(k-1),d}) \end{pmatrix}, \tag{3.45}$$

where $\boldsymbol{x} = \boldsymbol{z}_{(0)}$ and $\boldsymbol{z} = \boldsymbol{z}_{(n)}$. Thus, the total diffeomorphism $\boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}$ is just given by the concatenation of all $\boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi},(k)}$ for $k \in \{1, \ldots, n\}$, namely

$$\boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})} = \boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi},(n)} \circ \cdots \circ \boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi},(1)}, \tag{3.46}$$

whereby the piecewise rational quadratic spline normalizing flow is defined.

3.9.2 Training process for piecewise rational quadratic spline normalizing flows

As the precice tranformation rules for the piecewise rational quadratic spline flow are quite complex, the exact formula for the loss function allowing for a direct implementation is not given here. However, a few remarks shall be made with reference to training such a normalizing flow.

Since all involved functions used to define the piecewise rational quadratic spline flow are fully differentiable and invertible, one can just use eq. (3.21) as a means to implement the loss function $\mathcal{L}_{\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{x}, \boldsymbol{y})$. What one has to do to this end is to take negative logarithm of the probability density $p_{\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})(\boldsymbol{x}|\boldsymbol{y})}$ as the loss function, i.e.

$$\mathcal{L}_{\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{x}, \boldsymbol{y}) = -\log\left[p_{\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{x}|\boldsymbol{y})\right]. \tag{3.47}$$

Minimizing the loss function now means to minimize $\log\left[1/p_{\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{x}|\boldsymbol{y})\right]$ and therefore to maximize the likelihood $p_{\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{x}|\boldsymbol{y})$. This is to say, that one finds the most likely input $\boldsymbol{x}$ pertaining to a certain context $\boldsymbol{y}$, if the loss function is truly globally minimized. The logarithm comes into play, because multiplications then turn into additions, which are much easier to implement than multiplications. However, the exact calculation the determinant $\det\left(\partial \boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{x})/\partial \boldsymbol{x}\right)$ is tedious and not insightful, therefore it is not given here.

Chapter 4

Proof of concept: Affine coupling layer normalizing flows

This section provides several experiments and applications of an affine coupling layer normalizing flow as described in detail in section 3.8. Hereby, the implementation[1] was carried out from scratch in `Python` following section 3.8, using the `PyTorch` framework.

4.1 Moons

4.1.1 Conceptual formulation

In this experiment, a normalizing flow using affine coupling layers was tested on data $\boldsymbol{x} = (x_1, x_2)^\top \in \mathbb{R}^2$ given by a probability density function $p_{\boldsymbol{x}}(\boldsymbol{x})$ representing two moons[2]. The goal was to map the probability density $p_{\boldsymbol{x}}(\boldsymbol{x})$ representing the moons distribution to a standard normal distribution $p_{\boldsymbol{z}}(\boldsymbol{z})$, i.e. to find a diffeomorphism

$$\boldsymbol{f}_{\boldsymbol{\phi}(\hat{\mathbf{x}})} : \mathbb{R}^2 \to \mathbb{R}^2, \quad p_{\boldsymbol{x}}(\boldsymbol{x}) \sim \boldsymbol{x} \mapsto \boldsymbol{z} = \boldsymbol{f}_{\boldsymbol{\phi}(\hat{\mathbf{x}})}(\boldsymbol{x}) \sim \mathcal{N}(0,1)^2. \tag{4.1}$$

The moons distribution is a distribution of points in a plane, which creates two interlacing moon-like shapes. This distribution is transformed via the found diffeomorphism $\boldsymbol{f}_{\boldsymbol{\phi}(\hat{\mathbf{x}})}$ to a standard normal distribution in two dimensions; in fig. 4.1, this mapping can be found visualized.

[1] See link in footnote 1 to access the general source code for the experiments presented in this chapter.

[2] See fig. 4.1 for a visualization of this dataset and a normalizing flow applied to it.

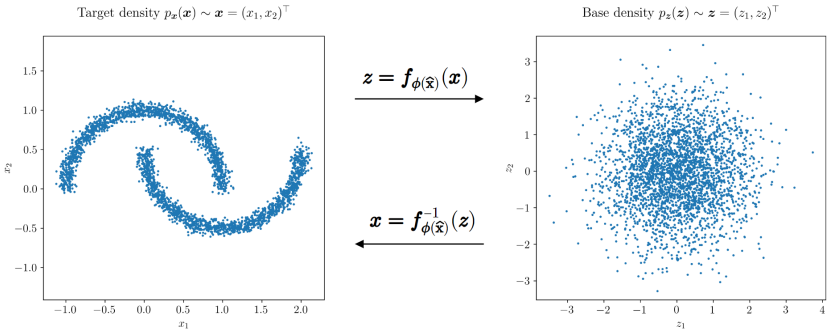


Figure 4.1: Visualization of a normalizing flow mapping the moons distribution to a standard normal distribution and vice versa in a plane.

In order to establish such a mapping, a normalizing flow to implement the function $\boldsymbol{f}_{\phi(\hat{\mathbf{x}})}$ was created as described in section 3.8. Herewith, $\hat{\mathbf{x}} \in \mathbb{R}^{d \times q}$ denotes the training data, where $d = 2$ and $q \in \mathbb{N}$. It is to be remembered, that the probability density $p_{\boldsymbol{x}}(\boldsymbol{x})$ can be written as being conditional on the network parameters $\boldsymbol{\phi}(\hat{\mathbf{x}})$. Hence, the notation $p_{\boldsymbol{x}}(\boldsymbol{x}) = p_{\boldsymbol{\phi}(\hat{\mathbf{x}})}(\boldsymbol{x})$ is adopted to indicate, that for a trained normalizing flow, $\boldsymbol{x}$ can be written as $\boldsymbol{x} = \boldsymbol{f}^{-1}_{\boldsymbol{\phi}(\hat{\mathbf{x}})}(\boldsymbol{z})$ and that the density $p_{\boldsymbol{x}}(\boldsymbol{x})$ as sampled by application of the inverse normalizing flow to normally distributed samples $\boldsymbol{z}$ is therefore dependent upon the network parameters $\boldsymbol{\phi}(\hat{\mathbf{x}})$.

4.1.2 Results and discussion

The subsequent results and figures were obtained[3] using a training sample size of 80000 a testing sample size of 20000, a batch size of 512, a hidden layer size of 512, a coupling layer amount of 10, a learning rate of 0.001, a scheduling rate of 0.999 and 30 epochs to train the flow. This hyperparameter setting was found by manual inspection and improvement of resulting learning curves. The training process is visualized by fig. 4.2.

The performance after training can be seen as shown in fig. 4.3. It is evident, that the normalizing flow clearly has learned most of the defining properties of the probability distribution for $\boldsymbol{z}$. While the presented results are far from perfect, it can safely be assumed that better results would be obtained if the optimal set of hyperparameters would be found and applied to the problem. Perfect hereby means, that the flow

[3]The code which produced the presented results is available at `https://github.com/danielzahnd/master-thesis/tree/main/code/nf-moons-example`.

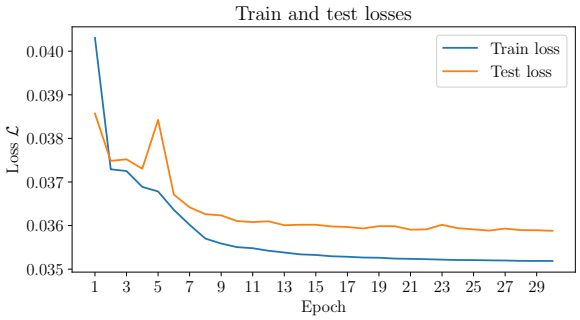


Figure 4.2: Learning curves for the model implementing a map from the moons distribution to a two-dimensional standard normal distribution.

$\boldsymbol{f}_{\phi(\hat{\mathbf{x}})}$ maps samples $\boldsymbol{x}$ from the distribution $p_{\boldsymbol{x}}(\boldsymbol{x})$ to samples $\boldsymbol{z}$ from the standard normal distribution $p_{\boldsymbol{z}}(\boldsymbol{z})$ without any mismaps. Analyzing

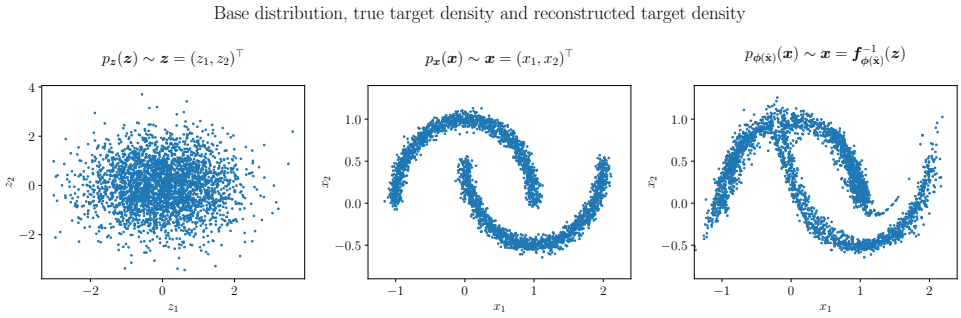


Figure 4.3: Results for a normalizing flow implementing a map from the moons distribution to a standard normal distribution in a plane.

the learning curve fig. 4.2 leads to the conclusion, that there might be some overfitting in the model; hence, reducing the hidden layer size and potentially also the coupling layer amount could result in better model performance. The test loss curve also features a significant jump at epoch 5, which could be an indication of underrepresentativeness of the training or testing dataset. Therefore, further investigation of this example should take into account an optimization of the hyperparameters, aswell as of an optimal training and testing datasize and ratio.

4.2 Linear regression

4.2.1 Conceptual formulation

In this experiment, a normalizing flow using affine coupling layers was tested on data $\boldsymbol{x} = (x_1, x_2)^\top = (a, b)^\top \in \mathbb{R}^2$ and associated context $\boldsymbol{y} = (y_1, \ldots, y_{10})^\top \in \mathbb{R}^{10}$ given by probability density functions $p_{\boldsymbol{x}}(\boldsymbol{x})$ and $p_{\boldsymbol{y}}(\boldsymbol{y})$. Hereby, the context $\boldsymbol{y}$ is generated based on $\boldsymbol{x}$ by means of a model

$$\boldsymbol{M} : \mathbb{R}^2 \to \mathbb{R}^{10}, \quad \boldsymbol{x} \mapsto \boldsymbol{y} = \boldsymbol{M}(\boldsymbol{x}), \tag{4.2}$$

where the model $\boldsymbol{M}$ is defined as

$$y_j = M_j(\boldsymbol{x}) = x_1 u_j + x_2 = a u_j + b_j, \quad j \in \{1, \ldots, 10\} \tag{4.3}$$

with $\boldsymbol{u} = (u_1, \ldots, u_{10})^\top \in \mathbb{R}^{10}$ being a vector containing 10 equally spaced numbers between -5 and 5. A sample of context elements $\boldsymbol{y}_i$ with $i \in \{1, \ldots, L\}$, $L \in \mathbb{N}$ from the training dataset is shown in fig. 4.4.

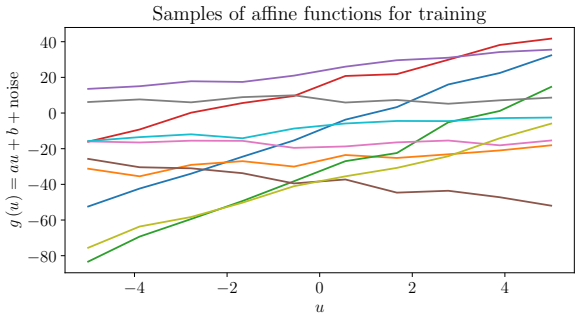


Figure 4.4: Samples from the training dataset of context used in the linear regression example.

The goal was to map the probability density $p_{\boldsymbol{x}}(\boldsymbol{x})$ representing the distributions of the slopes and intercepts of an affine function to a standard normal distribution $p_{\boldsymbol{z}}(\boldsymbol{z})$, i.e. to find a diffeomorphism

$$\boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})} : \mathbb{R}^2 \to \mathbb{R}^2, \quad p_{\boldsymbol{x}}(\boldsymbol{x}) \sim \boldsymbol{x} \mapsto \boldsymbol{z} = \boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{x}) \sim \mathcal{N}(0,1)^2. \tag{4.4}$$

To this end, a normalizing flow with context was created as described above and in section 3.8, in order to implement the function $\boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}$. Herewith, $\hat{\mathbf{x}} \in \mathbb{R}^{d \times q}$ and $\hat{\mathbf{y}} \in \mathbb{R}^{D \times q}$ denote the training data, where $d = 2$, $D = 10$ and $q \in \mathbb{N}$. It is to be remembered, that the probability

density $p_{\boldsymbol{x}}(\boldsymbol{x})$ can be written as being conditional on $\boldsymbol{y}$, since the two are intertwined by $\boldsymbol{x} = \boldsymbol{M}^{-1}(\boldsymbol{y})$. Furthermore and because this conditional probability is to be learned by the normalizing flow $\boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}$, the notation $p_{\boldsymbol{x}}(\boldsymbol{x}) = p_{\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{x}|\boldsymbol{y})$ is adopted to indicate, that for a trained normalizing flow, $\boldsymbol{x}$ can be written as $\boldsymbol{x} = \boldsymbol{f}^{-1}_{\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{z})$ and that the density $p_{\boldsymbol{x}}(\boldsymbol{x})$ as sampled by application of the inverse normalizing flow to normally distributed samples $\boldsymbol{z}$ is therefore dependent upon the network parameters $\boldsymbol{\phi}(\hat{\mathbf{x}}, \hat{\mathbf{y}})$.

4.2.2 Results and discussion

The following results and figures were obtained[4] using a training sample size of 200000, a testing sample size of 50000 a batch size of 512, a hidden layer size of 32, a coupling layer amount of 7, a learning rate of 0.001, a scheduling rate of 0.999 and 10 epochs to train the flow. This hyperparameter setting was found by manual inspection and improvement of resulting learning curves. In fig. 4.5, the training process of the normaliz-

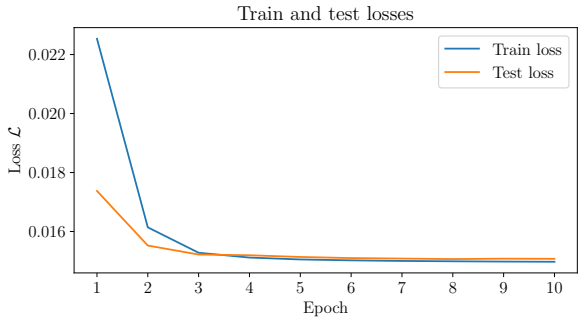


Figure 4.5: Learning curves for the model implementing a map from parameter distributions of an affine function to a two-dimensional standard normal distribution, given the affine function values associated with the function parameters as context.

ing flows is visualized in terms of the training loss $\mathcal{L}$. The train and test loss seem to converge to a nearly constant value after about 6 epochs, thereby exercising little difference to one another. As far as one can tell from inspection of these curves and comparing them with the different cases of non-ideal model performance shown in fig. 3.4, the model seems to have nearly optimal hyperparameters.

[4]The code which produced the presented results is available at `https://github.com/danielzahnd/master-thesis/tree/main/code/nf-linear-regression-example`.

In fig. 4.7, the latent densities for the slope and intercept parameters can be seen; these density plots show what the training samples look like, if they are inserted into the normalizing flow function; that is to say, if $\hat{\boldsymbol{x}}_j$ and $\hat{\boldsymbol{y}}_j$ with $j \in \{1, \ldots, L\}$, $L \in \mathbb{N}$ denote training samples, then one obtains samples $\bar{\boldsymbol{z}}_j$ by means of calculating

$$\bar{\boldsymbol{z}}_j = \boldsymbol{f}_{\hat{\boldsymbol{y}}, \boldsymbol{\phi}(\hat{\mathbf{x}}, \hat{\mathbf{y}})}(\hat{\boldsymbol{x}}_j), \tag{4.5}$$

which should be standard normally distributed. As is readily seen, the

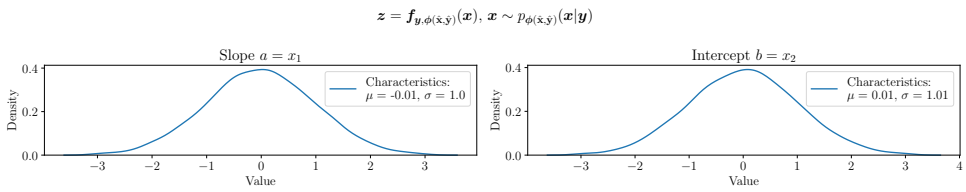


Figure 4.6: Calculated latent densities for the parameter densities in the training dataset.

mean μ and standard deviations σ for both the slope and intercept parameters a and b are close to 0 and 1, as they should be.

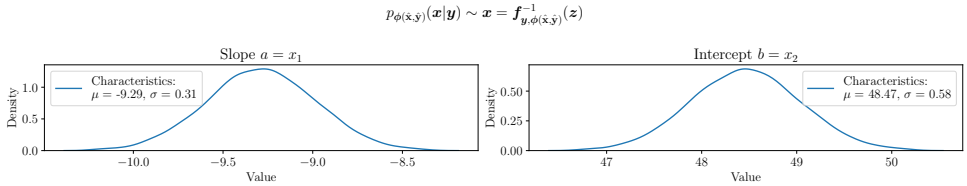


Figure 4.7: Calculated densities for a test observation $\boldsymbol{y} = \boldsymbol{y}_{test}$ generated by parameters $\boldsymbol{x}_{test} = (-9.34, 47.64)^\top$ given to the normalizing flow.

Moreover, in fig. 4.6, the calculated densities for the slope and intercept parameters are depicted for a particular test observation $\boldsymbol{y} = \boldsymbol{y}_{test}$ given to the normalizing flow as context. In particular, $\boldsymbol{y}_{test}$ consists of 10 values calculated by means of the above defined linear model eq. (4.2) based on test parameters $\boldsymbol{x}_{test} = (a_{test}, b_{test})^\top = (-9.34,\ 47.64)^\top$, plus additional noise. This choice of $\boldsymbol{x}_{test}$ is arbitrary, with the only constraint that the chosen parameter values must be within the range of parameters the normalizing flow was trained on. As it can be seen from fig. 4.6, the trained normalizing flow yields a probability density for the parameters $\boldsymbol{x}$ based on the test observation $\boldsymbol{y}_{test}$. Comparing the means μ and standard deviations σ for both probability densities to the true

values $\boldsymbol{x}_{test}$, one can see that the true values lie well within the three sigma ranges of the normalizing flow predictions.

The visualization given in the left panel of fig. 4.8 provides a more instructive view of the distributions aswell as of correlations for both the slope and intercept parameters, given the context vector $\boldsymbol{y}_{test}$. The

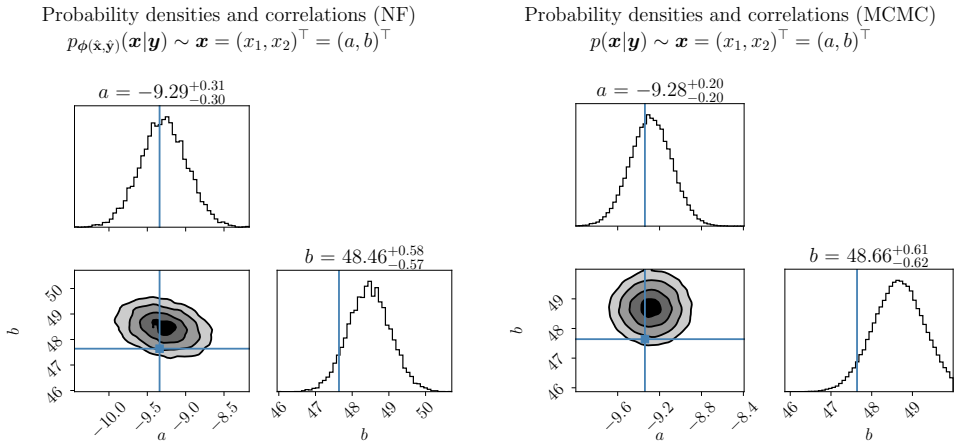


Figure 4.8: Left panel: Corner plot for the calculated densities of slope a and intercept parameter b using the `corner` package, given a test observation $\boldsymbol{y} = \boldsymbol{y}_{test}$ generated by test parameters $\boldsymbol{x}_{test}$. Right panel: Markov Chain Monte Carlo simulation of the same task as the normalizing flow does, applied to the same test observation $\boldsymbol{y} = \boldsymbol{y}_{test}$.

corner plots show nicely, that it indeed seems that the normalizing flow was successfully trained to perform linear regressions, as the blue cross indicates the true test parameters $\boldsymbol{x}_{test}$. First of all, the created Gaussian noise on observations used to train the normalizing flow is reflected in the Gaussian-shaped densities for the slope and intercept parameters. Secondly, the mean values and the associated standard deviations show, that the true values for a and b indeed lie well within the range covered by three standard deviations around the mean of the considered parameter a or b. Additionally, results of a Markov Chain Monte Carlo simulation[5] performing the same task as the normalizing flow are given in the right panel of fig. 4.8. The normalizing flow and the MCMC simulation results are very similar, as one would expect them to be; fig. 4.9 shows both results in one plot, which allows for direct comparison.

Now, the performance of an affine coupling layer normalizing flow for the task of linear regression analyzed in this section can also be assessed

[5]See section 7.1 for an explanation of how this was done.

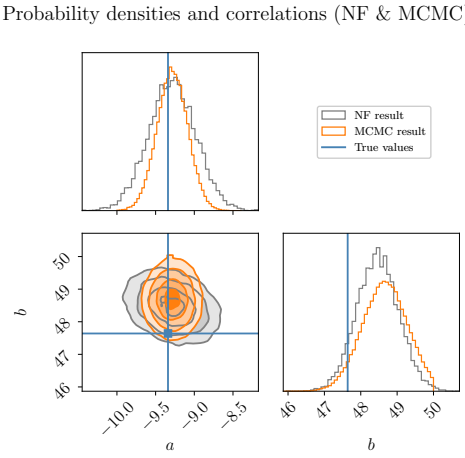


Figure 4.9: Comparison of normalizing flow (NF) and Markov Chain Monte Carlo simulation (MCMC) results for a test observation $\boldsymbol{y} = \boldsymbol{y}_{test}$ generated by test parameters $\boldsymbol{x} = \boldsymbol{x}_{test}$.

by means of making a prediction by the results of the trained flow. To this end, the obtained probability density $\boldsymbol{x} \sim p_{\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{x}|\boldsymbol{y}_{test})$ is used to make predictions for an affine regression line given the test observation $\boldsymbol{y}_{test}$, leading to the results shown in fig. 4.10. The red line denotes the input to the normalizing flow; in this case $\boldsymbol{y}_{test}$. The true curve $g(u)$ (line) is plotted in green and is given by the test parameters $\boldsymbol{x}_{test} = (a_{test}, b_{test})^\top =)(-9.34,\ 47.64)^\top$ as $g(u) = a_{test}u + b_{test}$. In order to obtain the prediction line drawn in blue, the maxima for each dimension of the obtained probability density $p_{\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{x}|\boldsymbol{y}_{test})$ were taken as the most likely parameters given the test observation $\boldsymbol{y}_{test}$. In this way, also an affine function can be drawn. Finally, the uncertainty area plotted in grey is obtained by evaluating an affine function for all parameters $\boldsymbol{x} \sim p_{\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{x}|\boldsymbol{y}_{test})$ and taking the minimum and maximum of the resulting function value at every evaluation point as the lower and upper boundary of possible predictions. These plots lead to the conclusion, that the normalizing flow seems to have successfully learned how to do linear regression on a given set of points $\boldsymbol{y}_{test}$ in a plane; this can be established by the following three arguments:

(1) The prediction by the normalizing flow almost everywhere partially to fully overlaps with the true curve.

(2) The prediction and uncertainty regions as obtained by the normalizing flow agree with those obtained by the Markov Chain Monte Carlo simulation.

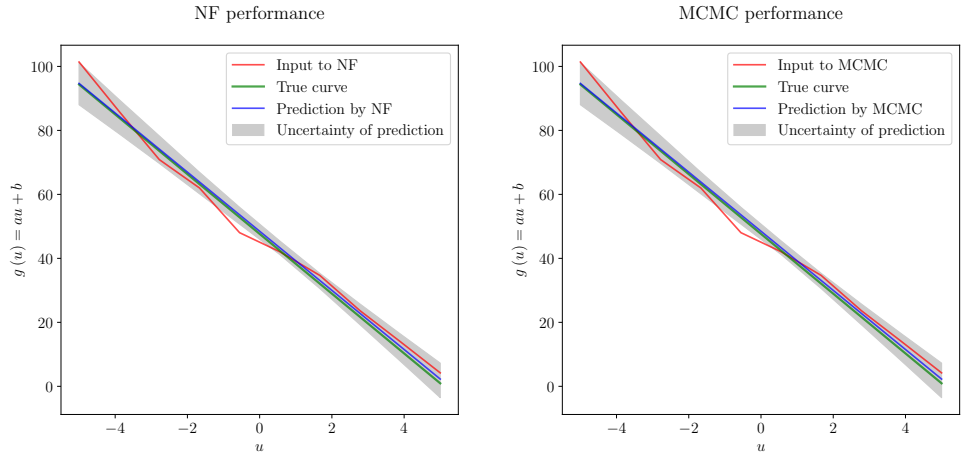


Figure 4.10: Left panel: Results obtained by the normalizing flow (NF) for $\boldsymbol{y} = \boldsymbol{y}_{test}$ and $\boldsymbol{x} = \boldsymbol{x}_{test}$. Right panel: Results obtained by the Markov Chain Monte Carlo simulation (MCMC) for the same test observation $\boldsymbol{y}_{test}$.

(3) The uncertainty region and the prediction itself seems to fit well with the input, on which the linear regression was carried out.

4.3 Feature extraction

4.3.1 Conceptual formulation

In these experiments, a normalizing flow using affine coupling layers was tested on data $\boldsymbol{x} = (x_1, \ldots, x_{10})^\top \in \mathbb{R}^{10}$ and associated context $\boldsymbol{y} = (y_1, \ldots, y_{240})^\top \in \mathbb{R}^{240}$ given by probability density functions $p_{\boldsymbol{x}}(\boldsymbol{x})$ and $p_{\boldsymbol{y}}(\boldsymbol{y})$. Hereby, the context $\boldsymbol{y}$ are spectra of the Sun taken by IRIS satellite. From these spectra, altogether 10 features for each spectrum can be extracted by means of a program implementing a function

$$\boldsymbol{M} : \mathbb{R}^{240} \to \mathbb{R}^{10}, \quad \boldsymbol{y} \mapsto \boldsymbol{x} = \boldsymbol{M}(\boldsymbol{y}). \tag{4.6}$$

The goal was to map the probability density $p_{\boldsymbol{x}}(\boldsymbol{x})$ representing the distributions of the extracted features to a standard normal distribution $p_{\boldsymbol{z}}(\boldsymbol{z})$, i.e. to find a diffeomorphism

$$\boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})} : \mathbb{R}^{10} \to \mathbb{R}^{10}, \quad p_{\boldsymbol{x}}(\boldsymbol{x}) \sim \boldsymbol{x} \mapsto \boldsymbol{z} = \boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{x}) \sim \mathcal{N}(0,1)^{10}. \tag{4.7}$$

To this end, a normalizing flow with context was created as described above and in section 3.8, in order to implement the function $\boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}$.

Here, $\hat{\mathbf{x}} \in \mathbb{R}^{d \times q}$ and $\hat{\mathbf{y}} \in \mathbb{R}^{D \times q}$ denote the training data, where $d = 10$, $D = 240$ and $q \in \mathbb{N}$. It is to be remembered, that the probability density $p_{\boldsymbol{x}}(\boldsymbol{x})$ can be written as being conditional on $\boldsymbol{y}$, since the two are intertwined by $\boldsymbol{x} = \boldsymbol{M}(\boldsymbol{y})$. Furthermore and because this conditional probability is to be learned by the normalizing flow $\boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}$, the notation $p_{\boldsymbol{x}}(\boldsymbol{x}) = p_{\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{x}|\boldsymbol{y})$ is adopted to indicate, that for a trained normalizing flow, $\boldsymbol{x}$ can be written as $\boldsymbol{x} = \boldsymbol{f}^{-1}_{\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{z})$ and that the density $p_{\boldsymbol{x}}(\boldsymbol{x})$ as sampled by application of the inverse normalizing flow to normally distributed samples $\boldsymbol{z}$ is therefore dependent upon the network parameters $\boldsymbol{\phi}(\hat{\mathbf{x}}, \hat{\mathbf{y}})$.

Given the similarity of the feature extraction task to performing inversions on a stellar atmosphere, a successful outcome of corresponding experiments is to be judged as a powerful proof of concept for the application of normalizing flows to inversions of the Sun's atmosphere.

4.3.2 Results and discussion: Test 1

As a first test for the normalizing flow implemented as described above, the flow was trained on all available data; that is to say, that a train and test split of the data at hand was created, whereby the flow was subsequently trained on the train split of the data. The trained flow was then tested by means of propagating a spectrum $\boldsymbol{y}_{test}$ from the test split through the trained normalizing flow. It was then tested, if the predicted feature distributions for the given test spectrum matched the exact feature values or not.

The following results and figures were obtained[6] using a training sample size of 39952, a testing sample size of 9988 a batch size of 512, a hidden layer size of 64, a coupling layer amount of 6, a learning rate of 0.001, a scheduling rate of 0.999, 60 epochs to train the flow and non-randomized splitting of the data into train and test splits. This hyperparameter setting was found by manual inspection and improvement of resulting learning curves.

In fig. 4.11, the training process of the normalizing flows is visualized in terms of the train and test losses. As one can see at first sight, there are many jumps in both the train and test loss curves. Furthermore, after about 40 epochs, the train loss can be observed to be continuously decreasing, while the test loss remains largely constant. Recalling the

[6]The code which produced the presented results is available at `https://github.com/danielzahnd/master-thesis/tree/main/code/nf-feature-extraction-example-1`.

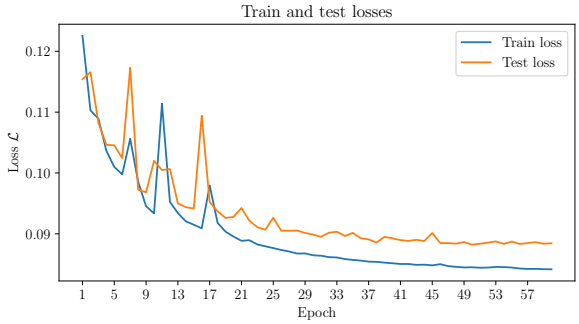


Figure 4.11: Learning curves for experiment 1 on the feature extraction task.

discussion in section 3.3.4 about learning curves (loss curves), the suspicion of overfitting of the model and/or underrepresentativeness of the training data is substantiated. The oscillatory behaviour of the train and test losses could also indicate, that the learning rate was set to too large a rate. The continuous decrease of the train loss, while the test loss remains constant or even increases, is furthermore to be considered as a sign of overfitting; that is, the model likely has learned too many parameters to generalize well to unseen data. Therefore, the used hyperparameter setting is not likely to be considered optimal for the task at hand. The possible underrepresentativeness of the training dataset might be explained by the non-randomness of the train-test splitting procedure. If a randomized train-test splitting were to be carried out prior to training, the jumps in the learning curves might vanish. For a further test using the model at hand, one should therefore focus on fine-tuning the hyperparameters of the model and finding a representative dataset for training and testing.

Moreover, in fig. 4.12, the calculated densities and correlations of feature parameters are depicted for a particular test observation $\boldsymbol{y}_{test}$ given to the normalizing flow as context. In particular, $\boldsymbol{y}_{test}$ belongs to the test set of the data only used for testing; which is to say, that the normalizing flow has no knowledge of the test observation. The blue lines in the mentioned plot indicate the true values $\boldsymbol{x}_{test}$ obtained as $\boldsymbol{x}_{test} = \boldsymbol{M}(\boldsymbol{y}_{test})$. As it is evident from said figure, the normalizing flow predicts feature distributions agreeing with the true values for the features very well for most of the features. There only seem to be some inaccuracies with the line width and triplet emission features, but optimizing the hyperparameters for the training process of the normalizing flow and potentially increasing the amount of train and test data could lead to an improvement of model performance.

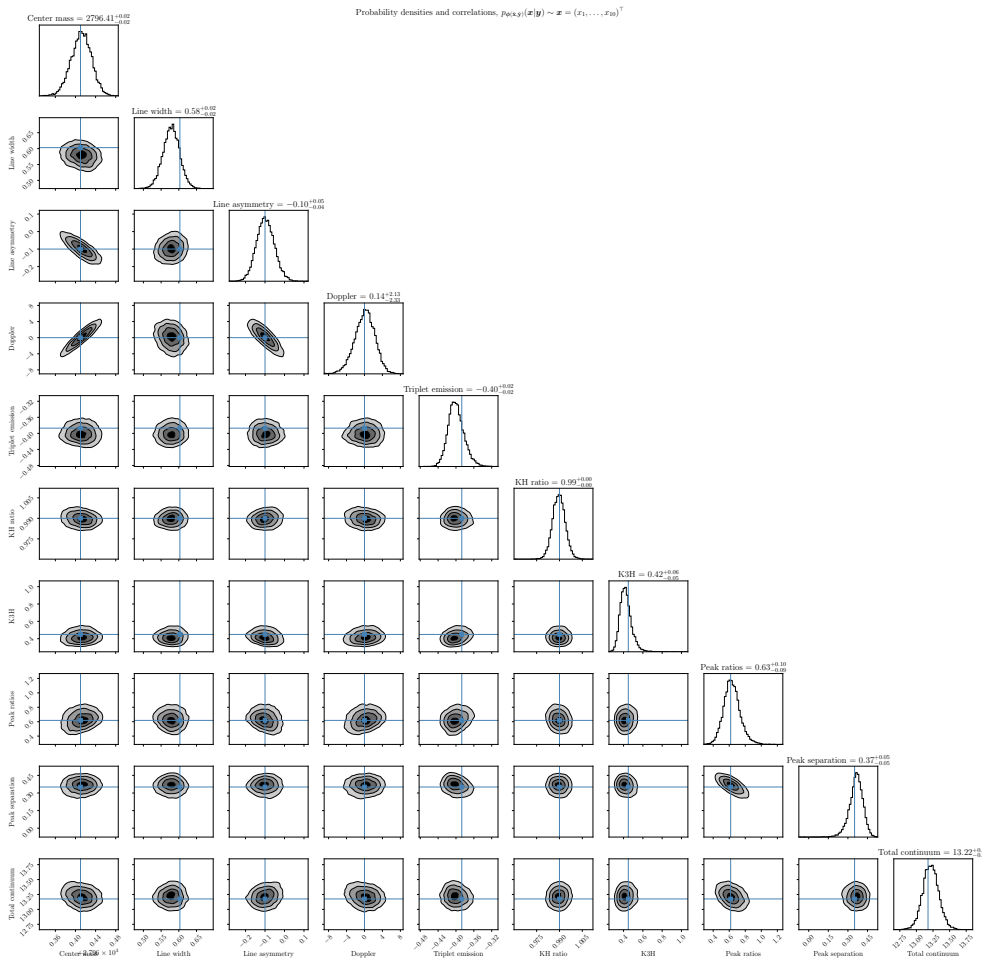


Figure 4.12: Corner plot for the calculated densities of feature parameters using the `corner` package.

4.3.3 Results and discussion: Test 2

As a second test for the normalizing flow implemented as described above, the flow was trained on an artificially created dataset of spectra and corresponding features. This dataset was created by means of first calculating an average spectrum based on the dataset used in the first test mentioned in section 4.3.2. To this average spectrum, random Gaussian noise was added, thus creating a dataset of Gaussian distributed spectra centered around the average spectrum. This was the dataset used to train the normalizing flow on; see fig. 4.13 for a visualization. After training, the normalizing flow was tested on the average spectrum; this is to say, that the average spectrum was propagated through the nor-

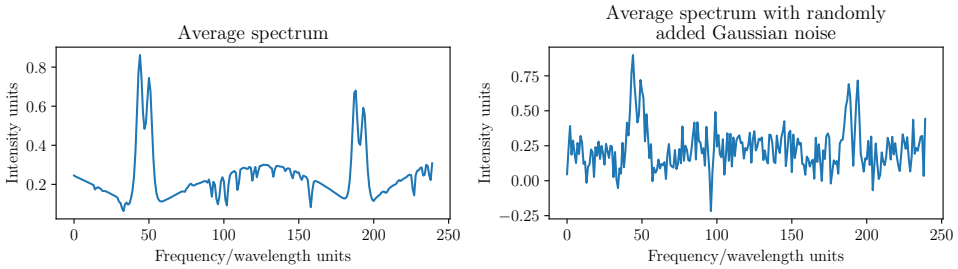


Figure 4.13: Average spectrum of dataset and average spectrum with added random Gaussian noise.

malizing flow, which allows for checking, whether the predicted feature distribution for the average spectrum is indeed Gaussian or not. Based on the Gaussian distributed spectra used to train the flow, a Gaussian distribution of the feature densities as predicted by the normalizing flow should be expected.

The following results and figures were obtained[7] using a training sample size of 44671, a testing sample size of 10000, a batch size of 512, a hidden layer size of 64, a coupling layer amount of 6, a learning rate of 0.001, a scheduling rate of 0.999, 60 epochs to train the flow and non-randomized splitting of the data into train and test splits. This hyperparameter setting was found by manual inspection and improvement of resulting learning curves.

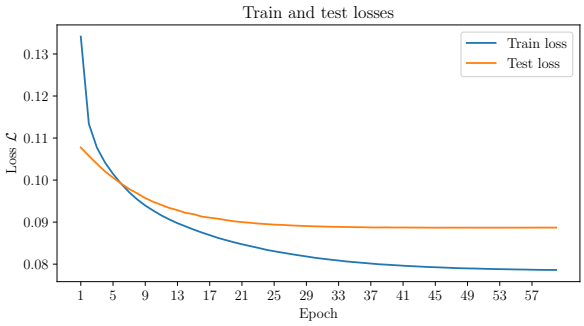


Figure 4.14: Learning curves for experiment 2 on the feature extraction task.

In fig. 4.14, the training process of the normalizing flows is visualized in terms of the train and test losses. Hereby, again the suspicion of

[7]The code which produced the presented results is available at `https://github.com/danielzahnd/master-thesis/tree/main/code/nf-feature-extraction-example-2`.

overfitting arises, because the test loss converges to a nearly constant value, while the train loss continuously decreases. Therefore, it is highly plausible, that the model has learned too many parameters and that hence the hidden layer size and possibly also the coupling layer amount should be decreased in search of optimal hyperparameters. Generally speaking, a so-called grid search for finding optimal hyperparameters could be carried out in future research concerning this task.

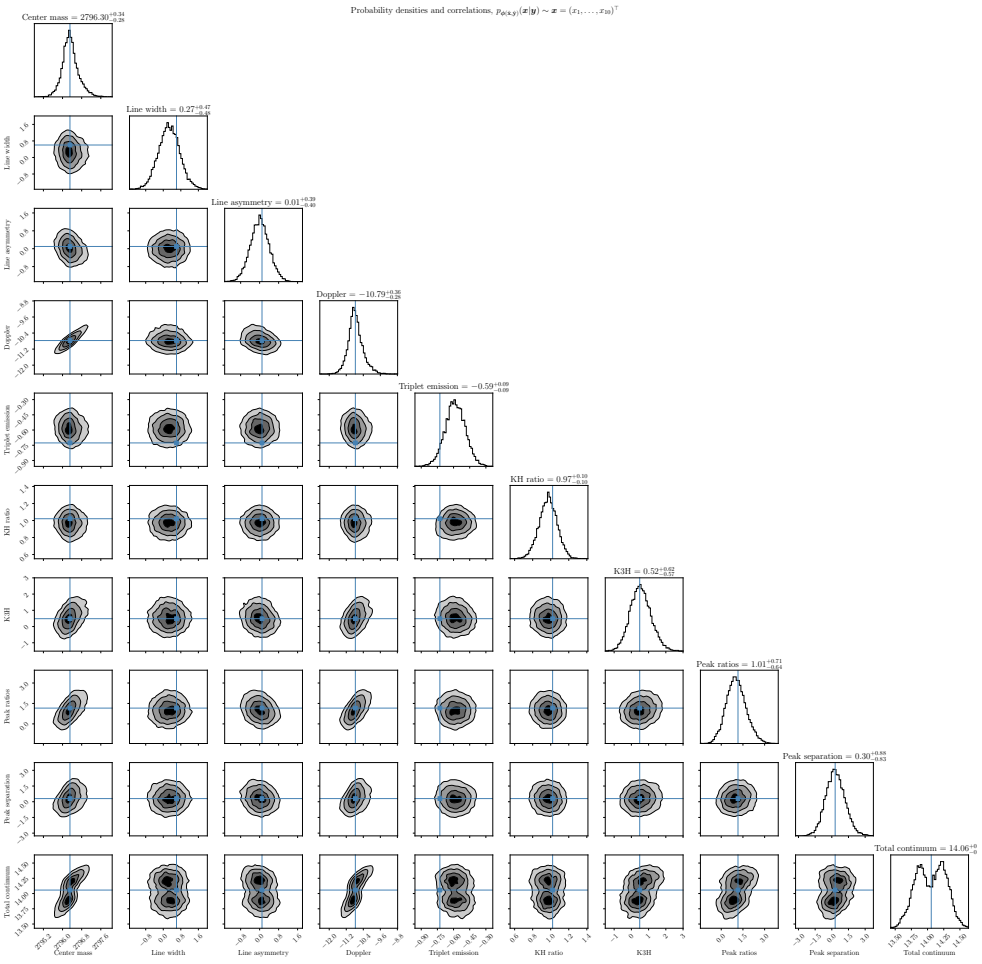


Figure 4.15: Corner plot for the calculated densities of feature parameters using the `corner` package.

Moreover, in fig. 4.15, the calculated densities and correlations of feature parameters are depicted for a particular test observation $\boldsymbol{y}_{test}$ given to the normalizing flow as context; in this case, $\boldsymbol{y}_{test}$ is identical to the average spectrum. The blue lines in the mentioned plot indicate the

true values. As it is evident from said figure, the normalizing flow predicts feature distributions agreeing with the true values for the features quite well for most of the features. There only seem to be some inaccuracies with the line width and triplet emission features, but hyperparameter tuning is likely to improve results even for these two mentioned features. As for the total continuum feature, there seem to be two peaks in the density distribution, the true values for the features corresponding to the average spectrum situated nicely between those peaks, as it would be expected given the generation procedure of the training dataset, provided that the model $\boldsymbol{M}$ is sufficiently linear in nature.

4.3.4 Results and discussion: Test 3

As a third test for the normalizing flow implemented as described above, the flow was trained on an a dataset containing spectra and corresponding features of two very different categories with respect to a `KMeans` clustering. This dataset was created by means of first clustering the available data into several clusters (categories) by a clustering algorithm[8]; fig. 4.16 shows the centroid spectra of both these categories to get an idea of how the spectral of both categories differ from one another. After that, the

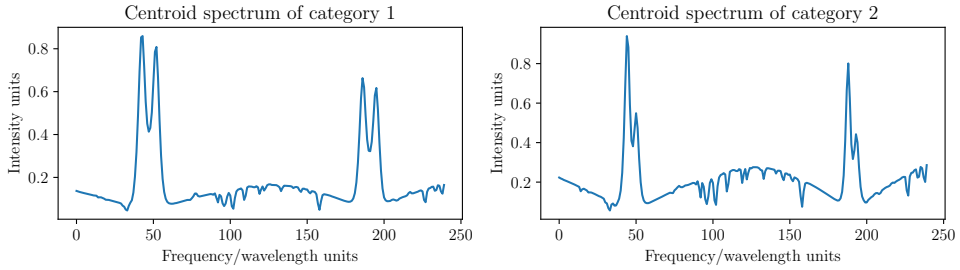


Figure 4.16: Centroid spectra for both of the two most separated clusters as obtained by the `KMeans` algorithm.

two most separated clusters of spectra and corresponding features were combined to one dataset, which was subsequently split into a train and test dataset. The normalizing flow was subsequently trained on the train split of the data. The trained flow was then tested by means of propagating the centroid spectra for both clusters of spectra contained in the train dataset through the trained normalizing flow. It was then checked, whether the predicted feature distributions for these two centroid spectra matched the exact feature values corresponding to those spectra.

[8]In this case, the `KMeans` function of the `sklearn.cluster` package was used.

The following results and figures were obtained[9] using a training sample size of 4018, a testing sample size of 1005, a batch size of 512, a hidden layer size of 256, a coupling layer amount of 6, a learning rate of 0.001, a scheduling rate of 0.999, 35 epochs to train the flow and non-randomized splitting of the data into train and test splits. This hyperparameter setting was found by manual inspection and improvement of resulting learning curves.

Figure 4.17: Learning curves for experiment 3 on the feature extraction task.

In fig. 4.17, the training process of the normalizing flows is again visualized in terms the train and test losses. These loss curves indicate signs of underrepresentativeness of the training data, since there are some significant jumps in the curves. Furthermore, the gap between the train and test loss increases with epochs, indicating further possible overfitting done by the model. The underrepresentativeness could either be caused by the relatively small datasets used to train and test the flow, or it might also be a result of the non-randomized train-test splitting procedure employed prior to training. Furthermore, the train and test loss curves seem to exhibit periodic behaviour, which could be a sign of a too high learning rate. Future experiments concerning this task should therefore use randomized train-test splitting, aswell as a larger dataset coupled with fine-tuning of the hyperparameters, especially of the learning rate.

Furthermore, in fig. 4.18, the calculated densities and correlations of feature parameters are depicted for two test observations given to the normalizing flow as context; in this case, the centroid spectra of both categories of spectra used to train the normalizing flow on were used as test observations. The blue lines in the mentioned plot indicate the

[9]The code which produced the presented results is available at `https://github.com/danielzahnd/master-thesis/tree/main/code/nf-feature-extraction-example-3`.

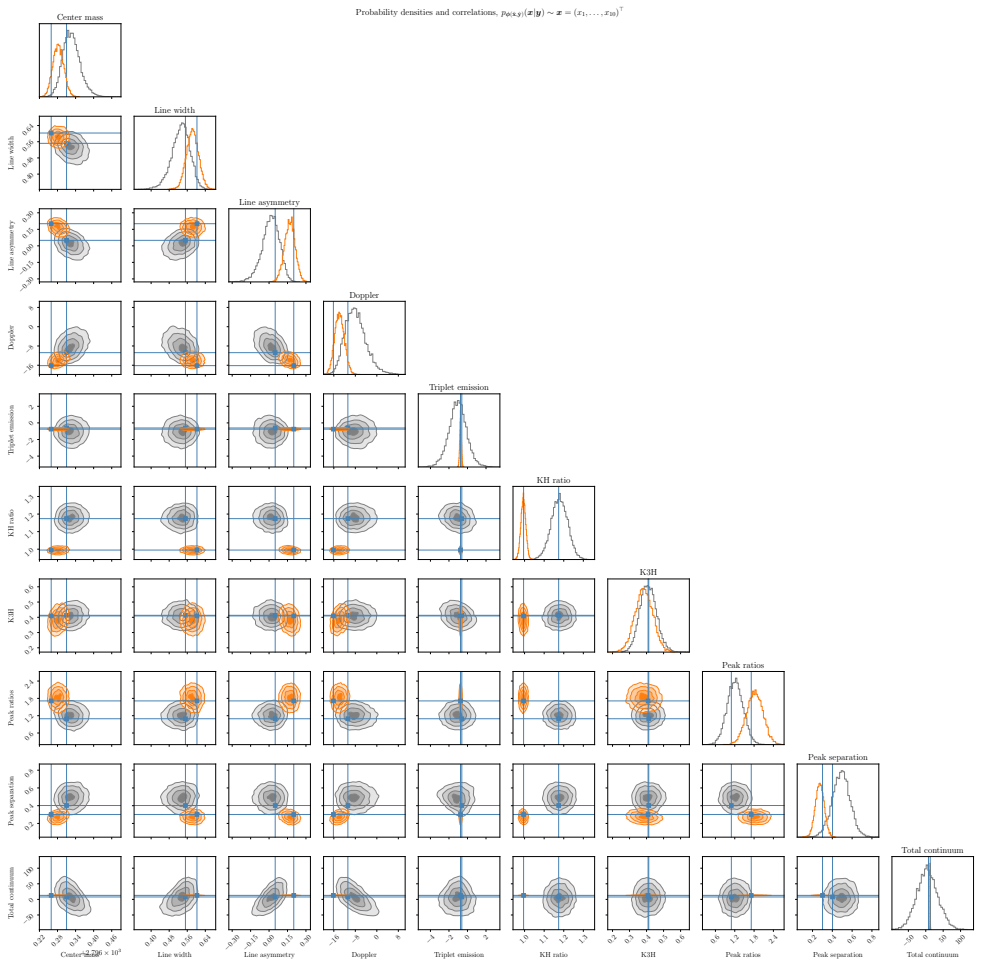


Figure 4.18: Corner plot for the calculated densities of feature parameters using the `corner` package.

true values for the feature distributions, whereas orange/grey differentiate, whether a feature distribution corresponds to either one or the other centroid spectrum. As it is evident from said figure, the normalizing flow predicts the feature distributions for both centroid spectra to agree with the true values well for only about half of the features. This could however likely be resolved by means of using a larger dataset to train the normalizing flow on and thereby acquiring better generalization properties of the flow; or optimizing the hyperparameters of the model. In general however, the normalizing flow seems to be able to differentiate, if an input spectrum belongs to one or the other category of spectra.

Chapter 5

Results and discussion

In this section, the main results of the thesis at hand are presented. Several experiments using the Milne-Eddington model as explained in section 2.3.6 as the stellar inversion model $\boldsymbol{x} = \boldsymbol{M}(\boldsymbol{y})$ were conducted, where $\boldsymbol{x}$ are the physical parameters defining the stellar atmosphere and $\boldsymbol{y}$ are Stokes parameter observations at different wavelengths and places of the latter.

If not otherwise mentioned, the experiments explained in the following material were conducted using a series of penumbra formation maps[1] obtained by the Swedish Solar Telescope (SST) situated near La Palma, Tenerife. There are a total of 16 frames (maps) in this dataset and they show the active region 13010[2] on May 16, 2022. They contain observations for all four Stokes parameters at 13 different wavelengths and a field of view (FOV) of 550 by 600 pixels; the observed wavelength points are centered around $\lambda_0 = 6302.4931\,\text{Å}$ ranging from $\lambda_{min} = 6302.2134\,\text{Å}$ to $\lambda_{max} = 6302.6934\,\text{Å}$. At λ_0 the intensity profile of Stokes I features an absorption line of neutral or singly ionized iron (Fe I).

Again, if not otherwise mentioned, the inversion model $\boldsymbol{M}$ aswell as the normalizing flow model $\boldsymbol{f}_{\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}$ for the subsequent experiments are given as follows. The atmospheric parameter data $\boldsymbol{x} = (x_1, \ldots, x_9)^\top \in \mathbb{R}^9$ featuring the probability density $p_{\boldsymbol{x}}(\boldsymbol{x})$ is connected to observational data $\boldsymbol{y} \in (y_1, \ldots, y_{52})^\top \in \mathbb{R}^{52}$ accounted for by the density $p_{\boldsymbol{y}}(\boldsymbol{y})$ via the

[1] These files can be accessed at `https://drive.google.com/drive/folders/1-W3vCJC4gEsQWW0pzwF8PbQ3erE0eGPI?usp=drive_link`.

[2] See `http://helio.mssl.ucl.ac.uk/helio-vo/solar_activity/arstats/arstats_page5.php?region=13010` for more data on this active region; website last accessed on December 20, 2023.

Milne-Eddington model

$$\boldsymbol{M} : \mathbb{R}^9 \to \mathbb{R}^{52}, \quad \boldsymbol{x} = \begin{pmatrix} |\boldsymbol{B}| \\ \theta \\ \varphi \\ v_{los} \\ v_{dop} \\ a \\ \eta_0 \\ S_0 \\ S_1 \end{pmatrix} \mapsto \boldsymbol{y} = \boldsymbol{M}(\boldsymbol{x}) = \begin{pmatrix} \boldsymbol{I}_{\lambda_{min}} \\ \vdots \\ \boldsymbol{I}_{\lambda_0} \\ \vdots \\ \boldsymbol{I}_{\lambda_{max}} \end{pmatrix}, \tag{5.1}$$

where $\boldsymbol{I}_\lambda = (I_{\lambda,n} \doteq I_\lambda/I_{\lambda,c}, Q_\lambda/I_{\lambda,n}, U_\lambda/I_{\lambda,n}, V_\lambda/I_{\lambda,n})^\top$ is the Stokes vector containing the four normalized Stokes parameters at wavelength λ, where $I_{\lambda,n}$ is the normalized intensity using the continuum intensity at wavelength λ. In the following material, the wavelength dependence index λ and the normalization index n will be omitted for simplification purposes, hence the Stokes vector at a specific wavelength λ as given above will be just written as $(I, Q/I, U/I, V/I)^\top$. In order to perform inversions of the Milne-Eddington model $\boldsymbol{M}$, a diffeomorphic mapping from $p_{\boldsymbol{x}}(\boldsymbol{x})$ representing the parameter densities to a multidimensional standard normal distribution $p_{\boldsymbol{z}}(\boldsymbol{z})$ conditioned on observational data represented by the density $p_{\boldsymbol{y}}(\boldsymbol{y})$ was to be found. This mapping was implemented by a normalizing flow

$$\boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})} : \mathbb{R}^9 \to \mathbb{R}^9, \quad p_{\boldsymbol{x}}(\boldsymbol{x}) \sim \boldsymbol{x} \mapsto \boldsymbol{z} = \boldsymbol{f}_{\boldsymbol{y},\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{x}) \sim \mathcal{N}(0,1)^9 \tag{5.2}$$

using several piecewise quadratic spline coupling layers as specified in section 3.9.1. This normalizing flow then allows for the calculation of the posterior probability density $p_{\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{x}|\boldsymbol{y})$ of atmospheric parameters $\boldsymbol{x}$ given some observation $\boldsymbol{y}$.

5.1 Experiment 1: Training on synthetic data

5.1.1 Explanation of the experiment

This first experiment using normalizing flows applied to learn the Milne-Eddington inversion model only deals with synthetic data. That is to say, observational data $\hat{\mathbf{y}} = (\boldsymbol{y}_1, \dots)^\top$ is generated using the Milne-Eddington model implementation `pyMilne` given by [de La Cruz Rodríguez, 2019]

based on a defined dataset $\hat{\mathbf{x}} = (\boldsymbol{x}_1, \dots)^\top$ of atmospheric parameter combinations. An inversion using the Milne-Eddington model of the first penumbra formation map is used to calculate mean values and standard deviations for atmospheric parameters to generate observational data from using the Milne-Eddington model $\boldsymbol{y}_j = \boldsymbol{M}(\boldsymbol{x}_j)$. The data were generated in such a way, that for each of the nine Milne-Eddington parameters values were sampled from a normal distribution with mean and standard deviation as calculated from the existing inverted map. These parameter combinations were then propagated through the Milne-Eddington model, thus obtaining associated observations. The generated dataset of parameters and associated synthetic observations was then split into train and test splits using randomized train-test splitting, on which a normalizing flow model using a piecewise rational quadratic spline architecture as provided by the `nflows` package implemented by [Durkan et al., 2019] was then trained. Note, that the Milne-Eddington model in this case was initialized for 51 wavelength points centered around the Fe I line at 6302.4931 Å instead of just 13 wavelength points as specified in eq. (5.1).

5.1.2 Results, discussion and interpretation of the experiment

The following results and figures were obtained[3] using a training sample size of 400000, a testing sample size of 100000, a batch size of 512, a hidden layer size of 32, a coupling layer amount of 5, a learning rate of 0.001, a scheduling rate of 0.999, 8 epochs to train the flow and randomized splitting of the data into train and test splits. This hyperparameter setting was found by manual inspection and improvement of resulting learning curves.

In fig. 5.1, the loss curves for the specified model is shown. Furthermore, fig. 5.2 shows a sample $\boldsymbol{y}_{test}$ from the test dataset which propagated though the trained normalizing flow yields the parameter densities as depicted in fig. 4.18.

Consider the learning curve shown in fig. 5.1 first; it shows a train loss that continuously decreases and seems to converge to a stable value after a couple of epochs. The train loss however does indicate, that the model has not learned anything of significance with respect to the ability

[3]The code which produced the presented results is available at `https://github.com/danielzahnd/master-thesis/tree/main/code/nf-milne-eddington-example-3-nflows-piecewisequadratic`.

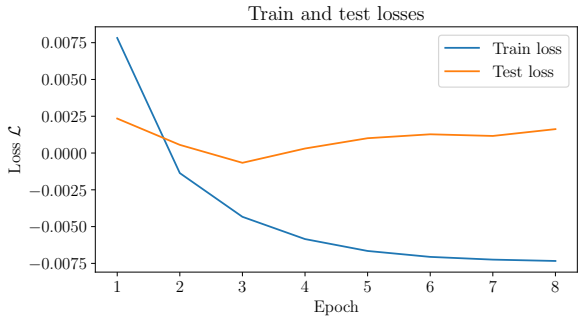


Figure 5.1: Learning curve for experiment 1 of normalizing flows applied to learn Milne-Eddington inversions.

to perform well on unseen data. With respect to fig. 3.4, this behaviour could be the result of a mixture between overfitting and underrepresentativeness of both or either one of the training or testing datasets. The latter case is to be considered as very likely, since in order for the normalizing flow to perform well on unseen data, one needs to have a big variation of data in the training data. Since 5 of the resulting parameter densities as seen in fig. 5.3 seem to be quite flat generally, this could be a hint in the direction of too little variation in the training dataset, such that the flow cannot learn sufficient information on how to do an inversion.

Moving on to fig. 5.2 and fig. 5.3, one can conclude that the ex-

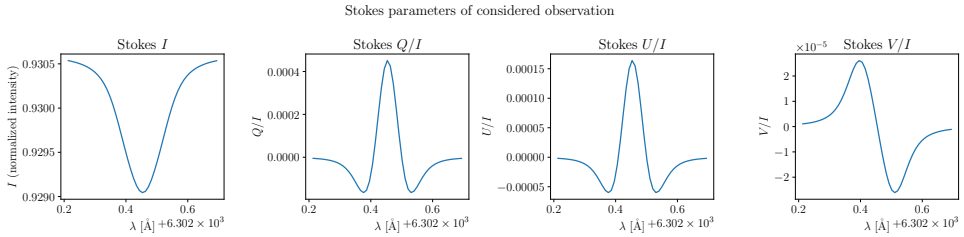


Figure 5.2: Sample Stokes parameters $\boldsymbol{y}_{test}$ propagated through the normalizing flow to obtain densities for the associated atmospheric parameters.

periment did not yield satisfactory results. That is to say, the trained normalizing flow seems to fail to predict plausible parameter densities $\boldsymbol{x} \sim p_{\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{x}|\boldsymbol{y}_{test})$ given a test observation $\boldsymbol{y}$ taken from the test dataset. Provided enough time and computational power however, the approach with training and testing on synthetic data could be promising. In order to obtain good results, one would nevertheless have to generate much larger datasets with more variation for training and furthermore fine-tune

the hyperparameters for in order to obtain good results. Due to lack of time and computational resources, this approach was not followed further by the author.

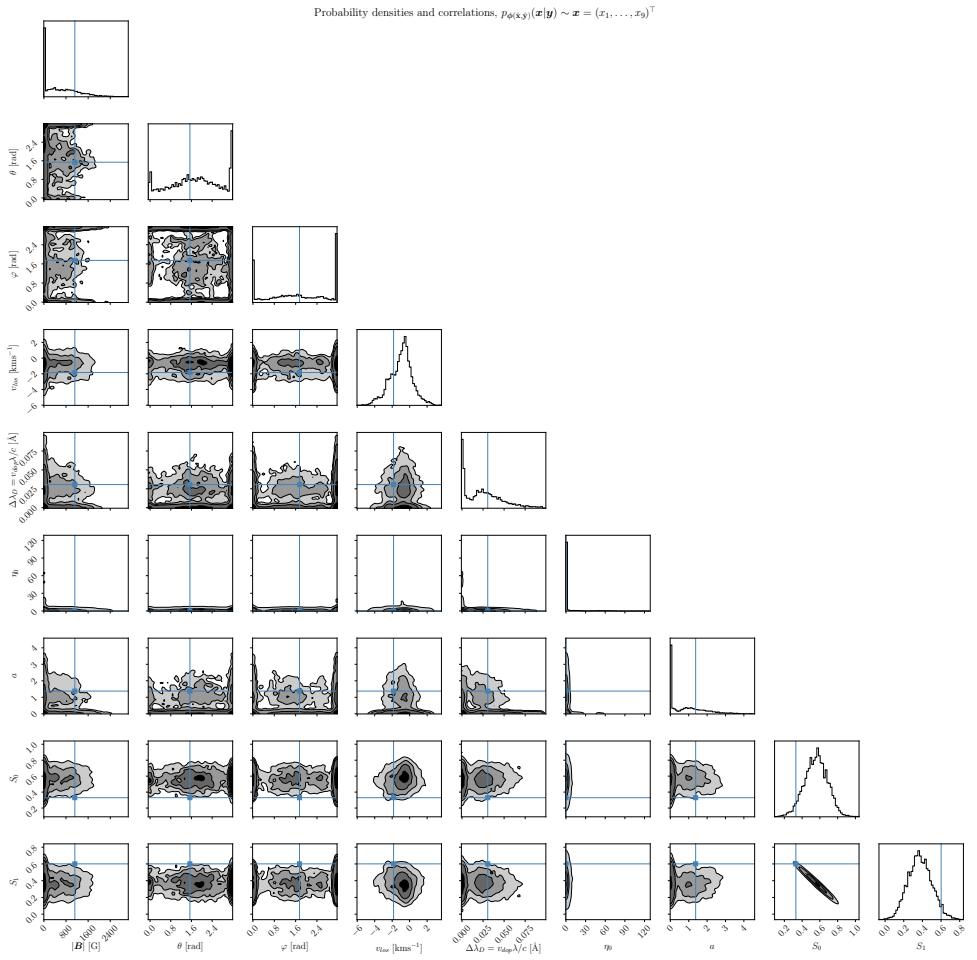


Figure 5.3: Obtained parameter densities for the sample spectra $\boldsymbol{y} = \boldsymbol{y}_{test}$ shown in fig. 5.3. The blue lines represent the true values $\boldsymbol{x}_{test}$ associated to $\boldsymbol{y}_{test}$ by $\boldsymbol{y}_{test} = \boldsymbol{M}(\boldsymbol{x}_{test})$.

Given that one knows how to generate synthetic data for an arbitrary atmospheric model however, one could possibly even think about just training a normalizing flow on a huge dataset containing an as large as possible variation of synthetic data and applying the trained flow to real observational data. However, errors due to measurement processes and instrument effects are reflected in real data; synthetic data however lacks such artifacts. Therefore, if one would train a normalizing flow solely on

synthetic data, one would have to modify the synthetic training data in such a way, that some realistic amount of noise present in real data is reflected in the synthetic data too.

5.2 Experiment 2: Single map analysis

5.2.1 Explanation of the experiment

For this second experiment, frames 0 and 1 of the penumbra formation dataset were used. Data pertaining to frame 0 was used for training a normalizing flow, while frame 1 was used for evaluating the model performance afterwards. This is to say, that map 0 (frame 0) containing the observational data $\hat{\mathbf{y}}$ was inverted using the Milne-Eddington algorithm, thus providing the associated parameter data $\hat{\mathbf{x}}$. The data $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ were split into train and test splits, on which the normalizing flow was trained. Data from frame 1 was only used to assess the performance of the trained normalizing flow for four different points as indicated in fig. 5.4 on the solar atmosphere. The considered pixel 1 ($\boldsymbol{y}_1$) represents quiet sun con-

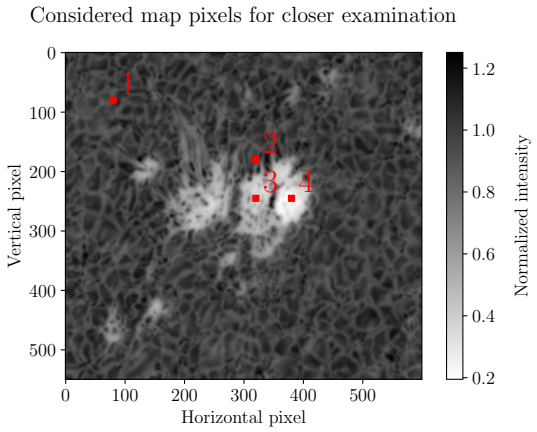


Figure 5.4: Considered map pixels for performance assessment of the trained normalizing flow for experiment 2.

ditions. Pixel 2 ($\boldsymbol{y}_2$) is located in a forming penumbra, whereas pixel 3 ($\boldsymbol{y}_3$) represents a partially developed umbra (a pore), while pixel 4 ($\boldsymbol{y}_4$) is representative of a fully developed umbra. These four pixels were chosen such that the performance of the trained normalizing flow could be assessed on areas representing the three characteristic structures of the solar atmosphere - the quiet sun, umbra and penumbra structures.

The results as obtained by the trained normalizing flow were compared to Markov Chain Monte Carlo (MCMC) sampling for the Milne-Eddington inversion process; this was done according to the explanations given in section 7.1.

5.2.2 Results, discussion and interpretation of the experiment

The following results and figures were obtained[4] using a training sample size of 297000, a testing sample size of 33000, a batch size of 512, a hidden layer size of 16, a coupling layer amount of 5, a learning rate of 0.001, a scheduling rate of 0.999, 10 epochs to train the flow and randomized splitting of the data into train and test splits. This hyperparameter setting was found by manual inspection and improvement of resulting learning curves.

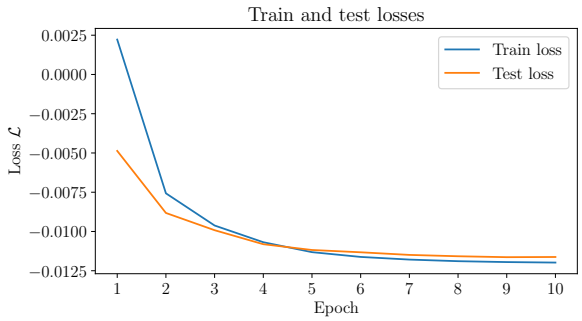


Figure 5.5: Learning curve for experiment 2 of normalizing flows applied to learn Milne-Eddington inversions.

First of all, consider the loss curve as shown in fig. 5.5. The curve shows a desirable convergence behaviour of both the train and test losses; however, one could question the magnitude of total improvement in terms of losses over the training period - the test loss only improves by about 2 % and the train loss by about 10 %. Probably the magnitude of total improvement would increase, if the training data size were even larger and more representative; because then the normalizing flow could learn more characteristic properties of the atmospheric inversion model.

[4]The code which produced the presented results is available at `https://github.com/danielzahnd/master-thesis/tree/main/code/nf-milne-eddington-example-6-nflows-piecewisequadratic`.

Furthermore, fig. 5.7 to fig. 5.10 show the resulting probability densities as predicted by both the normalizing flow (grey) and MCMC sampling (orange) for the atmospheric parameters for the test observations $\boldsymbol{y}_1$ to $\boldsymbol{y}_4$.

In order to assess if there are possible «gaps» in the training data possibly leading to bad results if data within these gaps would be processed by the trained flow afterwards, a histogram of the training data was plotted as shown in fig. 5.6 The training data show no significant

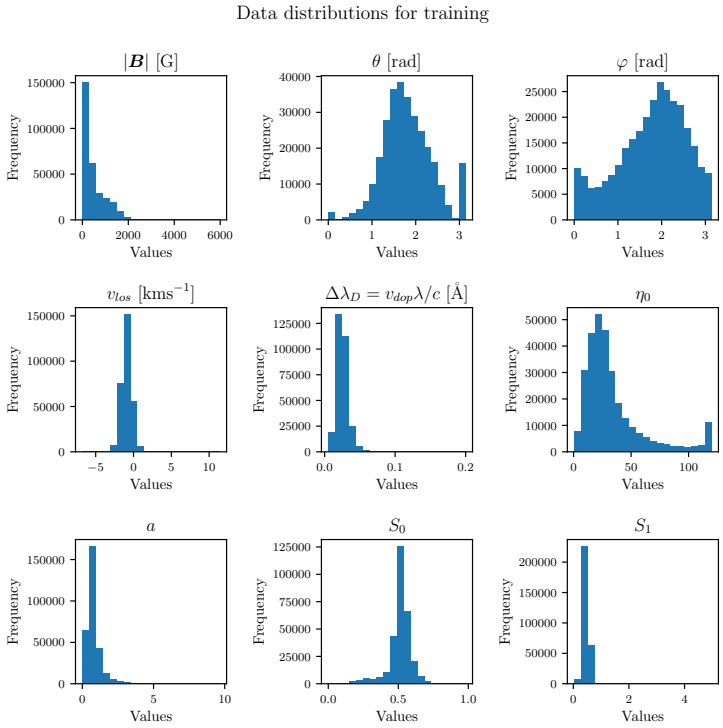


Figure 5.6: Distribution of training data for experiment 2

gaps, therefore, one can expect good performance of the trained flow for new data featuring roughly the same range of data as the training dataset.

Concentrating for starters on the similarities between all four test observation results, one can conclude, that the MCMC procedure yields narrower probability densities for the atmospheric parameters than the trained normalizing flow does, this is especially the case for the magnetic field magnitude parameter $|\boldsymbol{B}|$. This overall discrepancy between the normalizing flow and MCMC approach could be explained by means of two arguments.

First, the MCMC procedure assumes the likelihood function $p(\boldsymbol{y}|\boldsymbol{x})$ to be normally distributed - thereby an a priori assumption for the standard deviation of this normal distribution has to be made in order to sample from the posterior $p(\boldsymbol{x}|\boldsymbol{y}) \propto p(\boldsymbol{y}|\boldsymbol{x})p(\boldsymbol{x})$. Depending on how broad the normally distributed likelihood function is chosen, this could reflect in the resulting predictions for the posterior. This is to say that it might be, that a more realistic assumption for the likelihood function would have meant, that the likelihood function should have had to be chosen to be of a broader nature. The second reason could be, that the normalizing flow - having no access to the actual inversion model in contrast to the MCMC procedure - cannot learn enough information from the data in order to predict as narrow densities as the MCMC procedure does. The findings of [Díaz Baso et al., 2022, p.5] provide evidence for the second suggestion; they use many more wavelength points for the Stokes profiles than were used in the case at hand with only 13 wavelength points per Stokes profile and observation, leading to a much better agreement between the normalizing flow and MCMC results. Further investigation of this suggestion is carried out in the next experiment (experiment 3), which is exactly identical to the present experiment, but with interpolated Stokes profiles such that there are a similar amount of wavelength points per Stokes profile as used in [Díaz Baso et al., 2022, p.5].

Another general observation concerning the results of all pixels is, that the parameter densities for the line strength parameter η_0 is very broad; the line strength parameter is a measure for how pronounces an absorption line is relative to the continuum. Supposedly, this effect has to do with the scarceness of wavelength points for the Stokes profiles - it is not clear for the normalizing flow, whether fluctuations from one wavelength point to another indicate a spectral line or just a measurement noise. This reflects in a large uncertainty and hence in a broad probability density function for η_0. The MCMC procedure on the other hand provides a much narrower density. It is evident, that the magnetic field strength $|\boldsymbol{B}|$ predictions by the normalizing flow behave in the same way as the field strength parameter η_0; the MCMC method yields a much narrower density function for the magnetic field strength than the normalizing flow does. The Milne-Eddington algorithm - which the normalizing flow attempts to learn - infers the magnetic field strength using the Zeeman line splitting as given in eq. (2.38) - the clearer a separation of an absorption line into two separate lines reflects in the Stokes profiles, the stronger the magnetic field is inferred to be. Again, due to the scarceness of wavelength points, the Zeeman splitting might not be easy

for the normalizing flow to «see». Hence, a large uncertainty compared to the MCMC method - which makes explicit use of the Milne-Eddington model - results for the normalizing flow inferences of the magnetic field strength.

Now on to the peculiarities of individual results. First of all, the Stokes parameters for the quiet sun pixel show no strong polarization, as none of the Stokes profiles Q/I, U/I and V/I are pronounced strongly relative to one another. This agrees with the expectation for the quiet sun, because the quiet sun shows no macroscopic structures like sunspots, which are correlated to strong magnetic fields and thus polarized magnetic and electric fields. There seems to be quite some ambiguity in the azimuth parameter φ for the quiet sun, which again can be explained by the expectation of largely unpolarized light originating from the quiet sun.

The Stokes profiles for the penumbra pixel however show, that there seems to be a non-negligible circular polarization, as Stokes V/I is quite large in magnitude compared to Stokes Q/I and U/I accounting for linear polarization. This is to say, that there should not be a large amount of ambiguity in the results for the inclination and azimuth parameters θ and φ. This indeed reflects in the results and can be understood by recalling the sunspot sketch fig. 1.3. The light originating in the penumbra should be in a well-defined polarization state, since one would expect the magnetic field lines - and therefore also the electric field lines - to be aligned as shown in fig. 1.3; of course only given, that this sunspot model is correct.

For the weak umbra (pore) (observation 3), Stokes Q/I and V/I are roughly of the same magnitude. Again considering the Sunspot model fig. 1.3, one would expect a magnetic field roughly aligned with the surface normal of the sun, or exactly opposite - this is to say, the inclination θ would be expected to be either $\theta \approx 0\,\mathrm{rad}$ or $\theta \approx \pi\,\mathrm{rad}$. Furthermore, there should not be much ambiguity in the inclination parameter, since given the common sunspot model, the magnetic field at the center of an umbra or pore should either face inwards or outwards of the sun along the surface normal. Both of these things are observed to be the case in the results shown in fig. 5.9; the inclination is indeed approximately half of pi with little uncertainty, meaning that the magnetic field faces towards the solar interior.

As for the umbra observation, Stokes Q/I and V/I are quite pronounces compared to Stokes U/I, meaning that there should not be much ambiguity present in the inclination and azimuth parameters of

the magnetic field for this observation. This is actually reflected in the results; what is somewhat surprising is the value of the inclination being about $\theta \approx 2\,\mathrm{rad}$, rather than half of pi as one would expect for an observation of the magnetic field at the center of an umbra. This discrepancy between expectation and prediction could be explained by the inherent ≪inaccuracy≫ of the expectation. One cannot expect an umbra observed in reality to follow exactly the model one has of it, rather one must infer the model from the data. While it represents a plausible assumption, that the inclination parameter of the magnetic field is either zero or half of pi for the magnetic field at the center of an umbra, this does not have to be the case exactly.

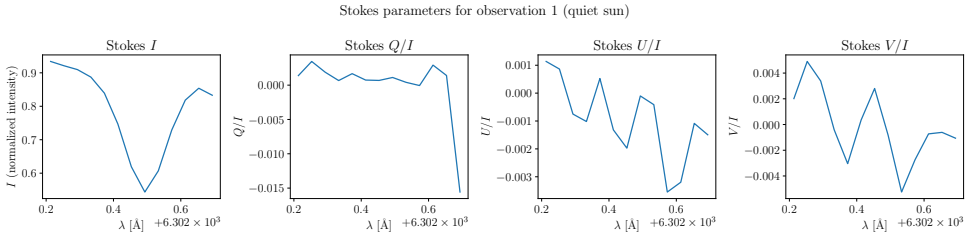
Analyzing the results as presented here, one can conclude, that a trained normalizing flow yields satisfactory results if applied to unseen data with similar ranges and variations; furthermore, the trained normalizing flow seems to reflect possible uncertainty in the inverted observations as expected. For example, if there is an observation showing Stokes profiles with for nearly unpolarized light; this should reflect in broad densities for the magnetic field inclination and azimuth parameters and hence in a large uncertainty thereof.

As for closing remarks to this section it shall be mentioned, that two other variations of the same experiment as presented here were conducted. One variation constituted of interpolating the observed Stokes profiles, such as to get more wavelength points per Stokes profile. The goal of this experiment[5] was to examine, if this would improve the performance of the trained normalizing flow as compared to the MCMC procedure. However, this experiment did not yield better results in terms of less discrepancy between the normalizing flow and MCMC results. This outcome could actually have been anticipated, since interpolating between wavelength points of some measurement does not introduce more (useful) information than is already present in the data; and on the other hand, there could even be the danger of introducing information that is not at all connected to the actual measurements and is merely an artifact of interpolation. The second variation of the presented experiment in this section was to use a different normalizing flow architecture, namely the affine coupling layer one. This second experiment[6] was conducted to assess and compare results obtained by the affine coupling layer and

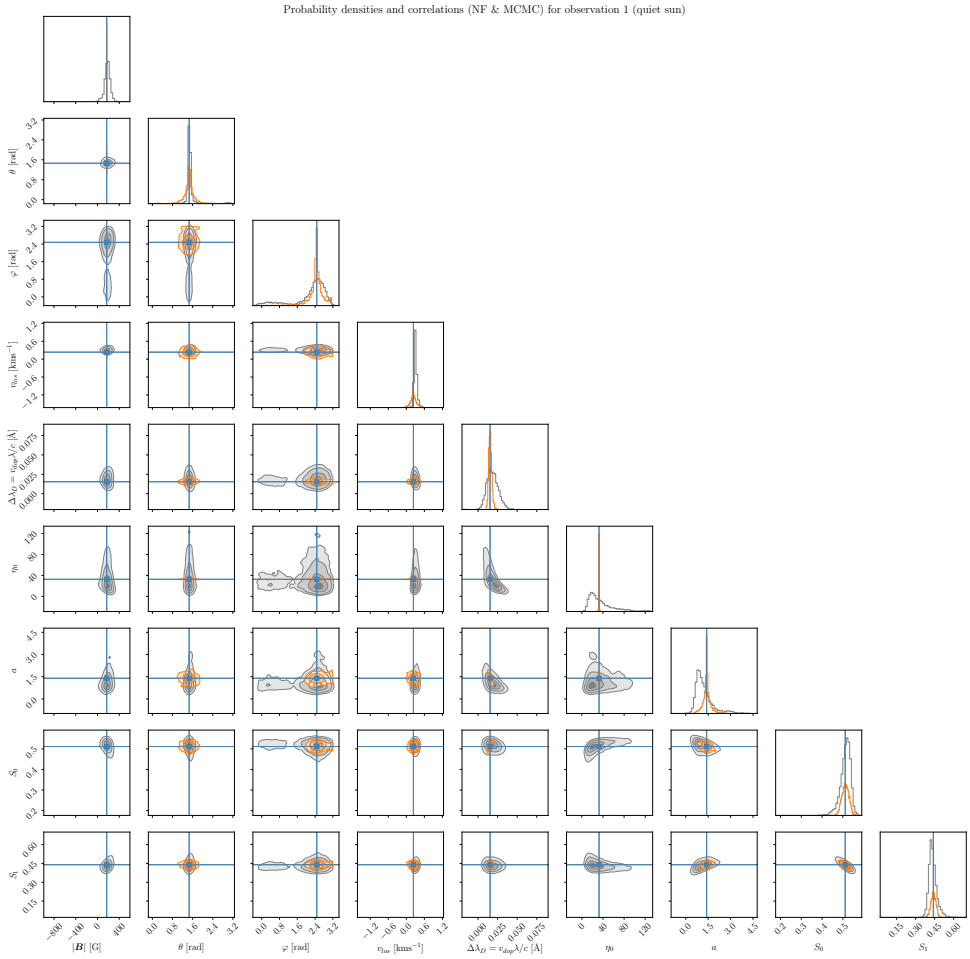
[5]The code and data for this experiment are available at `https://github.com/danielzahnd/master-thesis/tree/main/code/nf-milne-eddington-example-8-nflows-piecewisequadratic`.

[6]The code and data for this experiment are available at `https://github.com/danielzahnd/master-thesis/tree/main/code/nf-milne-eddington-example-6`.

the piecewise quadratic rational spline coupling layer architectures. As the affine coupling layer flow indeed seems to learn how to do the Milne-Eddington inversions too, it generally predicts much broader densities for the atmospheric parameters and hence larger uncertainties. This does not have to be an indication of bad choice of model - it could indicate better performance, if there actually should be expected large uncertainties given the inversion model. But the comparison with the MCMC performance of both flow architecture shows, that the piecewise quadratic rational spline coupling layer flow as used in this section is the better model choice, since the results are much closer to the MCMC results.

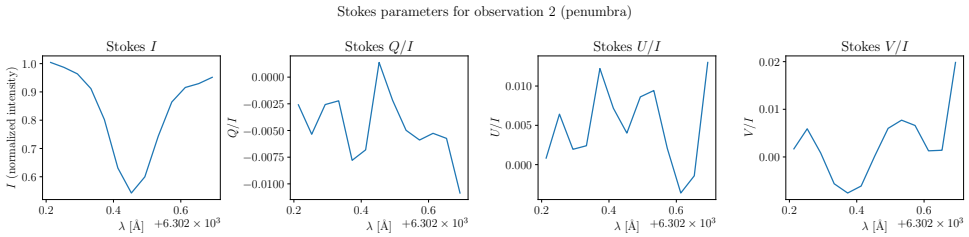


(a) Observation $\boldsymbol{y}_1$.

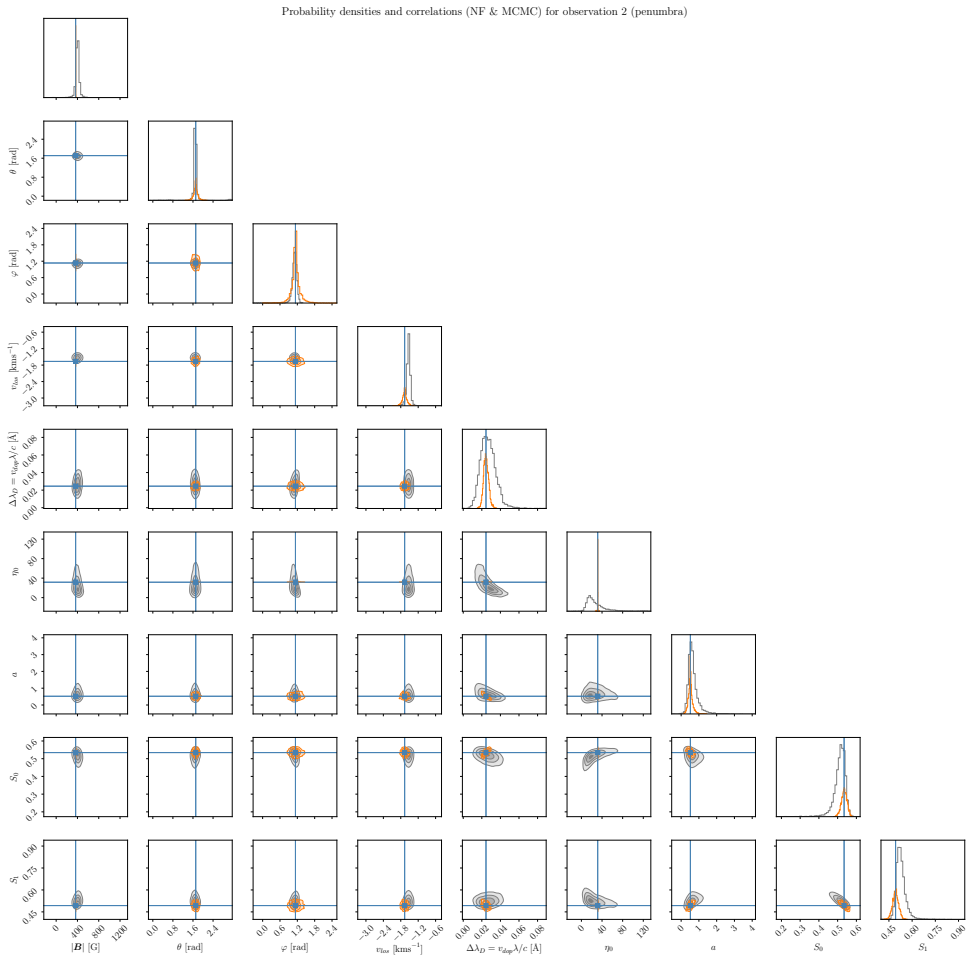


(b) Posterior distribution $\boldsymbol{x} \sim p_{\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{x}|\boldsymbol{y})$ for $\boldsymbol{y} = \boldsymbol{y}_1$.

Figure 5.7: Visualization of observation $\boldsymbol{y}_1$ and the corresponding posterior distribution as obtained by the trained normalizing flow (grey) and MCMC sampling (orange). The blue lines represent the truth, i.e. $\boldsymbol{x}_1 = \boldsymbol{M}^{-1}(\boldsymbol{y}_1)$.

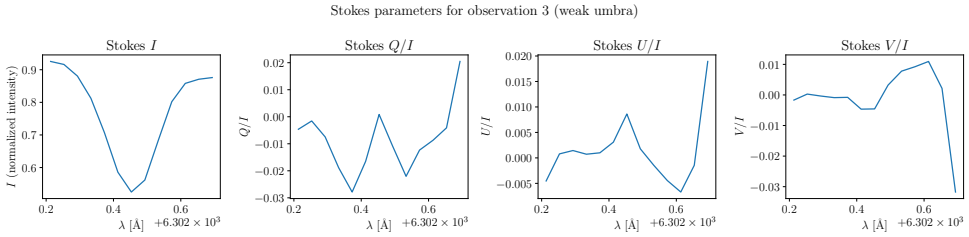


(a) Observation $\boldsymbol{y}_2$.

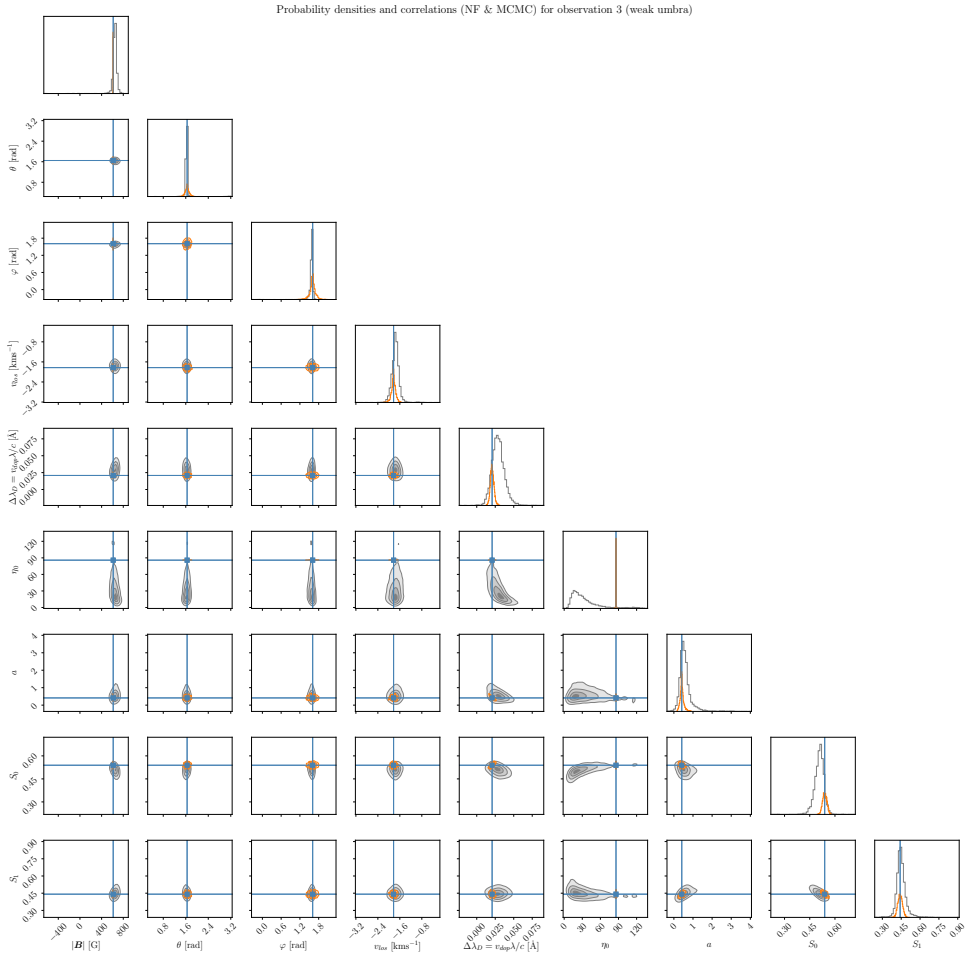


(b) Posterior distribution $\boldsymbol{x} \sim p_{\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{x}|\boldsymbol{y})$ for $\boldsymbol{y} = \boldsymbol{y}_2$. The blue lines represent the truth, i.e. $\boldsymbol{x}_1 = \boldsymbol{M}^{-1}(\boldsymbol{y}_1)$.

Figure 5.8: Visualization of observation $\boldsymbol{y}_2$ and the corresponding posterior distribution as obtained by the trained normalizing flow (grey) and a MCMC sampling (orange).

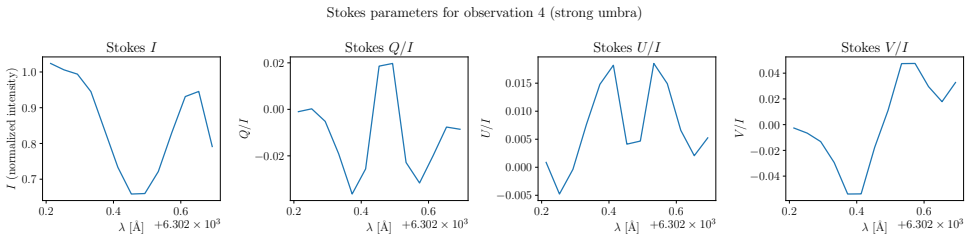


(a) Observation $\boldsymbol{y}_3$.

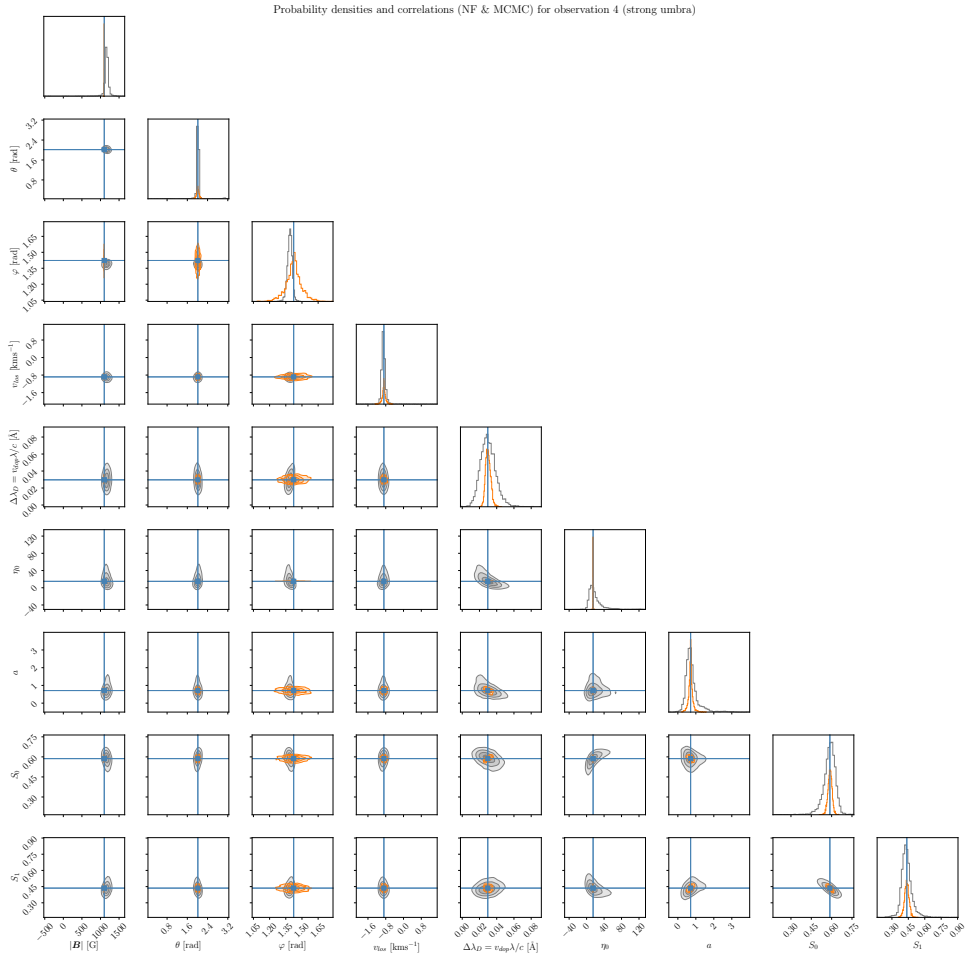


(b) Posterior distribution $\boldsymbol{x} \sim p_{\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{x}|\boldsymbol{y})$ for $\boldsymbol{y} = \boldsymbol{y}_3$.

Figure 5.9: Visualization of observation $\boldsymbol{y}_3$ and the corresponding posterior distribution as obtained by the trained normalizing flow (grey) and a MCMC sampling (orange). The blue lines represent the truth, i.e. $\boldsymbol{x}_1 = \boldsymbol{M}^{-1}(\boldsymbol{y}_1)$.



(a) Observation $\boldsymbol{y}_4$.



(b) Posterior distribution $\boldsymbol{x} \sim p_{\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{x}|\boldsymbol{y})$ for $\boldsymbol{y} = \boldsymbol{y}_4$.

Figure 5.10: Visualization of observation $\boldsymbol{y}_4$ and the corresponding posterior distribution as obtained by the trained normalizing flow (grey) and a MCMC sampling (orange). The blue lines represent the truth, i.e. $\boldsymbol{x}_1 = \boldsymbol{M}^{-1}(\boldsymbol{y}_1)$.

5.3 Experiment 3: Multiple map analysis

5.3.1 Explanation of the experiment

As a third experiment, frames 0, 11 and 19 of the penumbra formation dataset were used. Frame 0 was only used for training a normalizing flow, whereas frames 11 and 19 were solely used for assessing the performance of the normalizing flow after training. Therefore, map 0 (frame 0) comprising of the observational data $\hat{\mathbf{y}}$ was inverted using the Milne-Eddington algorithm, providing the corresponding parameter data $\hat{\mathbf{x}}$. The dataset consisting of $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ was subsequently slit into a train and test splits as usual, on which the normalizing flow was then trained.

Defining frame 11 as test map 1 and frame 19 as test map 2, the trained normalizing flow was the used to invert all observational data contained in these maps. The goal of this experiment comprised of assessing the trained normalizing flow by a direct comparison with Milne-Eddington inversions of the same test maps. The question remains, how the inversion process is actually carried out using the trained normalizing flow. Now, consider some observation $\boldsymbol{y}_p$ for a certain pixel on a map. Using the trained normalizing flow, the posterior distribution $\boldsymbol{x} \sim p_{\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{x}|\boldsymbol{y}_p)$ can be ascertained by taking samples $\boldsymbol{z} \sim \mathcal{N}(0,1)^9$ from a multivariate standard normal distribution with diagonal covariance and propagating these samples through the inverse normalizing flow as $\boldsymbol{x} = \boldsymbol{f}^{-1}_{\boldsymbol{y}_p,\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{z})$, where as always $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ denote the training data and $\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})$ the neural network parameters depending on the training data. This procedure however yields a probability density for all nine atmospheric parameters per observation $\boldsymbol{y}_p$. To obtain a single solution to the inversion problem rather than a probability density, the used approach as visualized in fig. 5.11 involved selecting the parameter value with the highest probability as the solution for each atmospheric parameter and observation. The standard deviation of the densities for each parameter were taken as a measure of the uncertainty of inversion, henceforth referred to as the error of the normalizing flow inversions.

5.3.2 Results, discussion and interpretation of the experiment

The following results and figures were obtained[7] using a training sample size of 297000, a testing sample size of 33000, a batch size of 512, a hidden

[7]The code which produced the presented results is available at `https://github.com/danielzahnd/master-thesis/tree/main/code/`

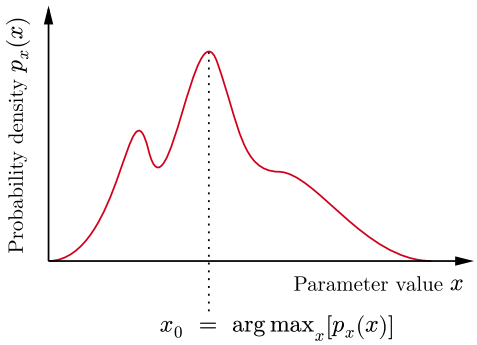


Figure 5.11: Determining the solution of an inversion problem using normalizing flows. Suppose, $x \sim p_x(x)$ is the output of the normalizing flow. In order to get the most probable solution x_0 as given by the normalizing flow, one can take x_0 to be the argument of maximal probability of $p_x(x)$.

layer size of 16, a coupling layer amount of 5, a learning rate of 0.001, a scheduling rate of 0.999, 10 epochs to train the flow and randomized splitting of the data into train and test splits. This hyperparameter setting was found by manual inspection and improvement of resulting learning curves.

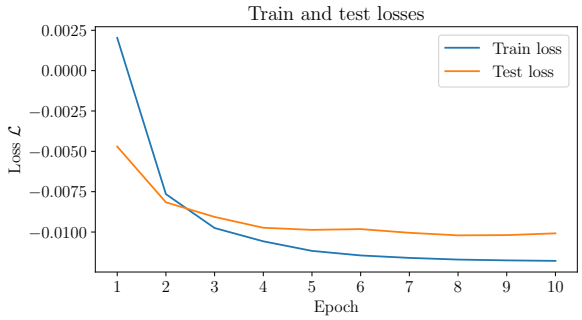


Figure 5.12: Learning curve for experiment 3 of normalizing flows applied to learn Milne-Eddington inversions.

Consider the loss curve as shown in fig. 5.12. The test loss curve shows an improvement of largely 40 %, while the train loss improves over 40 %. While the train loss shows a steady and nice convergence after roughly 7 epochs, the test loss does not seem do decrease anymore after epoch 7, while there remains quite a significant gap between both curves. Regarding fig. 3.4, the behaviour of the train and test loss curves presented here is not ideal and might show a slight tendency of overfitting.

`nf-milne-eddington-example-2-nflows-piecewisequadratic`.

The inversion results of test maps 1 and 2 as obtained by the Milne-Eddington algorithm aswell as the trained normalizing flow are depicted in fig. 5.13 and fig. 5.14. In order to locate where umbras, penumbra(s) and quiet sun regions in these results are situated, one can consider fig. 5.4 as an overview. At first sight, one can see that the trained normalizing flow indeed performs its job well, as the normalizing flow (NF) inversions look very similar to the Milne-Eddington (ME) inversions. In general, it can be seen that the normalizing flow has the tendency to smooth out edges, corners, and sharp transitions - this property of the normalizing flow shall henceforth be referred to as the smoothing property. The explanation for this effect is to be found within the technique of inverting the solar atmosphere using normalizing flows as explained above and visualized in fig. 5.11. As stated, the normalizing flow gives a probability density for each atmospheric parameter and observation $\boldsymbol{y}_p$; the argument of maximal probability for each parameter is then taken as the solution to the inversion of observation $\boldsymbol{y}_p$. Taking the argument of maximal probability involves finding maxima of $\boldsymbol{x} \sim p_{\boldsymbol{\phi}(\hat{\mathbf{x}},\hat{\mathbf{y}})}(\boldsymbol{x}|\boldsymbol{y}_p)$. Because the probability densities for the parameters can be quite broad functions, this procedure introduces some amount of uncertainty to the inversion process. Suppose, one has a nearly flat probability distribution for a parameter; then, the uncertainty on the argument of maximal probability is high, because of the flatness of the distribution it is not very clear, where the probability density actually has its maximum value. Hence, the smoothing property of normalizing flows used for inversions of stellar atmospheric models results. In order to reduce this smoothing effect, one would need to provide more information to be learned by the normalizing flow, thus enabling it to produce narrower posterior distributions and hence to reduce the uncertainty in finding the most probable parameter combination as a solution to the inversion problem.

Considering the magnetic field magnitude $|\boldsymbol{B}|$, the errors tend to be higher, where the magnetic field is stronger. For example, the errors are quite high in the pixels pertaining to an umbra or pore, where the magnetic field is strong. Concerning the inclination parameter it is evident, that the errors are highest on the quiet sun regions generally, aswell as in filaments of the forming penumbra located roughly around vertical pixel 220 and horizontal pixel 330. Considering that the electric and magnetic fields in quiet sun regions are not expected to be significantly polarized, it is no surprise that the errors on these regions are quite high, since the normalizing flow hence has difficulties in constraining the resulting probability density for the inclination. Comparing the inclination results

with the results for the magnetic field strength one can see, that generally speaking the errors in the inclination results are high, where the magnetic field is weak, relatively speaking. In an umbra for example, the inclination is quite well-defined, whereas for the quiet sun this is not the case. From the inclination results one concludes, that the magnetic field in the prominent umbra of the map faces towards the interior of the sun at $\theta \approx \pi$ rad.

A particular peculiarity can be ascertained for the damping parameter a; the error for this parameter seems to almost be a copy of Stokes I as visualized in fig. 5.4. Going back to the definitions of this parameter in eq. (2.62) and eq. (2.63) one realizes, that a high value of a means strong damping of spectral lines. Again comparing the error map for parameter a with the magnetic field strength map it becomes clear, that the normalizing flow has difficulties determining the damping parameter a, where the magnetic field is strong and where Stokes I is low. The reason for this difficulty is likely explained by the latter, as the profile for Stokes I represents a possible absorption line. If Stokes I however is not sufficiently high, the normalizing flow probably experiences difficulty in assessing what kind of absorption line there is to see, e.g. if it is broad, narrow, pronounced or strongly damped. Likely, the performance of the normalizing flow regarding a could be improved by using more wavelength points for the Stokes profiles.

Before concluding, a final remark for this experiment shall be made about the performance of the normalizing flow on the line strength parameter η_0. As one can see from the results fig. 5.13 and fig. 5.14, the smoothing property of the normalizing flow seems to have a stronger effect on this parameter than on others, and the overall performance is not very good. Likely, this is also explained by the scarceness in wavelength points of the data, because in order to assess the line strength of a spectral line, there must be some minimal threshold in terms of resolution for the line - in the worst case, a spectral line would not even be visible, if there were not enough wavelength points.

In conclusion however, the normalizing flow seems to have learned how to do Milne-Eddington inversions sufficiently well to serve as a tool for astrophysicists investigating the solar atmosphere and wanting to gain first insights to atmospheric parameters pertaining to some available observations. As [Díaz Baso et al., 2022] have shown, normalizing flows can also learn more complex inversion algorithms than the relatively simple Milne-Eddington model. Hence, normalizing flows can provide a tool to accelerate gaining first insights into solar atmosphere inversions, because

if a single map has been inverted by a time-consuming complex algorithm of choice, a less time-consuming normalizing flow can be trained on the obtained data and applied to new data of similar variation; for example a sequence of maps featuring the same FOV showing the evolution of some structure of interest, as in the penumbra formation dataset used in the experiment underlying this section.

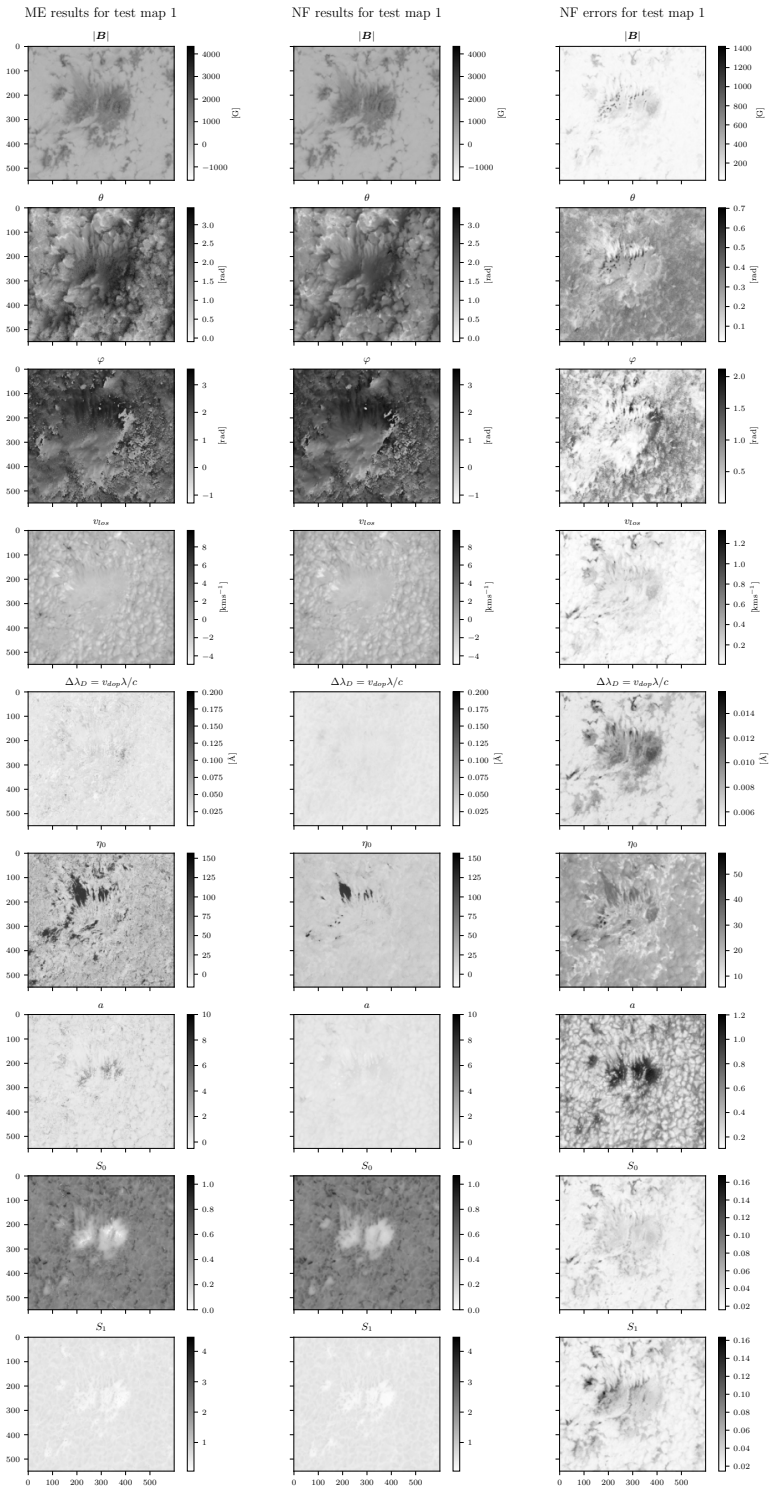


Figure 5.13: Inversion results for test map 1 (frame 11 of the penumbra formation dataset).

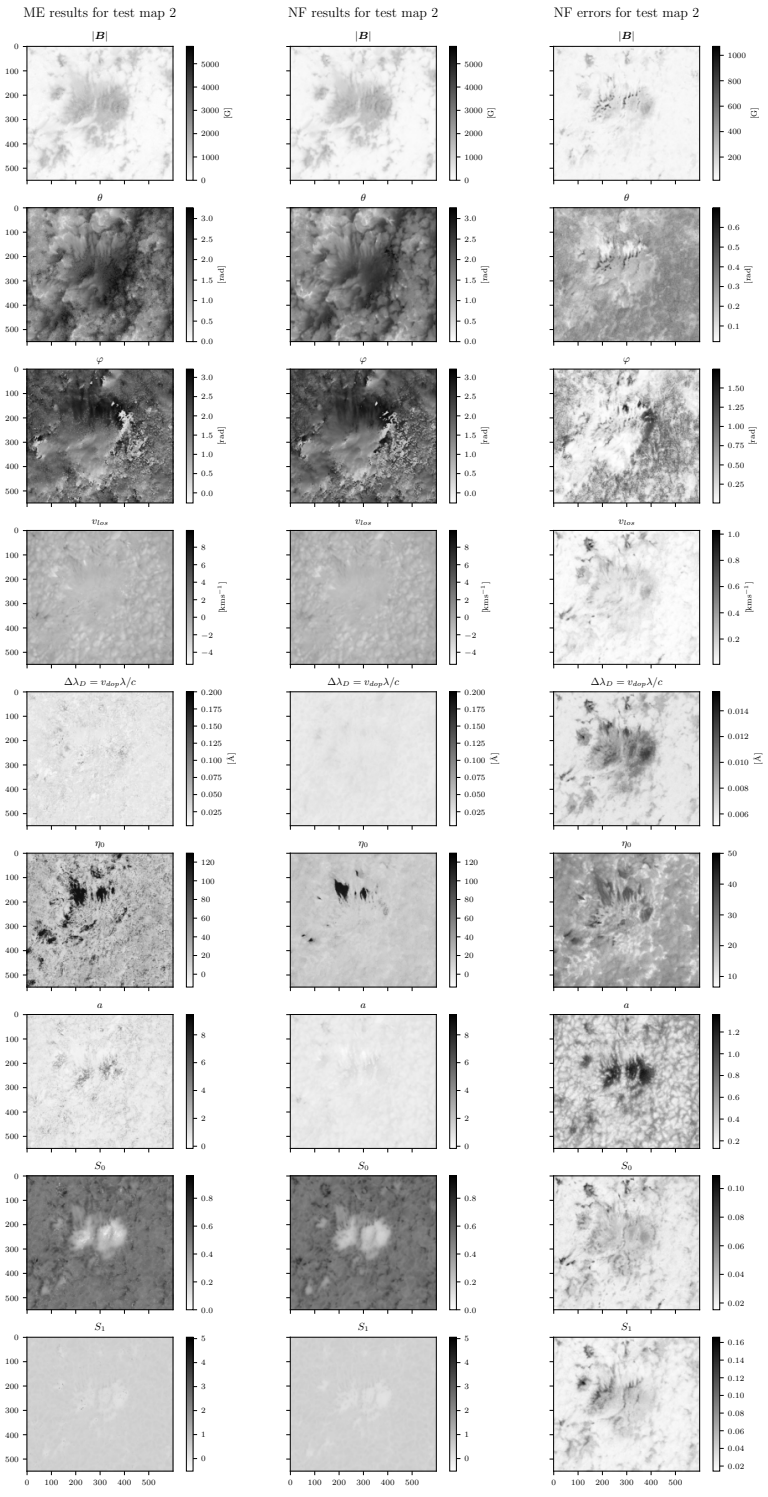


Figure 5.14: Inversion results for test map 2 (frame 19 of the penumbra formation dataset).

5.4 Experiment 4: Multiple map analysis using a different map for training

5.4.1 Explanation of the experiment

The fourth experiment conducted on normalizing flows applied to learn stellar atmosphere inversions is identical to the previous experiment, but with a different training dataset. That is to say, that frames 0, 11 and 19 of the penumbra formation dataset were used as testing maps for the trained normalizing flow, whereas the flow itself was trained on observations and corresponding parameter values of a different active region, namely a frame[8] of the active region 13014[9] observed on May 17, 2022. This observation shall be referred to in the following material as the sunspot map. For the remainder of this section, the testing frame shall be called the test map, whereas frames 0, 11, and 19 of the penumbra formation dataset shall be denoted as test map 1, test map 2 and test map 3. The training map also features the Fe I line at the same wavelength points as the test maps. In fig. 5.15, a comparison between the training map and one of the testing maps can be seen. The training map con-

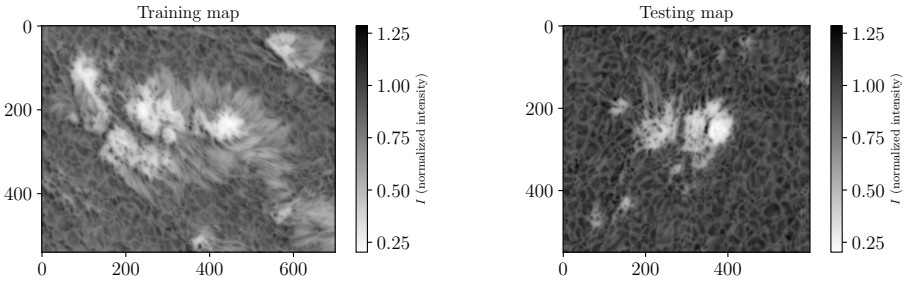


Figure 5.15: Comparison between the training map and one of the test maps. Note, that the plots show the normalized Stokes I values at a wavelength of $\lambda_{min} = 6302.2134\,\text{Å}$.

sisting the observational data $\hat{\mathbf{y}}$ was inverted using the Milne-Eddington algorithm, providing the associated parameter data $\hat{\mathbf{x}}$. The obtained dataset consisting of $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ was split into train and test splits, whereon the normalizing flow was subsequently trained. The test split obtained

[8]This file can be accessed at `https://drive.google.com/drive/folders/1AM6oA1mLYQ_DtIlSv52aYXDNDTygRQyq?usp=drive_link`.

[9]See `http://helio.mssl.ucl.ac.uk/helio-vo/solar_activity/arstats/arstats_page5.php?region=13014` for more data on this active region; website last accessed on December 23, 2023.

accordingly was solely used in the training process, such that the train and test losses could be analyzed to assess the convergence behaviour of the model. Apart from what was mentioned, the experiment at hand was conducted exactly as experiment 3 explained above in section 5.3.1.

5.4.2 Results, discussion and interpretation of the experiment

The following results and figures were obtained[10] using a training sample size of 340200, a testing sample size of 37800, a batch size of 512, a hidden layer size of 16, a coupling layer amount of 5, a learning rate of 0.001, a scheduling rate of 0.999, 10 epochs to train the flow and randomized splitting of the data into train and test splits. This hyperparameter setting was found by manual inspection and improvement of resulting learning curves.

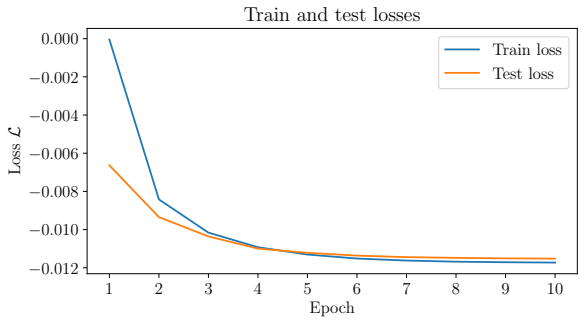


Figure 5.16: Learning curve for experiment 4 of normalizing flows applied to learn Milne-Eddington inversions.

A consideration of the train and test loss curves as given for the experiment at hand in fig. 5.16 shows, that those curves behave in a desirable way. The train and test loss both converge to a nearly constant value after about 7 epochs with little spacing between one another. Surely, the normalizing flow model could still be improved by increasing the train and test datasets, aswell as fine-tuning the hyperparameters for training; but the model as used in this experiment can be considered as a good start, given the shown train and test loss curves.

[10]The code which produced the presented results is available at `https://github.com/danielzahnd/master-thesis/tree/main/code/nf-milne-eddington-example-4-nflows-piecewisequadratic`.

The three plots fig. 5.18, fig. 5.19 and fig. 5.20 show the inversion results of test maps 1, 2 and 3 as obtained by both the Milne-Eddington algorithm aswell as the trained normalizing flow.

A striking first observation would be, that the smoothing property of the normalizing flow seems to be stronger in this case, than in the previously presented case of experiment 3 with the results fig. 5.13 and fig. 5.14. As the smoothing effect results from the probabilistic nature of atmospheric parameter results as obtained by the normalizing flow, the density functions obtained in the case at hand can be expected to be broader and less constrained than in the previous case of experiment 3. This is because in the previous case, the normalizing flow was trained on a very similar variation of data as found in the test maps; but as for the present case, the variation of data in the training dataset deviated more from the one present in the test maps. Consider fig. 5.17 for a visualization of this explanation, where one can see that the variation of training and testing data show deviations. Critical to a good performance of the normalizing flow on the test maps is, that the variation of data in the test maps should be covered by the variation of data in the training map. Regarding the plot fig. 5.17, one would have some train data, wherever there appears to be some test data in the ideal case. In this ideal case therefore, there should not be any yellow (test data) histogram bars, where one does not have any blue bars (train data). However for the case at hand, this ideal condition does not seem to be met for all data. There is almost no training data for inclination values θ between 0 rad and roughly 1 rad. One can distinguish other parameter ranges, where there is nearly to none training data despite there being test data within these ranges, for example for the damping parameter a and the intensity parameter S_0. One can therefore conclude, that the increased smoothing effect compared to the results of experiment 3 might indeed be correlated or even caused by the non-ideal coverage of test data variance by train data variance.

Apart from the stronger smoothing effect as compared to experiment 3, the results seem to be of similar quality, except for the azimuth results. For the azimuth parameter φ, the normalizing flow in the present case seems to tend to more «extreme» solutions. This is to say, that the normalizing flow tends to render even larger solutions for a large true value of φ, and conversely even lower solutions for low true values of φ. Furthermore, the smoothing property is particularly strong for the normalizing flow. However, the plot for the errors of the azimuth parameter helps to assess this behaviour. Generally speaking, the error is large,

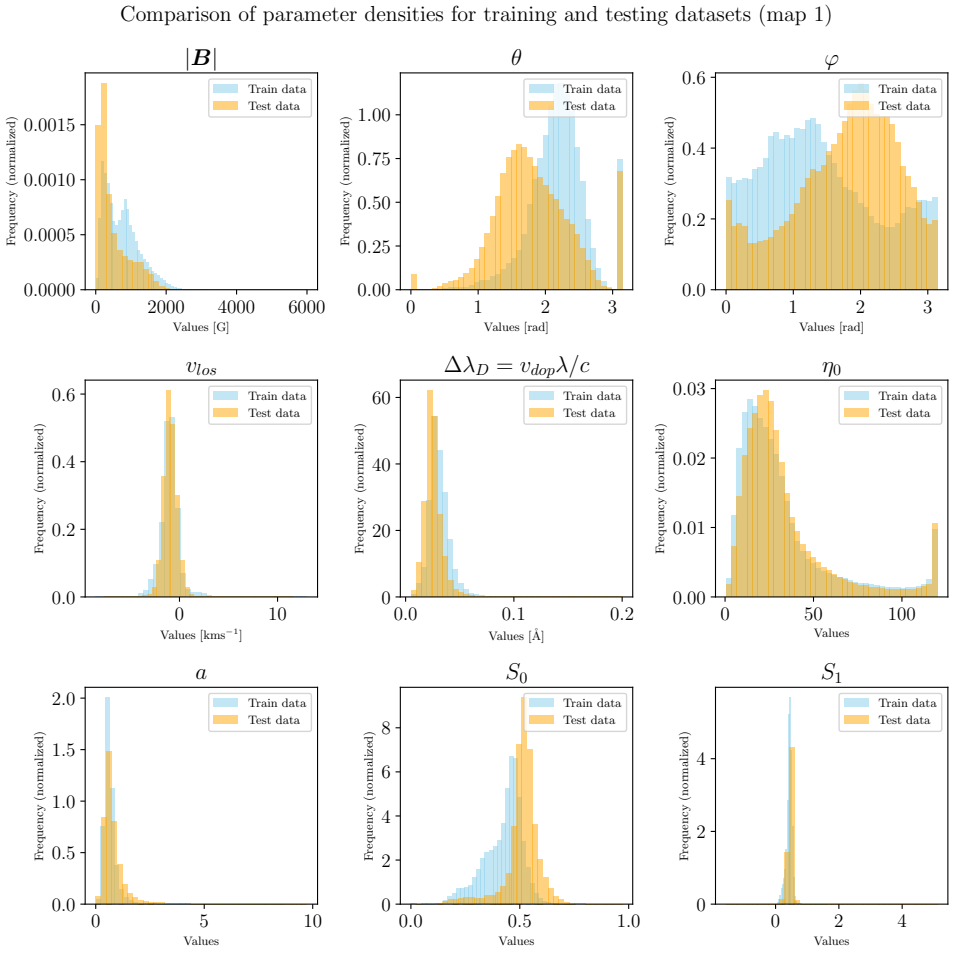


Figure 5.17: Visualization of variations in training and testing data. This plot shows data from the sunspot map as training data; and data of frame 0 (test map 1) from the penumbra formation dataset as testing data.

wherever the normalizing flow is seen to perform bad as compared to the ≪true≫ solution as given by the Milne-Eddington model. A possible explanation for this behaviour might be, that the azimuth predictions are somehow correlated by the normalizing flow with the inclination parameter, where the coverage of test data by the train data is not ideal. But apart from this suggestion, the author however does not have any conclusive explanation to offer for why the flow does not perform very well on the azimuth parameter; further investigation would be needed to answer that question. A possible experiment to resolve this would be to train the normalizing flow on a dataset with better coverage of the

inclination parameter θ.

For final remarks to this section, note that the normalizing flow in the present case seems surprisingly to perform better on the line strength parameter η_0. The smoothing property of the flow appears to be not as strong as for experiment 3; and the flow was able to identify more areas of strong line strength. As for the peculiar behaviour of the azimuth parameter, the author cannot offer a conclusive explanation for this at present stage.

Consider the approach of using Stokes profile data and the associated atmospheric parameters obtained by using a potentially complex inversion algorithm to train a normalizing flow, and subsequently applying it for inversion purposes on other arbitrary Stokes profile observational data of the same wavelengths featuring roughly similar structures as seen in the training data. If such an approach would succeed, normalizing flows could turn out to be a very useful tool for accelerating stellar atmospheric inversions for researchers in solar and stellar physics. The goal of the experiment as presented here was to show, if this approach could be worthwhile or not. In conclusion, the results for the simple approach as presented here can indeed be regarded as evidence, that a more general approach involving more complex inversion algorithms to be learned by a normalizing flow could be a valuable tool not only for accelerating stellar atmospheric inversions, but also to gain insight into uncertainties of the obtained inversion solutions.

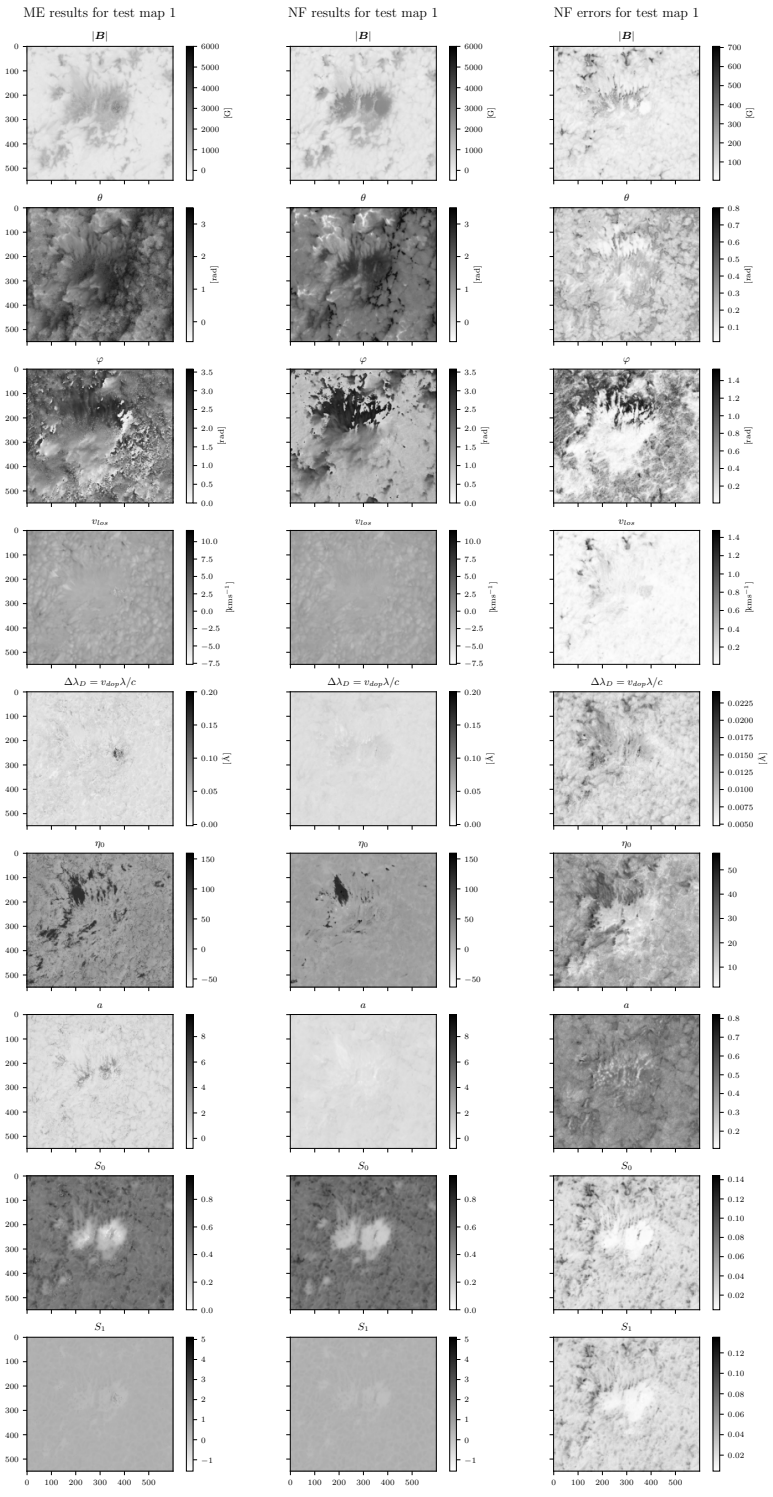


Figure 5.18: Inversion results for test map 1 (frame 0 of the penumbra formation dataset).

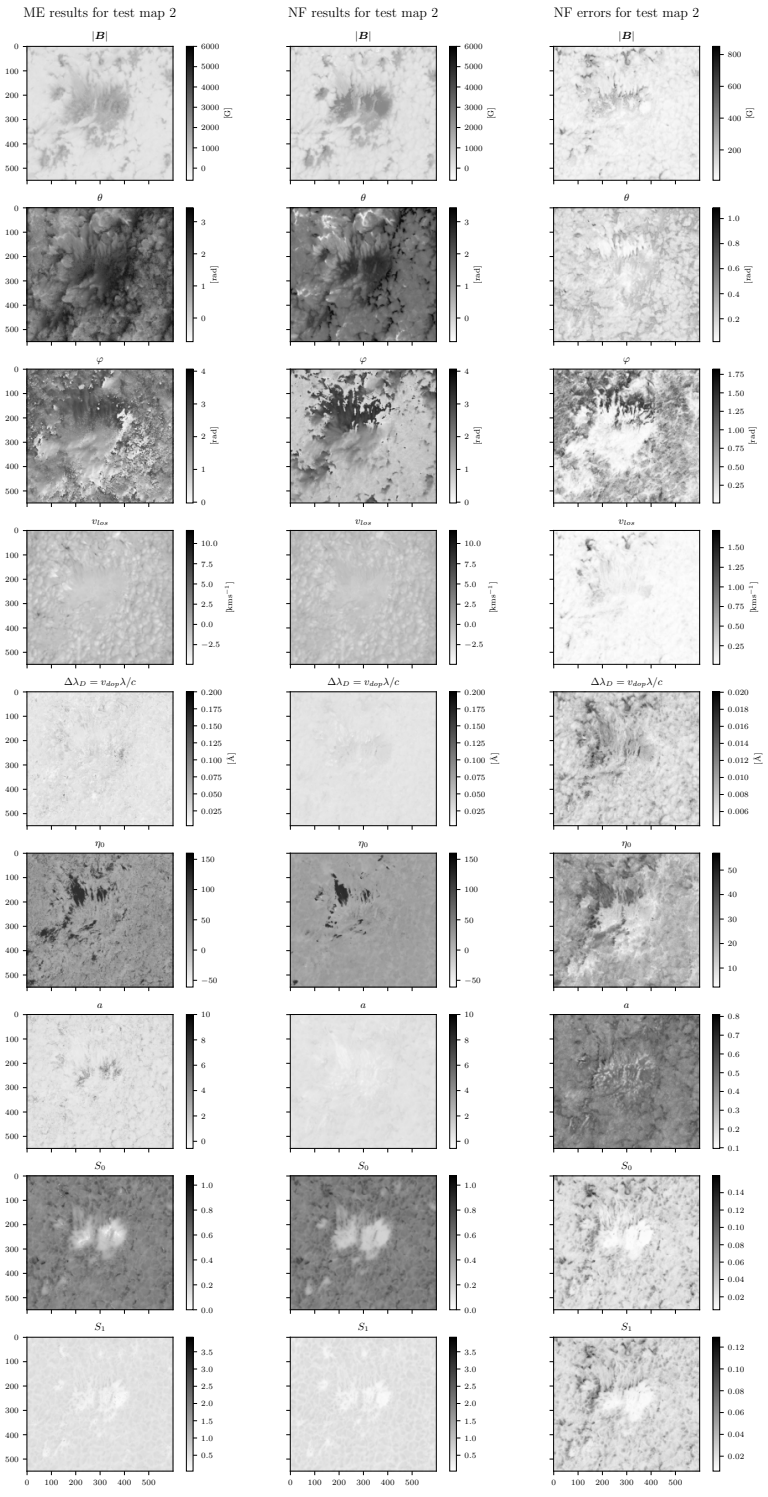


Figure 5.19: Inversion results for test map 2 (frame 11 of the penumbra formation dataset).

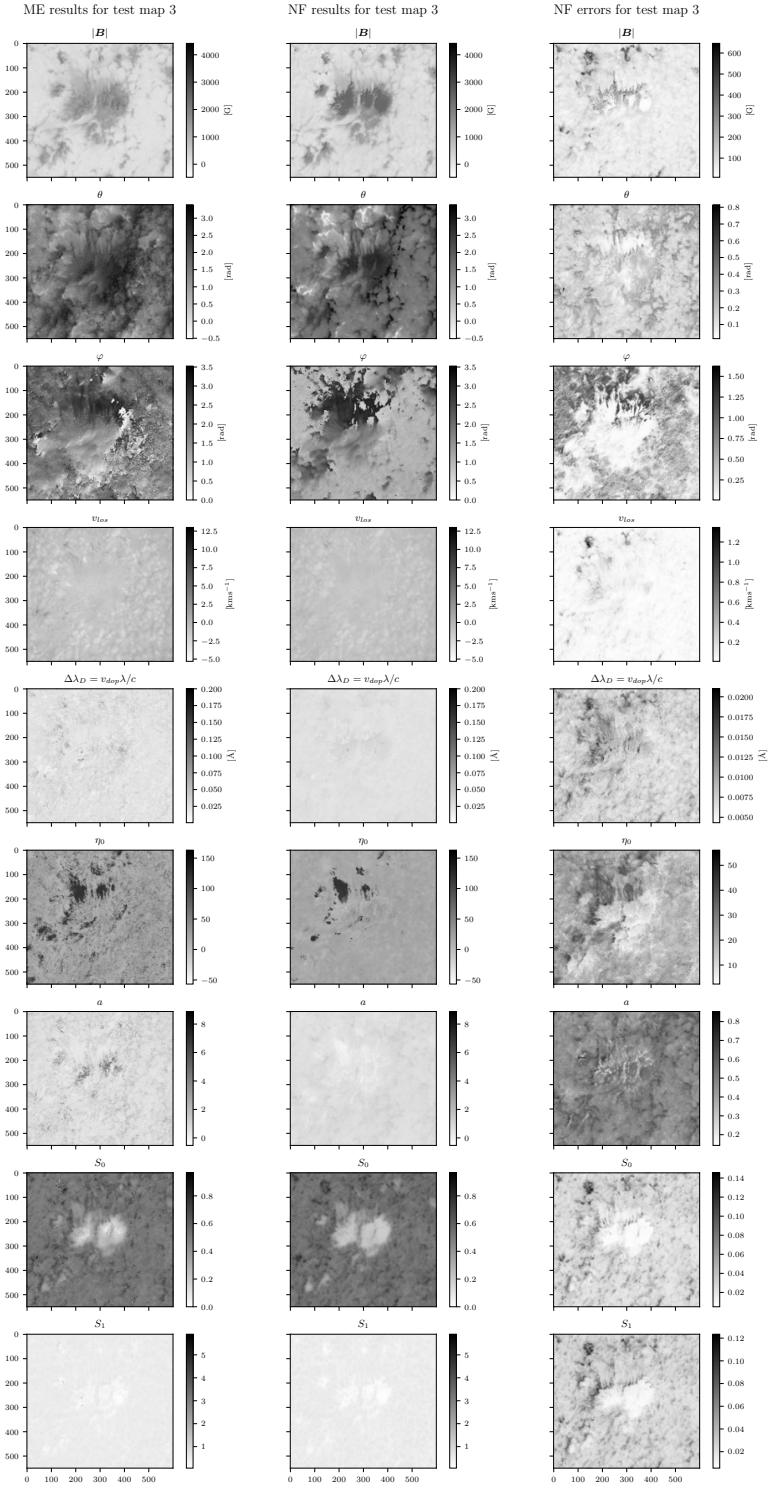


Figure 5.20: Inversion results for test map 3 (frame 19 of the penumbra formation dataset).

5.5 Experiment 5: Improving normalizing flow performance by enhancing the training dataset

5.5.1 Explanation of the experiment

It was suggested in the comments for several prior experiments, that increasing the training dataset and especially the variance within in it could improve the performance of a normalizing flow on new, unseen data. Hence, the experiment underlying this section was dedicated to explore this approach, by iteratively increasing the training dataset, while the testing dataset remained the same throughout the experimentation process.

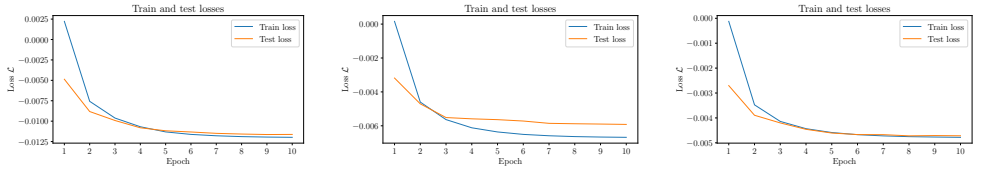
This experiment consisted of training and evaluating three normalizing flow models. The first normalizing flow model was trained on the observational data $\hat{\mathbf{y}}$ and associated parameter data $\hat{\mathbf{x}}$ obtained from frame 0 of the penumbra formation dataset. The second normalizing flow model was trained on observational data and associated parameter data from both frame 0 and frame 1; furthermore, the third flow model was trained on frames 0, 1, and 2. All of these three normalizing flow models were subsequently evaluated on the last frame (frame 19) of the penumbra formation dataset. This is to say, that the trained normalizing flows were used to invert the observational data of frame 19 as explained in section 5.3.1, which made a comparison by the «true» inversion results obtained by the Milne-Eddington algorithm possible.

5.5.2 Results, discussion and interpretation of the experiment

The following results and figures were obtained[11] using a training sample sizes of 297000, 594000, 891000, testing sample sizes of 33000, 66000, 99000, a batch size of 512, a hidden layer size of 16, a coupling layer amount of 5, a learning rate of 0.001, a scheduling rate of 0.999, 10 epochs to train the flow and randomized splitting of the data into train and test splits. These hyperparameter settings were found by manual inspection and improvement of resulting learning curves.

[11]The code which produced the presented results is available at `https://github.com/danielzahnd/master-thesis/tree/main/code/nf-milne-eddington-example-5-nflows-piecewisequadratic`.

In fig. 5.21, the learning curves for all three cases (training on one map, training on two maps, training on three maps) are visualized. The



(a) Learning curve for experiment 5, normalizing flow trained on one map.

(b) Learning curve for experiment 5, normalizing flow trained on two maps.

(c) Learning curve for experiment 5, normalizing flow trained on three maps.

Figure 5.21: Learning curves pertaining to experiment 5.

train and test losses appear to exhibit a desirable behaviour, with exception of the curves for training on two maps (fig. 5.21b). There, the train and test losses do converge to a stable value after several epochs, but remain at quite a large gap to one another, which is to be identified as a sign of possible overfitting. However, it might also just be the case, that the test split was underrepresentative in this case, leading to the only small improvement of the test loss over all epochs. Possibly, the learning curves for this case would improve, if the test split would encompass $1/5$ of the whole training dataset, rather than just $1/10$ as in the present case.

In fig. 5.23, one can find the results of this experiment for all three cases. As an overall observation it can be stated, that the smoothing property of the normalizing flow indeed does decrease with increasing size (and variation) of the training dataset. Take for example the line strength parameter η_0; for this parameter, the normalizing flow results clearly improve with increasing training dataset size as compared to the Milne-Eddington solution. The same can be observed for the magnetic field strength $|\boldsymbol{B}|$, inclination θ and damping a parameters, although not as strongly as for the line strength η_0. For the other atmospheric parameters, the normalizing flow performance does not seem to improve significantly. Concluding on the findings from this experiment so far it can be stated, that herewith credible evidence is provided for the earlier suggestion, that increasing the size and possibly variance of the training dataset is likely to improve the normalizing flow performance.

A further comment is to be made about the error development of the inversion results as depicted in fig. 5.24. What can be seen there is that the plots appear to become sharper from left to right. So, white areas tend to become whiter, and black areas tend to become darker as more data is included for training. This means, that for most areas on the maps

the obtained probability densities for the parameters become narrower as the training data amount and possibly variation increase, hence the uncertainty calculated as the standard deviations of those densities decrease. This observation corresponds with the expectations; with increasing size and variance in the training dataset, one expects the normalizing flow to learn to better constrain the possible solutions for atmospheric parameters, resulting in a narrowing of the probability densities of the parameters. There are however some areas, where the error appears to be increasing as the training data amount increases, for example the errors for the magnetic field strength $|\boldsymbol{B}|$ at and around umbra and pore structures in the map. A similar behaviour can also be detected with the other atmospheric parameters. This might be connected with the fact, that with increasing variance in the training dataset the normalizing flow could learn, that some parameter distribution $x \sim p_x(x)$ might be multimodal, hence giving a larger standard deviation than for an unimodal density. This reasoning is visualized in fig. 5.22. If this were true, this

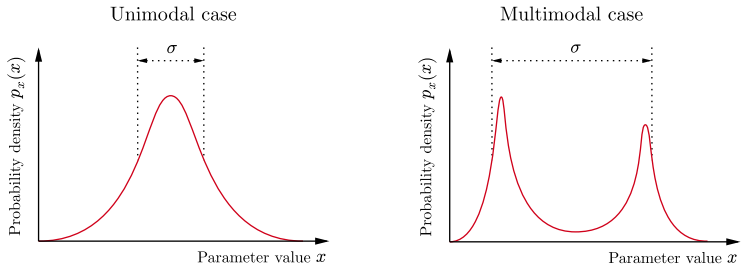


Figure 5.22: Visualization of the principle, that the standard variation σ of a multimodal probability density is larger than that of an unimodal one, even if the multiple modes are sharply peaked - as long as the multiple peaks are sufficiently separated.

should reflect for example in the error for the inclination values θ in and near umbras, since the magnetic field there can be expected to be more or less exactly either directed towards or away from the observer, considering the sunspot model as shown in fig. 1.3. Analyzing the error plots fig. 5.24, this truly seems to be the case; so this might be evidence for the suggested explanation.

It has been shown in the experiment underlying this section, that in the context of stellar atmosphere inversion increasing the variance and amount of data used for training indeed improves the performance of the trained normalizing flow on new data.

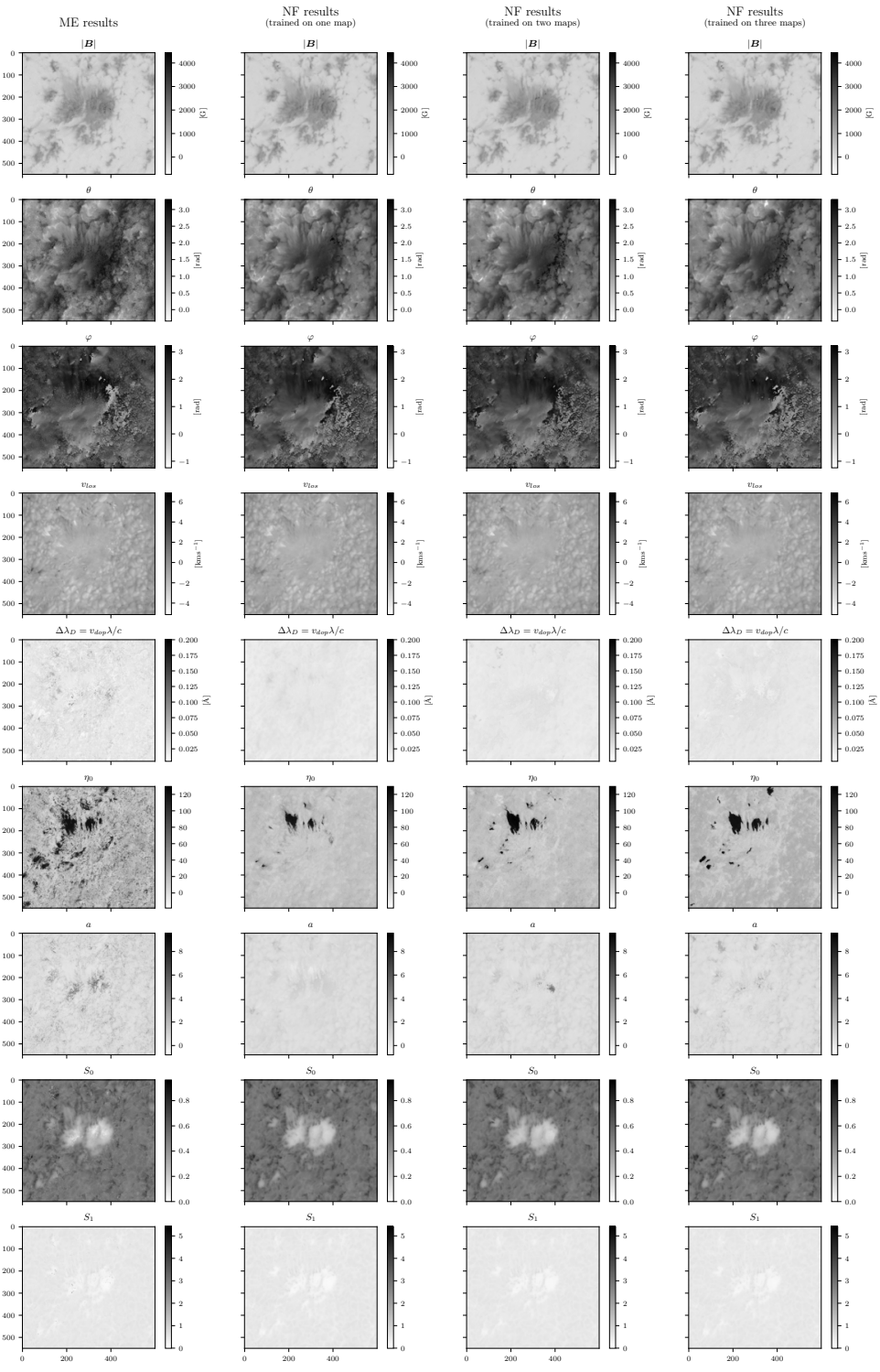


Figure 5.23: Inversion results as obtained from a normalizing flow trained on either one, two or three maps for experiment 5.

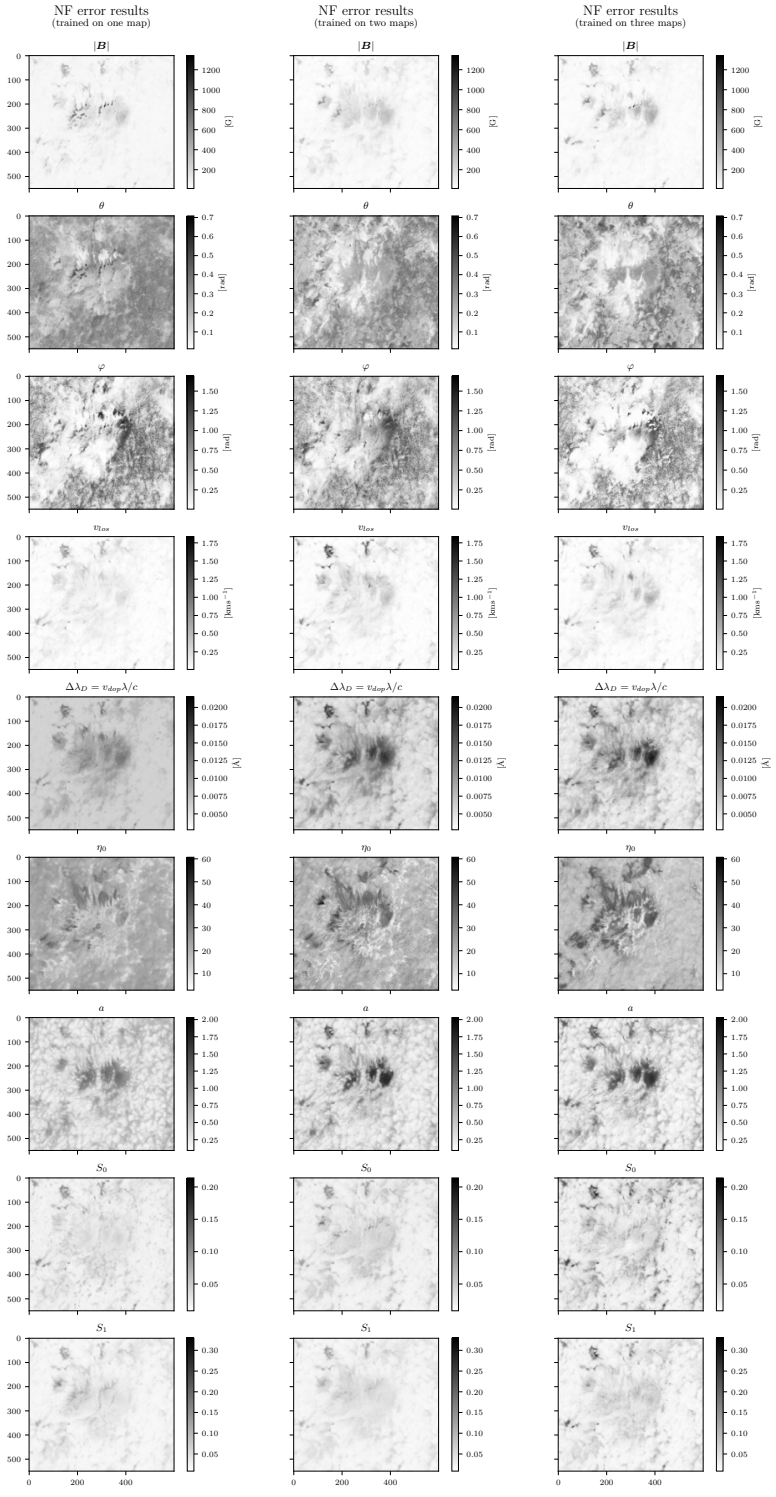


Figure 5.24: Inversion error results as obtained from a normalizing flow trained on either one, two or three maps for experiment 5.

5.6 Experiment 6: Tracking parameter evolution during penumbra formation

5.6.1 Explanation of the experiment

This final experiment for the thesis at hand is concerned with the evolution of the atmospheric parameters as inferred from inversion of all maps from the penumbra formation dataset. In this case, the observational data $\hat{\mathbf{y}}$ was inverted using the Milne-Eddington model, thus obtaining the corresponding atmospheric parameters $\hat{\mathbf{x}}$. The normalizing flow was then trained on this data, whereupon the trained flow was used to invert all observational data from subsequent frames as explained in section 5.3.1. This experiment particularly focused on the evolution of atmospheric parameters in the area of the red rectangle as given in fig. 5.25.

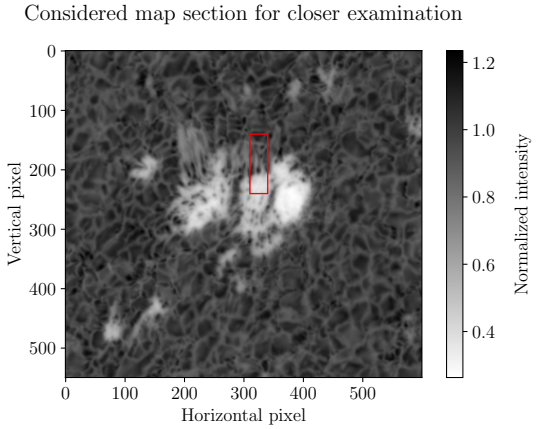


Figure 5.25: The area of particular interest for atmospheric parameter evolution during penumbra formation is indicated by the red rectangle. From frame 0 to frame 19 of the penumbra formation dataset, a penumbra can be observed to form in this region.

5.6.2 Results, discussion and interpretation of the experiment

The following results and figures were obtained[12] using a training sample size of 297000, a testing sample size of 33000, a batch size of 512, a hidden

[12]The code which produced the presented results is available at `https://github.com/danielzahnd/master-thesis/tree/main/code/nf-milne-eddington-example-7-nflows-piecewisequadratic`.

layer size of 16, a coupling layer amount of 5, a learning rate of 0.001, a scheduling rate of 0.999, 10 epochs to train the flow and randomized splitting of the data into train and test splits. This hyperparameter setting was found by manual inspection and improvement of resulting learning curves.

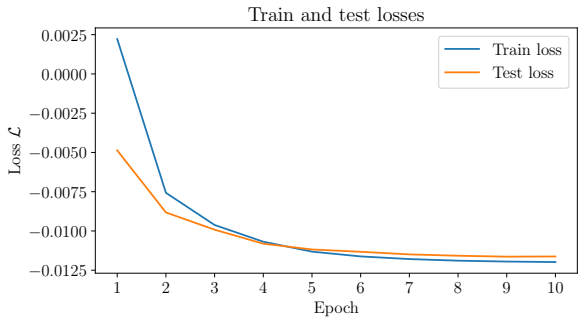


Figure 5.26: Learning curve for experiment 7 of normalizing flows applied to learn Milne-Eddington inversions.

The train and test loss curves as shown in fig. 5.26 show a desirable convergence behaviour. However, there remains a slight gap between train and test loss after 10 epochs. This gap might be reduced by increasing the variance and overall data amount in the training data and furthermore fine-tuning the hyperparameters. However, the normalizing flow model as used in this experiment appears to have an at least satisfactory convergence behaviour to gain some insight into the parameter evolution of the penumbra formation dataset.

The inversion and corresponding parameter evolution results obtained by the trained normalizing flow are shown in the two figures fig. 5.28 and fig. 5.29. In fig. 5.28, there are two columns; the first column shows the inversion results of the first frame of the penumbra formation dataset for each of the nine atmospheric parameters. The second column however shows the evolution of the atmospheric parameters within the slit of pixels indicated by the vertical red line in the plots of the first column, which is situated at the horizontal coordinate where the penumbra begins to form. The horizontal red lines are a visual aid to constrain the area of vertical coordinates, where the penumbra forms with evolving time. The vertical line of pixels between the two horizontal red lines shall be referred to as the considered vertical slit portion in the following comments. The magnetic field strength $|\boldsymbol{B}|$ appears to become more homogeneous within the considered vertical coordinates over time, the same seems to happen with the inclination parameter θ. Initially, there

are some areas where the inclination is about 3 rad in contrast to the more common inclination value of roughly 2 rad in considered vertical slit portion. With evolving time, the inclination seems to equilibrate to a value of roughly 2 rad, which is about 115°. This is to say, that the magnetic field at the end of the considered time window and in the considered vertical slit position has an almost homogeneous angle of 115° degrees with respect to line of sight, which matches the expectations considering the sketched penumbra formation model given in fig. 5.27, recalling that this means magnetic field lines exactly inverse to the direction as shown the figure. The azimuth parameter in the considered vertical slit portion

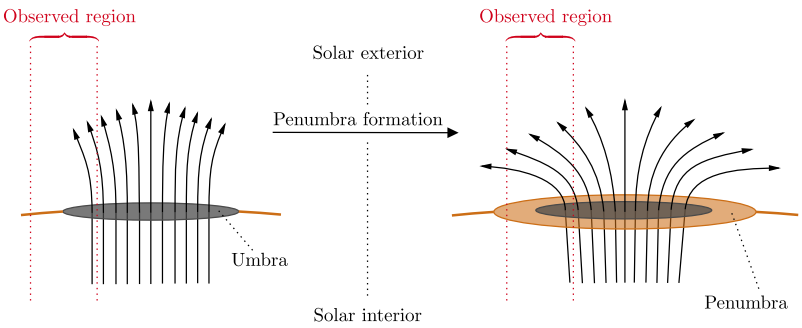


Figure 5.27: Schematic of what is expected to happen to the magnetic field lines during the penumbra formation process. Note, that the magnetic field lines as drawn here could also be oriented exactly opposite, i.e. facing towards the Sun's interior.

initially shows some pixels around vertical pixel 175 to have an azimuth of roughly 0.2 rad, while most other pixels have a azimuth value of up to 3 rad or more. Within the considered vertical slit portion therefore, the magnetic field has an azimuth value roughly in the range $\varphi \in [11°, 172°]$. Given that the azimuth angle φ is measured clockwise from solar north having $\varphi = 0$, this is also matching the expectations, since there is a penumbra-developing pore (umbra) located around horizontal pixel 390 and vertical pixel 250. Umbra and penumbra together form a sunspot; hence again recalling the sunspot model as given in fig. 1.3, one would expect the magnetic field to have an azimuth value in the second quadrant, given magnetic field lines pointing towards the Sun's interior, as it is the case. Corresponding to the second quadrant is the fourth quadrant, as the azimutal range only goes from 0 rad to π rad in theory. As can be seen from the plot, the expectation is hence indeed met by the results. Towards the end of the considered time window, the fluctuation in azimutal values for the magnetic field has equilibrated. The line of sight velocity v_{los} shows a spike in the considered vertical slit portion around

maps 5 to 8, indicating some plasma located around vertical pixel 150 is ejected towards the observer during that time, judging by the relative speeds as compared to other areas in the slit. This effect accordingly also reflects in the Doppler shift $\Delta\lambda_D$ of the observed Fe I spectral line. It is to be considered as at least interesting, that the spike in v_{los} occurs as soon as the azimuth φ quite rapidly changes from roughly 0.2 rad to around 2.8 rad at vertical pixels around 175 - exactly where the spike in v_{los} occurs. It might therefore be, that the spike in v_{los} is causally related or at least correlated to mentioned rapid change of φ. The evolution of the other atmospheric parameters are left uncommented, as the author cannot provide any conclusive explanations for the observed behaviour at present stage.

Moving on to an examination of fig. 5.29, an explanation of the plot is in order. The inversion results for the first frame of the penumbra formation dataset are again plotted in the first column, accompanied by a red rectangle. This red rectangle encompasses the area, where the penumbra is observed to form as time evolves. In order to track the parameter evolution in this area of interest, for each one of the nine atmospheric parameters the mean values for each row in the red rectangle was calculated. The two columns on the right of the plot hence show the results of these calculations. The values plotted in the far right column of fig. 5.29 were obtained by averaging over every row of beforehand obtained errors in the red rectangle. The above explained behaviour of the azimutal parameter φ, the line of sight velocity v_{los} and Doppler shift $\Delta\lambda_D$ is found confirmed. The initial disturbance in azimutal values around vertical pixel 156 to 175 however is associated to a quite large error of magnitude up to roughly 0.8 rad, therefore there might not even be a disturbance in truth, as it could be explained away by the large error.

Concerning the inclination θ, one observes that going out from the center of the umbra towards the penumbra (from higher to lower vertical pixel in the red rectangle), the inclination decreases from around $\theta \approx 2.4\,\text{rad} \approx 138°$ at the center of the umbra to $\theta \approx 1.6\,\text{rad} \approx 92°$ in the penumbra. With reference to fig. 5.27, this is exactly what one would expect the inclination to do for the considered red rectangle. Moreover, the evolution of this gradient seems to evolve such, that the inclination slightly decreases for the areas located more towards the penumbra than the umbra. This means, that the magnetic field at the boundary of an umbra (pore) initially at an angle of $\theta \approx \pi\,\text{rad}$ with respect to line of sight gets flatter with respect to the surface of the sun with evolving time,

as it is sketched in fig. 5.27. Eventually, if the magnetic field around the umbra gets flat enough with respect to the photospheric surface, the penumbra begins to form. It can be considered as a success, that the normalizing flow is apparently able to capture these effects.

Another interesting observation one can make about fig. 5.29 is that the magnetic field strength $|\boldsymbol{B}|$ tends to slightly decrease just at the boundary region of the umbra (pore) over time. Furthermore, the plots show the line strength parameter η_0 to get much stronger in areas, where the penumbra forms. At present time, the author however cannot provide any insightful explanation to these observations, aswell as to the evolution of the atmospheric parameters not discussed here.

In conclusion, normalizing flows have been shown to capture important features of penumbra formation physics, especially concerning magnetic field parameters. Herewith, additional evidence for the usefulness of normalizing flows for stellar atmospheric inversions is provided.

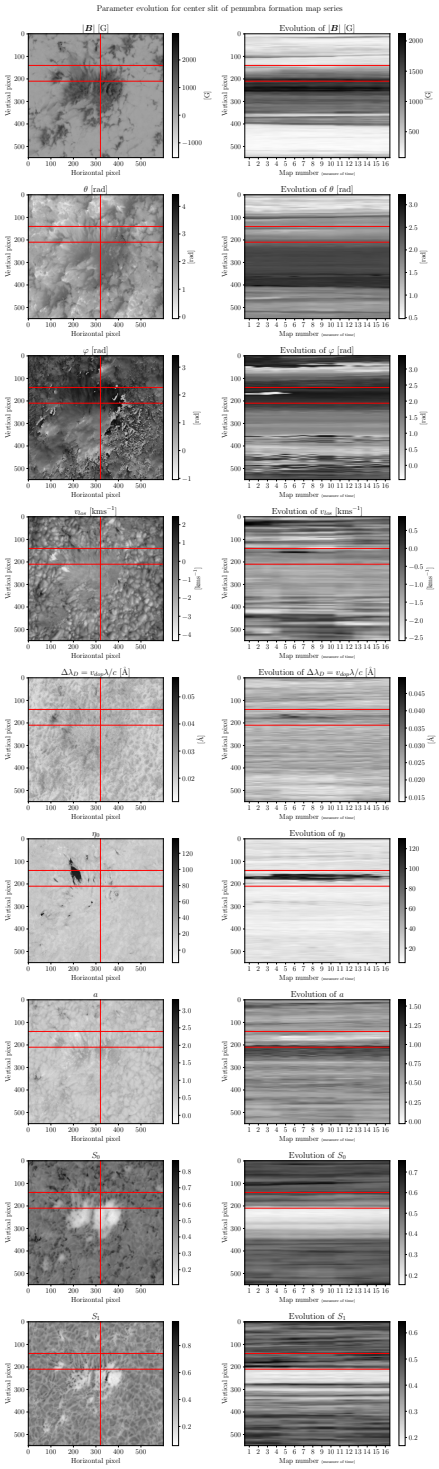


Figure 5.28: Inversion results concerning atmospheric parameter evolution in the penumbra formation dataset as obtained by a trained normalizing flow for experiment 6.

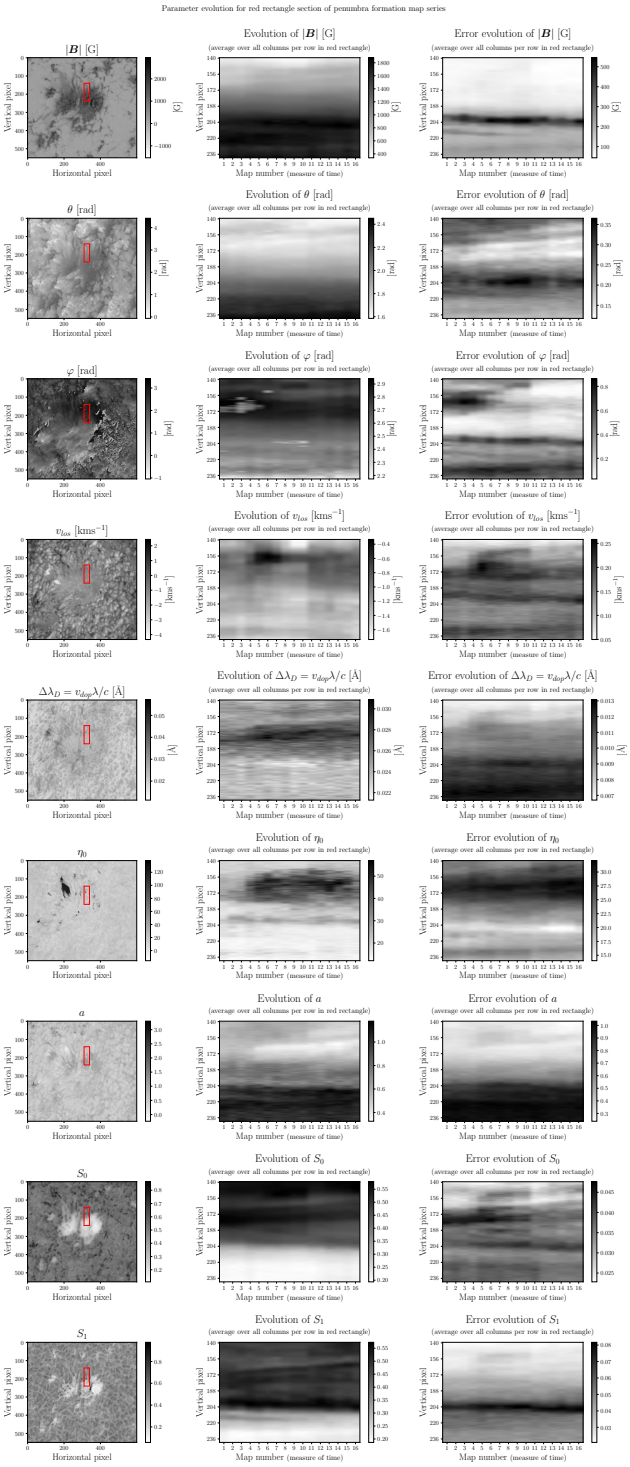


Figure 5.29: Inversion results concerning atmospheric parameter evolution and associated errors in the penumbra formation dataset as obtained by a trained normalizing flow for experiment 6.

Chapter 6

Conclusions

In this last chapter, the main conclusions from the beforehand discussed material shall be recalled, developed and presented.

Chapter 7

Appendix

7.1 Calculating a posterior probability density function

Suppose, one has some data $\boldsymbol{x} \in \mathbb{R}^d$ and corresponding data $\boldsymbol{y} \in \mathbb{R}^D$ connected to one another by a model $\boldsymbol{y} = \boldsymbol{M}(\boldsymbol{x})$, where $d, D \in \mathbb{N}$. Furthermore, assume that $\boldsymbol{x}$ and $\boldsymbol{y}$ follow the probability density functions $\boldsymbol{x} \sim p(\boldsymbol{x})$ and $\boldsymbol{y} \sim p(\boldsymbol{y})$. Given the Bayes theorem

$$p(\boldsymbol{x}|\boldsymbol{y}) = \frac{p(\boldsymbol{y}|\boldsymbol{x})p(\boldsymbol{x})}{p(\boldsymbol{y})} = \frac{p(\boldsymbol{y}|\boldsymbol{x})p(\boldsymbol{x})}{\int_{\mathbb{R}^d} p(\boldsymbol{y}|\boldsymbol{x})p(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}}, \tag{7.1}$$

one can in principle sample from the posterior density function of $\boldsymbol{x}$ conditioned on $\boldsymbol{y}$, if both the likelihood density function $p(\boldsymbol{y}|\boldsymbol{x})$ and the prior density function $p(\boldsymbol{x})$ are known.

If one knows $p(\boldsymbol{x})$ or makes a credible assumption for it, one can sample $L \in \mathbb{N}$ datapoints $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_L\}$ from it. For the likelihood density function, one oftentimes takes a Gaussian distribution with mean $\boldsymbol{M}(\boldsymbol{x})$, standard deviation σ and a diagonal covariance matrix as an Ansatz, namely

$$p(\boldsymbol{y}|\boldsymbol{x}) = \frac{1}{\sqrt{(2\pi\sigma^2)^D}} \exp\left(-\frac{\|\boldsymbol{y} - \boldsymbol{M}(\boldsymbol{x})\|^2}{2\sigma^2}\right). \tag{7.2}$$

Hereby, the components of $\boldsymbol{x}$ are assumed not being correlated with one another, which is the case with the components of $\boldsymbol{x}$ being independent parameters of the model $\boldsymbol{M}$. If there are $L \in \mathbb{N}$ samples $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_L\}$ taken from the prior $p(\boldsymbol{x})$, one can write the posterior probability density

$p(\boldsymbol{x}_k|\boldsymbol{y}_k)$ for any $k \in \{1, \ldots, L\}$ as

$$p(\boldsymbol{x}_k|\boldsymbol{y}_k) = \frac{p(\boldsymbol{y}_k|\boldsymbol{x}_k)p(\boldsymbol{x}_k)}{\sum_{j=1}^{L} p(\boldsymbol{y}_k|\boldsymbol{x}_j)p(\boldsymbol{x}_j)} = \frac{\exp\left(-\frac{\|\boldsymbol{y}_k - \boldsymbol{M}(\boldsymbol{x}_k)\|^2}{2\sigma^2}\right) p(\boldsymbol{x}_k)}{\sum_{j=1}^{L} \exp\left(-\frac{\|\boldsymbol{y}_k - \boldsymbol{M}(\boldsymbol{x}_j)\|^2}{2\sigma^2}\right) p(\boldsymbol{x}_j)}, \quad (7.3)$$

where the Bayes theorem and the law of total probability were used. For $L \to \infty$, this leads to the exact posterior probability density $p(\boldsymbol{x}|\boldsymbol{y})$. Since the denominator in the above formula is just a normalizing constant, one can write

$$p(\boldsymbol{x}|\boldsymbol{y}) \propto p(\boldsymbol{y}|\boldsymbol{x})p(\boldsymbol{x}) = \exp\left(-\frac{\|\boldsymbol{y} - \boldsymbol{M}(\boldsymbol{x})\|^2}{2\sigma^2}\right) p(\boldsymbol{x}). \quad (7.4)$$

Taking the natural logarithm yields the log-likelihood as

$$\log\left[p(\boldsymbol{x}|\boldsymbol{y})\right] \propto \log\left[p(\boldsymbol{y}|\boldsymbol{x})\right] + \log\left[p(\boldsymbol{x})\right] = -\frac{\|\boldsymbol{y} - \boldsymbol{M}(\boldsymbol{x})\|^2}{2\sigma^2} + \log\left[p(\boldsymbol{x})\right]. \quad (7.5)$$

Assuming a prior $p(\boldsymbol{x})$ of the form

$$p(\boldsymbol{x}) = \begin{cases} W^{-d}, & \boldsymbol{x}_{min} \leq \boldsymbol{x} \leq \boldsymbol{x}_{max} \\ 0, & \text{otherwise} \end{cases} \propto \begin{cases} 1, & \boldsymbol{x}_{min} \leq \boldsymbol{x} \leq \boldsymbol{x}_{max} \\ 0, & \text{otherwise} \end{cases}, \quad (7.6)$$

with $\|\boldsymbol{x}_{min} - \boldsymbol{x}_{max}\| = W^d$ for a constant $1 < W \in \mathbb{R}$ satisfying the normalizing condition $1 = \int_{\mathbb{R}^d} p(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x}$, one can write

$$\log\left[p(\boldsymbol{x})\right] = \begin{cases} -d\log(W), & \boldsymbol{x}_{min} \leq \boldsymbol{x} \leq \boldsymbol{x}_{max} \\ -\infty, & \text{otherwise} \end{cases}. \quad (7.7)$$

Using $\log(1) = 0$ and $\log(0) = -\infty$ one can establish the proportionality

$$\log\left[p(\boldsymbol{x})\right] \propto \begin{cases} 0, & \boldsymbol{x}_{min} \leq \boldsymbol{x} \leq \boldsymbol{x}_{max} \\ -\infty, & \text{otherwise} \end{cases} \quad (7.8)$$

and thus

$$\log\left[p(\boldsymbol{x}|\boldsymbol{y})\right] \propto -\frac{\|\boldsymbol{y} - \boldsymbol{M}(\boldsymbol{x})\|^2}{2\sigma^2} + \begin{cases} 0, & \boldsymbol{x}_{min} \leq \boldsymbol{x} \leq \boldsymbol{x}_{max} \\ -\infty, & \text{otherwise} \end{cases}. \quad (7.9)$$

The sampling process needed to obtain a posterior density close to the true one, Markov Chain Monte Carlo (MCMC) methods are usually applied.

Acknowledgments

I would like to express my sincere gratitude to Prof. Dr. Lucia Kleint for accepting me as a master's student and allowing me to write a thesis on the fascinating topics of solar physics and machine learning. Through the process of researching and writing the master's thesis, as well as her continuous support, guidance, and encouragement, I have learned a great deal. I would also like to sincerely thank Jonas Zbinden for the many valuable ideas he contributed to my master's thesis, for answering my questions and for preparing many of the datasets I used in my thesis. Additionally, I want to express my gratitude to my parents, who initially provided financial support and later, most importantly, emotional support throughout my extended educational journey.

However, mighty thanks are reserved for my loving wife, Regula. Without her continuous encouragement during the master's thesis and, more broadly, throughout my studies and life, this work would not have been possible and I would never be in such a privileged stage of life as I am now. Through her loving support, I not only found the energy and strength to persevere in my studies but, more importantly, I was able to grow and mature personally, aswell as a couple together with her.

Bibliography

[Amini, 2023] Amini, A. (2023). *Introduction to Deep Learning - Lecture notes*. Massachusetts Institute of Technology.

[de La Cruz Rodríguez, 2019] de La Cruz Rodríguez, J. (2019). A method for global inversion of multi-resolution solar data. *Astronomy & Astrophysics*, 631:A153.

[Degl'Innocenti, 2005] Degl'Innocenti, E. L. (2005). *Polarization in Spectral Lines*, volume 307 of *Springer eBook Collection Physics and Astronomy*. Springer Netherlands, Dordrecht.

[del Toro Iniesta, 2003] del Toro Iniesta, J. C. (2003). *Introduction to Spectropolarimetry*. Cambridge University Press, Cambridge.

[Díaz Baso et al., 2022] Díaz Baso, C. J., Asensio Ramos, A., and de La Cruz Rodríguez, J. (2022). Bayesian stokes inversion with normalizing flows. *Astronomy & Astrophysics*, 659:A165.

[Durkan et al., 2019] Durkan, C., Bekasov, A., Murray, I., and Papamakarios, G. (2019). Neural spline flows.

[Durkan et al., 2020] Durkan, C., Bekasov, A., Murray, I., and Papamakarios, G. (2020). nflows: normalizing flows in PyTorch.

[Goodfellow et al., 2016] Goodfellow, I., Courville, A., and Bengio, Y. (2016). *Deep learning*. Adaptive computation and machine learning. The MIT Press, Cambridge, Massachusetts.

[Kobyzev et al., 2021] Kobyzev, I., Prince, S. J. D., and Brubaker, M. A. (2021). Normalizing flows: An introduction and review of current methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(11):3964–3979.

[Mitchell, 1997] Mitchell, T. M. (1997). *Machine learning*. McGraw-Hill international editions. McGraw-Hill, New York, NY, international ed. edition.

[Raschka, 2022] Raschka, S. (2022). *Machine Learning with PyTorch and Scikit-Learn: Develop machine learning and deep learning models with Python*. Packt Publishing Limited, Birmingham, 1 edition.

[Rutten, 2015] Rutten, R. J. (2015). *Introduction to astrophysical radiative transfer: Lecture notes*.

[Stix, 2002] Stix, M. (2002). *The Sun: An Introduction*. Astronomy and Astrophysics Library. Springer, Berlin and Heidelberg, second edition edition.

[Uitenbroek, 2020] Uitenbroek, H. (2020). *Polarized Radiative Transfer: Milne-Eddington Inversions Workshop - Lecture notes*. National Solar Observatory.

[Weigert et al., 2006] Weigert, A., Wendker, H. J., and Wisotzki, L. (2006). *Astronomie und Astrophysik: Ein Grundkurs*. Lehrbuch Physik. Wiley-VCH Verlag, Weinheim, 1. nachdr. der 4., völlig überarb. und erw. aufl. edition.

[Weng, 2018] Weng, L. (2018). Flow-based deep generative models. *lilianweng.github.io*.