



universität
wien

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

„Reconstructing the Cosmic Web from the Lyman-alpha Forest with Machine Learning / Rekonstruktion des Kosmischen Netzes anhand von Messungen des Lyman-Alpha-Waldes mit Maschinellem Lernen“

verfasst von / submitted by

Branko Mitic

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of
Master of Science (MSc)

Wien, 2022 / Vienna 2022

Studienkennzahl lt. Studienblatt /
degree programme code as it appears on
the student record sheet:

UA 066 861

Studienrichtung lt. Studienblatt /
degree programme as it appears on
the student record sheet:

Astronomie / Astronomy

Betreut von / Supervisor:

Univ.-Prof. Dipl.-Phys. Dr. sc. ETH Oliver Hahn

Mitbetreut von / Co-Supervisor:

B.Sc. M.Sc. PhD, Florian List

I declare that this master's thesis was written entirely by me and that I have not used any publications that are not mentioned in the text.

Vienna, October 2022

.....
(Branko Mitic)

RECONSTRUCTING THE COSMIC WEB FROM THE LYMAN-ALPHA FOREST WITH MACHINE LEARNING

Lyman-alpha transitions in neutral hydrogen clouds cause characteristic spectral features in observations of very distant quasars and galaxies. Density fluctuation estimates can be made for each line of sight in redshift space. Combining multiple lines of sight enables three-dimensional reconstruction of the underlying field and therefore the mapping of the universe at high redshifts, contributing to the understanding of large-scale structures, but also small scale clustering, dark matter properties and the shape of the primordial spectrum of fluctuation from inflation. A common method used exactly for this purpose is the Wiener Filter, an analytic approach which minimizes the expected variance of the error for Gaussian random fields. As datasets become very large, however, solving the Wiener Filter equation becomes increasingly difficult requiring large amounts of computational power. Furthermore, the assumption of a Gaussian prior neglects the non-linear nature of structure formation. In order to contribute to further development of Lyman-alpha forest tomography, I successfully applied two new methods for three-dimensional density field reconstruction one of which is based on a conditional generative adversarial network and the other on compressed sensing, a signal processing technique. Both of these techniques yield results improving upon the Wiener Filter, especially on smaller scales identifying nodes, filaments and sheets with greater accuracy while concurrently reproducing Lyman-alpha statistics such as power spectrum and probability density function to a significantly higher grade at lower computational expense.

Lyman-Alpha-Übergänge in neutralen Wasserstoffwolken verursachen charakteristische Merkmale in Spektren sehr weit entfernter Quasaren und Galaxien. Für jede Sichtlinie können Dichteschwankungen im Rotverschiebungsraum abgeschätzt werden. Die Kombination mehrerer Sichtlinien ermöglicht eine dreidimensionale Rekonstruktion des zugrunde liegenden Feldes und damit die Kartierung des Universums bei hohen Rotverschiebungen, was zum Verständnis großräumiger Strukturen, aber auch kleinräumiger Clusterbildung, Eigenschaften dunkler Materie und der Form des ursprünglichen Schwankungsspektrums beiträgt. Eine gängige Methode, die genau für diesen Zweck verwendet wird, ist der Wiener Filter, ein analytischer Ansatz, der die erwartete Varianz des Fehlers für Gauß'sche Felder minimiert. Dadurch, dass Datensätze sehr groß werden, wird das Lösen der Wiener Filter-Gleichung immer schwieriger und erfordert große Mengen an Rechenleistung. Darüber hinaus vernachlässigt die Annahme eines Gauß'schen Priors die nichtlineare Natur der Strukturbildung. Um zur Weiterentwicklung der Lyman-Alpha-Wald-Tomographie beizutragen, habe ich erfolgreich zwei neue Methoden zur dreidimensionalen Dichtefeldrekonstruktion angewendet, von denen eine auf einem Neuronalen Netzwerk und die andere auf Compressed Sensing, einer Signalverarbeitungstechnik basiert. Beide Techniken liefern Ergebnisse, die den Wiener Filter mit weniger Rechenaufwand verbessern; insbesondere auf kleinen Skalen können dadurch die Komponenten des kosmischen Netzes mit größerer Genauigkeit identifiziert werden, während gleichzeitig Lyman-alpha-Wald Statistiken wie Leistungsspektrum und Wahrscheinlichkeitsdichtefunktion deutlich genauer reproduziert werden.

Contents

1. Introduction	1
1.1. Notation	2
1.2. The history of the universe in five stages	3
1.3. Inhomogeneities in the universe	5
1.3.1. Perturbation growth in Gaussian density fields	5
1.3.2. Non-linear structure evolution	8
1.4. Lyman-alpha forest	12
1.4.1. Fluctuating Gunn-Peterson approximation	12
1.4.2. Surveys	13
2. Reconstruction Methods	15
2.1. Notation	16
2.2. Wiener filter	17
2.3. Optimized reconstruction with constraints on absorption	18
2.4. Compressed sensing	19
2.5. Machine Learning	23
2.5.1. Neural networks, training and activation functions	23
2.5.2. Conditional generative adversarial networks	25
2.5.3. Network architecture	26
3. Results	29
3.1. Wiener filter parameters	30
3.2. 2D early and late universe Lyman-alpha flux reconstructions	32
3.2.1. Residual standard deviation	33
3.2.2. Power- and cross spectra	33
3.2.3. Probability density functions	34
3.3. 3D late universe Lyman-alpha flux and density reconstructions	38
3.3.1. Power- and cross spectra	41
3.3.2. Probability density functions	41
3.3.3. Structure classification	42
3.3.4. Computational cost analysis	44
3.3.5. Summary	45
4. Discussion and Future Perspectives	47
List of Figures	51
Bibliography	53
Appendix	59
A. Convex Sets and Functions	59
B. Conditions for Convex Optimization	60
C. Python implementations	61

1. Introduction

Chapter 1 contains a brief introduction to the history of the universe, cosmic large-scale structure formation, to the Lyman-alpha forest and the history of its discovery and the fluctuating Gunn-Peterson approximation. Two important surveys that have taken place in the past couple of years are also included. Chapter 2 provides a description on traditional and novel three-dimensional field reconstruction methods including the Wiener filter, Optimized Reconstruction with Constraints on Absorption (ORCA), compressed sensing and an approach using a generative neural network. Chapter 3 finally reveals the numerous interesting results discovered during the course of this thesis. The results and future perspectives are discussed in chapter 4. Very specific details including full code implementations are contained in the appendix.

1.1. Notation

Before proceeding with the description of perturbation growth in Gaussian density fields, it's useful to introduce the most important notation which is kept consistent, unless clearly stated otherwise, throughout this entire chapter.

Quantity	Description
z	Redshift
t	Time
V	Volume
$\mathbf{x}_i$	Points in Eulerian coordinates
r	Distance between two points $\mathbf{x}_1$ and $\mathbf{x}_2$
ρ	Density
ρ_0	Initial density
$\bar{\rho}$	Mean cosmic matter density at a time t
$\delta, \tilde{\delta}$	Relative density contrast and its Fourier transform
δ^{ic}	Primordial density fluctuation
ξ_N	N point correlation function
$\mathbf{k}, k$	Wave vector, wave number
D_+	Growth factor
P	Power spectrum
n_s	Power law exponent of initial power spectrum
A	Empirically determined normalization constant of the primordial power spectrum
T	Transfer function
$\mathbf{x}$	Point in Eulerian coordinates
$\mathbf{q}$	Point in Lagrangian coordinates
$\mathbf{s}$	Potential vector field
ϕ	Displacement potential
J	Jacobian matrix
$\alpha_{1,2,3}$	Eigenvalues of the deformation tensor
τ	Optical depth
κ, β	Empirically determined parameters for FGPA
S_0	Classical action of motion in Zeldovich approximation
K_0	Free propagator
ψ_0	Initial wave function
ϕ^{ic}	Back scaled gravitational potential
Ξ	Displacement field between Lagrangian and Eulerian coordinates
Ξ_0	Initial displacement field
$\mathbf{K}_{\text{RSD}}$	Propagator from Eulerian to redshift space
$\hbar$	Free parameter controlling effective smoothing scale of a field
F	Flux
χ_0	Wave function, mapping optical depth to redshift space.
$D_{ij}, \tilde{D}_{ij}$	Deformation tensor and its Fourier transform

1.2. The history of the universe in five stages

The entire overview in section 1.2 is written based on a collection of literature [Dod03, Muk05, MVdBW10, Ryd17, DEFW85], for further details please refer to these references.

Stage one: Big Bang and Planck epoch

The firmly established model of how the universe came to be is the *hot big bang*, which was first suspected by Gamow and later observationally confirmed¹ by Penzias and Wilson in 1965 [PW65]. Shortly after the big bang, the four fundamental forces starting with gravitation followed by electromagnetic, weak and strong interactions were established. In the grand unified model it is assumed that the latter three fundamental forces were merged into a single force at these very early times when the universe was extremely hot and dense. This phase is called the *Planck epoch*. Small quantum fluctuations in this phase are thought to be the foundation of large-scale structure formation much later. The universe started to expand rapidly during the period of *inflation*. At this point particles and anti-particles were formed in almost equal amounts, annihilating immediately; however, a small amount of matter survived.

Stage two: Quark-hadron transition and BBN

One microsecond after the big bang protons and neutrons were formed, and from around 0.01 seconds to around three minutes the big bang Nucleosynthesis (BBN) took place. In this phase, light elements such as deuterium, helium-4, and lithium were synthesized followed by a temperature drop, making nuclear fusion no longer possible. At these times the energy density of radiation exceeded the energy density of matter.

Stage three: Recombination and decoupling

In the early stages of the universe, matter and photons were strongly coupled through Compton scattering. At around 380 000 years post big bang, neutral hydrogen formed, resulting in the previously opaque universe becoming transparent. This phase is called *recombination*. Hydrogen atoms typically form with electrons in an excited state since direct recombinations to the ground state of hydrogen are inefficient. After the atom is formed however, the electrons drop to the ground state releasing a photon. These photons can still be observed today as the *cosmic microwave background*.

Stage four: Reionization and the dark ages

Following recombination, it took several hundred million years following the big bang for hydrogen clouds to collapse slowly to form stars and galaxies, making photons released during decoupling the only sources of electromagnetic radiation during this phase. Probing these times from recombination to reionization is done with the 21 cm line which can be observed at radio frequencies on Earth. The Square Kilometre Array is planned to make great advances in this regard [RAB⁺04]. With galaxy clusters and superclusters forming, UV radiation from the first stars, galaxies and quasars ionizes the intergalactic medium. This phase is called the reionization period. In 2022 the James Webb Telescope conducted an observation of the earliest object ever observed (apart from CMB) called CEERS-93316 at redshift $z = 16.7$ corresponding to 235.8 million years of cosmic time [DMD⁺22].

¹Penzias and Wilson were the first to detect the cosmic microwave background which is a necessary remnant in the hot big bang model.

Stage five: Dark energy domination

Around five billion years ago an inflection point occurred when the universe entered the *dark energy dominated era*. This inflection point characterizes the change in the universe's expansion rate from deceleration to acceleration. The reason for this lies within the evolution of the Hubble parameter, following from the Friedmann equations from which it becomes evident that at some point dark energy will necessarily dominate over time.

In the following sections I am going to provide a concise description of how the small density fluctuations from the early universe could develop into the large-scale structures, i.e. nodes, filaments, sheets and voids we see today.

1.3. Inhomogeneities in the universe

The entire subsection 1.3.1 contains information collected from [Sch06, The21, Bra85, GP82, Zel72, Pee81, PY70, Har70, Muk05]. The standard model of cosmology is based on the cosmological principle, which assumes spatial homogeneity and isotropy of the universe on large-scales. It's not only interesting but also important to think about exactly how large these scales have to be for the cosmological principle to apply. Three-dimensional galaxy surveys have failed to find structures larger than around $100h^{-1}$ Mpc, meaning that the universe appears to be homogeneous averaged over scales of $R \sim 200h^{-1}$ Mpc [Sch06]. Comparing this homogeneity scale with the Hubble radius $R_H \equiv c/H_0 \approx 3000h^{-1}$ Mpc we can see that $R \ll R_H$ which is why the universe can be approximated as isotropic. The small density fluctuations from the early universe have grown to their present day extent as a consequence from dark matter falling into potential wells without being pushed back by pressure. Furthermore, with self-gravity, overdense regions could accumulate more matter over time while underdense regions do the opposite. In other words, the gravitational instability lead to an increase of density fluctuations over time [Sch06].

1.3.1. Perturbation growth in Gaussian density fields

The fluctuations of the cosmic microwave background imply a Gaussian density field for early times in the evolution of the Universe on large scales which is why Gaussian density field statistics is assumed for those times and many inflation models predict this too [Bra85, GP82].

The two-point correlation function is method for characterizing structures of the cosmic matter distribution and can be defined as follows [The21],

$$\begin{aligned} \langle \rho(\mathbf{x}_1), \rho(\mathbf{x}_2) \rangle &= \bar{\rho}^2(t) \langle [1 + \delta(\mathbf{x}_1)][1 + \delta(\mathbf{x}_2)] \rangle \\ &= \bar{\rho}^2(t) (1 + \langle \delta(\mathbf{x}_1)\delta(\mathbf{x}_2) \rangle) \\ &=: \bar{\rho}^2(t) [1 + \xi_2(\mathbf{x}_1, \mathbf{x}_2)], \end{aligned} \quad (1.1)$$

where $\delta(\mathbf{x}, t)$ is the relative density contrast and the density is

$$\rho(\mathbf{x}, t) = \bar{\rho}(t) (1 + \delta(\mathbf{x}, t)), \quad (1.2)$$

and $\bar{\rho}(t)$ denotes the mean cosmic matter density at time t . ξ can only depend on the distance of $\mathbf{x}_1$ and $\mathbf{x}_2$, $r = |\mathbf{x}_1 - \mathbf{x}_2|$ because the Universe is isotropic. The angled brackets in equation (1.1) denote an ensemble average of distributions that have the same statistical properties. The infinite set of correlation functions,

$$\begin{aligned} \langle \delta(\mathbf{x}_1)\delta(\mathbf{x}_2) \rangle &= \xi_2 \\ \langle \delta(\mathbf{x}_1)\delta(\mathbf{x}_2)\delta(\mathbf{x}_3) \rangle &= \xi_3 \\ \langle \delta(\mathbf{x}_1)\delta(\mathbf{x}_2)\delta(\mathbf{x}_3)\delta(\mathbf{x}_4) \rangle &= \xi_4 \\ \langle \delta(\mathbf{x}_1)\delta(\mathbf{x}_2)\delta(\mathbf{x}_3) \cdots \delta(\mathbf{x}_N) \rangle &= \xi_N \end{aligned} \quad (1.3)$$

determines the statistical properties of the density field. Higher order correlators for Gaussian random fields, however, are determined fully by ξ_2 [Muk05]. Characterizing the density fields by their Fourier transforms gives further insight into the numerous interesting features of perturbation growth. The matter distribution of a universe can, for instance, also be described by the power spectrum, an equivalent method closely related to the correlation function. A density field can be expressed as

$$\delta(\mathbf{x}, t) = \sum \delta(\mathbf{k}, t) \exp(i(\mathbf{k} \cdot \mathbf{x})) = \frac{V}{(2\pi)^3} \int \delta(\mathbf{k}, t) \exp(i(\mathbf{k} \cdot \mathbf{x})) d\mathbf{k}, \quad (1.4)$$

here, V is a large volume in which the Universe is approximated as periodic [The21]. The power spectrum is defined as $P(k) \equiv \langle |\delta(\mathbf{k})|^2 \rangle$ and the relationship between the correlation function and the power spectrum is given by a Fourier transform,

$$P(k) = 2\pi \int_0^\infty dx x^2 \frac{\sin(kr)}{kr} \xi(r). \quad (1.5)$$

Generally, the power spectrum does not fully describe the statistical properties of a random field similar to the way in which the correlation function $\xi(r)$ doesn't. Gaussian random fields on the contrary can be fully characterized by the power spectrum and the probability distribution of the density fluctuations at any location is Gaussian. This is important because at very early times in the universe a large number of inflation models predict a Gaussian density field [Bra85, GP82]. As the evolution of large-scale structures progresses over time, the density field gradually shifts from a Gaussian to a non-Gaussian regime.

Henceforth, the time dependence of $P(k)$ and $\xi(r)$ is going to be displayed explicitly. Assuming that $\mathbf{r}$ is the comoving separation vector and

$$\delta(\mathbf{r}, t) = D_+(t) \delta_0(\mathbf{r}), \quad (1.6)$$

where $D_+(t)$ is the growth factor and $\delta_0(r)$ the linearly extrapolated density fluctuation field², then

$$\xi(r, t) = D_+^2(t) \xi(r, t_0), \quad (1.7)$$

and the time-dependent power spectrum becomes

$$P(k, t) = D_+^2(t) P(k, t_0) =: D_+^2(t) P_0(k). \quad (1.8)$$

Here, k is a comoving wave number [Sch06]. It is important to note that these relations are only valid in the framework of Newtonian, linear perturbation theory in the matter dominated era of the Universe.

²This refers to the distribution of density fluctuations at present times, assuming the evolution is linear at all times.

The primordial power spectrum

Many versions of the inflationary scenario for the very early universe produce a power-law form determining the initial fluctuation spectrum which was imprinted on the universe at some early time [Gut81, GP82].

$$P(k) \propto k^{n_s}. \quad (1.9)$$

More than 30 years ago, Harrison, Zeldovich and Peebles argued that $n_s = 1$, because this is the slope for which the amplitude of the fluctuations of the gravitational potential is constant, meaning that neither large nor small scales are preferred³. This type of spectrum is also called a scale-invariant spectrum or *Harrison-Zeldovich spectrum* [Zel72, Pee81, PY70, Har70]. Using such a spectrum to write the power spectrum at a time t_i after the inflationary epoch yields,

$$P(k, t_i) = D_+^2(t_i) A k^{n_s}. \quad (1.10)$$

Here, A is a normalization constant determined from temperature fluctuation measurements in the CMB [SBK⁺92]. Because the growth of the amplitude of density fluctuations is different in the radiation-dominated phase, the result in equation (1.10) can be further modified to improve the shape of the power spectrum, in terms of its applicability to regimes with a radiation dominated phase, for instance, with the introduction of a transfer function T ,

$$P(k, t) = D_+^2(t) A k^{n_s} T^2(k). \quad (1.11)$$

The transfer function, however, heavily depends on the dark matter model. Assuming the matter content of the universe is known, it is possible to calculate T for any cosmological model [Sch06].

Cold dark matter and hot dark matter

An important role played in these modifications is the nature of dark matter where there are two distinctions being made, *cold dark matter* (CDM) on one hand and *hot dark matter* (HDM) on the other, the difference between the two being the characteristic velocities of their components [Muk05, DEFW85]. Since hot dark matter has no observational basis and doesn't produce structure formation that agrees with observations, the focus in this section is going to be on CDM.

³A power law is a function without a characteristic scale.

1.3.2. Non-linear structure evolution

Linear perturbation theory cannot be applied for the evolution of structures like clusters, which is where new perturbation techniques come into play. It's important to cover some conceptual differences between Lagrangian and Eulerian perturbation theory for the sake of completeness. The section regarding Eulerian Perturbation follows the work done in [Fry84, JMW95, MSS92, SS91, GGRW86, Vis83] and the section about Lagrangian Perturbation theory is based on [BCHJ94] and [BCGS02]. For more information, please refer to the aforementioned literature. The main difference between the two approaches, however, can be summarized as follows. Eulerian perturbation theory focuses on density and velocity fields of matter for a fixed coordinate system. Lagrangian Perturbation Theory on the other hand describes individual particle trajectories. In Λ cold dark matter (Λ CDM) cosmology, structure formation occurs through baryon and dark matter collapse which compared to velocities introduced by gravitational forces is approximated as cold. Lagrangian perturbation theory is used to simulate the growth and dynamics of cosmological structures by directly describing the motion of fluid elements before they cross. Therefore, Lagrangian perturbation theory is only valid for large-scales or early times and the same applies for Eulerian perturbation theory. The only dynamical variable in Lagrangian Perturbation Theory is the Lagrangian displacement field.

Zel'dovich approximation

The Zel'dovich approximation is the first order term in the solution of Lagrangian Perturbation Theory for the displacement field and it relates the initial Lagrangian coordinates with Eulerian coordinates at a time t [Zel70]. This section is closely going to follow the descriptions in [HSVdW13, Jeo10, HR21]. The aforementioned relation can be written as follows,

$$\mathbf{x}(\mathbf{q}, D_+(t)) = \mathbf{q} + D_+(t)\mathbf{\Xi}_0(\mathbf{q}), \quad (1.12)$$

where $\mathbf{q}$ is the Lagrangian coordinate and the Eulerian coordinate is denoted as $\mathbf{x}$. The potential vector field $\mathbf{\Xi}_0(\mathbf{q}) = -\nabla_{\mathbf{q}}\phi(\mathbf{q})$ is the initial displacement, determined by the initial density perturbations and $D_+(t)$ is the linear growth factor. ϕ is the displacement potential. Taking into account the conservation of mass, the density evolution is,

$$\rho(\mathbf{x}, t) = \bar{\rho}(t) J \left[\frac{\partial \mathbf{x}}{\partial \mathbf{q}} \right]^{-1}, \quad (1.13)$$

The deformation of mass elements is specified by the Jacobian. By finding the eigenvectors of the Jacobian matrix and diagonalizing it in the basis of the calculated eigensystems as,

$$J(\mathbf{q}, t) = \begin{vmatrix} 1 + \alpha_1 D_+(t) & 0 & 0 \\ 0 & 1 + \alpha_2 D_+(t) & 0 \\ 0 & 0 & 1 + \alpha_3 D_+(t) \end{vmatrix}, \quad (1.14)$$

where $\alpha_{1,2,3}$ are the eigenvalues with which the density contrast can be written as

$$\delta(\mathbf{x}, t) = \frac{1}{(1 + \alpha_1 D_+(t))(1 + \alpha_2 D_+(t))(1 + \alpha_3 D_+(t))} - 1. \quad (1.15)$$

One of the big advantages is that the Zeldovich approximation predicts the first shell-crossing, or in other words, when the density becomes infinite as a consequence of crossing particle trajectories. At this point it becomes clear that locally $\delta \rightarrow \infty$ in the case where at least one of the three bracketed terms in the denominator becomes zero. Since the eigenvalues have a non-Gaussian distribution [Dor70] and δ is a non-linear function of $\alpha_{1,2,3}$, it also becomes apparent that Gaussian properties of early universe density fields are gradually reduced which isn't taken into account by using the Wiener filter for density field reconstruction purposes in the context of large-scale structures.

Propagator perturbation theory

In [PHJL20] the density and flux fields in redshift space are generated using Propagator Perturbation Theory, which is a semi classical approach describing cold collisionless dynamics. It is an alternative for Lagrangian Perturbation Theory translating it into an action⁴ and utilizing a propagator⁵ for wave function evolution. The evolved wave function provides the Eulerian density and velocity field [PHJL20]. The datasets employed in this thesis consist of Lyman-alpha-like structure simulations over the course of the time-line of the universe ranging from early to late times in two and three dimensions. The purpose of including early times is to elaborate the limits of Wiener filtering statistically, or in other words, to show how WF reconstruction quality is optimal for early Gaussian random fields and then gradually starts to decrease with increasing degrees of non-Gaussianity. The three-dimensional mock Lyman-alpha surveys are utilized for evaluations of reconstruction quality in potential real world usage. The datasets are created using propagator perturbation theory which is a semi-classical LPT alternative describing what happens between initial fluctuations and observation of spectra. The entire section on propagator perturbation theory closely follows the findings in [URGH19, PHJL20], as a matter of fact, the exact code used to generate the two and three-dimensional density and flux fields in [PHJL20] is used in this thesis to generate the Lyman-alpha forest. Since modelling time evolution of matter density is not the primary focus of my thesis, this section serves only as a brief overview and for detailed information please refer to the aforementioned reference.

Background

The classical action of a motion in the Zeldovich approximation where a fluid element moves in a time D_+ from its initial Lagrangian coordinate $\mathbf{q}$ to its final Eulerian coordinate $\mathbf{x}$ on a straight line is

$$S_0(\mathbf{x}, \mathbf{q}; t) = \frac{1}{2} \frac{(\mathbf{x} - \mathbf{q})^2}{D_+(t)}. \quad (1.16)$$

Step one

Use the Dirac-Feynman trick [Fey48, Dir05] to promote the action to a propagator K_0 ,

$$K_0(\mathbf{x}, \mathbf{q}; t) = (2\pi i \hbar D_+(t))^{-3/2} \exp \left[\frac{i}{\hbar} S_0(\mathbf{x}, \mathbf{q}; t) \right]. \quad (1.17)$$

Step two

Compute ψ_0 , the wave function that represents the transition amplitude from an initial state to the final state,

$$\psi(\mathbf{x}, t) = \int d^3q K_0(\mathbf{x}, \mathbf{q}; t) \psi_0(\mathbf{q}). \quad (1.18)$$

In this context, $\hbar$ is regarded as a free parameter controlling the effective smoothing scale [PHJL20].

Step three

Formulate a wave function that only has phase perturbations,

$$\psi_0(\mathbf{q}) := \exp \left[-\frac{i}{\hbar} \phi^{\text{ic}}(\mathbf{q}) \right], \quad (1.19)$$

where ϕ^{ic} is the gravitational potential at the initial time. The uniform mean density is $\rho_0 := \psi_0 \bar{\psi}_0 = 1$. In the previous sentence the complex conjugates are marked with overlines. In this context, the primordial density fluctuations are denoted as $\delta^{\text{ic}} := \nabla^2 \phi^{\text{ic}}$.

⁴An action is a value which describes how a system changes over time.

⁵A propagator describes the probability amplitude for a particle to go from one place to another place.

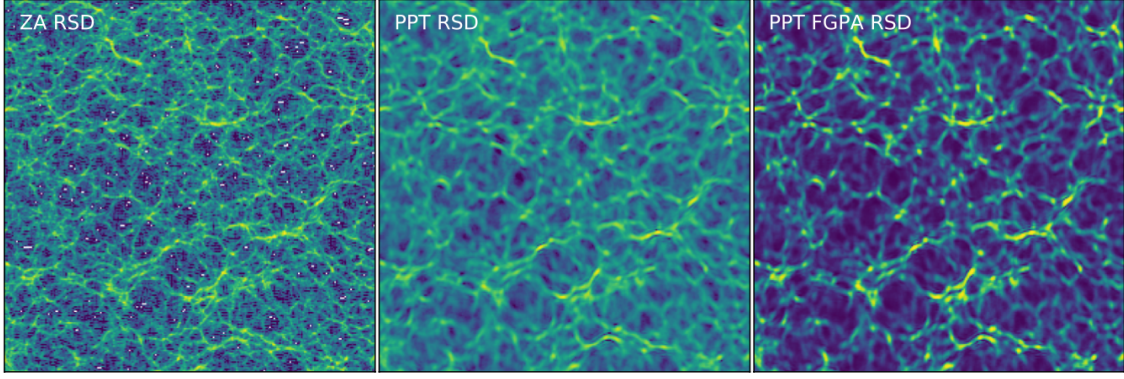


Figure 1.1.: Comparison: Zeldovich and PPT density fields in redshift space on the left, the image on the far right is the flux field generated by the code used in [PHJL20]. The yellow pixels indicate areas of high density and low flux while the blue pixels indicate areas of high flux and low density. This image is taken from [PHJL20].

Step four

Plugging K_0 and ψ_0 into equation (1.18) constitutes a convolution, allowing the integral to be solved with a Fourier transform,

$$\psi(\mathbf{x}, t) = \text{DFT}^{-1} \left[\exp \left(-i\hbar \frac{k^2}{2} D_+(t) \right) \text{DFT} \left[\exp \left(-\frac{i}{\hbar} \phi^{ic}(\mathbf{q}) \right) \right] \right]. \quad (1.20)$$

This result is achieved through exploitation of circular convolution on a periodic domain and describes the full relationship between ϕ^{ic} and ψ . The discrete values $\mathbf{q}$ and $\mathbf{x}$ in equation (1.20), are positioned on a three dimensional grid.

Step five

Use the propagated wave function in the following way,

$$\rho = \psi \bar{\psi}, \quad (1.21)$$

to get the normalized density $\rho := 1 + \delta$ where δ is the fractional overdensity. This results in a coinciding solution with a smoothed Zeldovich approximation. The benefit of this approach is that the field can be calculated in redshift space.

Step six

Include the deviations from pure Hubble expansion given by,

$$\mathbf{s} := \mathbf{x} + f(t)(\mathbf{\Xi} \cdot \hat{\mathbf{e}}_{\text{LOS}})\hat{\mathbf{e}}_{\text{LOS}}, \quad (1.22)$$

where $\mathbf{s}$ is the redshift space position⁶, $\hat{\mathbf{e}}_{\text{LOS}}$ is a unit vector along the line of sight and $\mathbf{\Xi} := \mathbf{x} - \mathbf{q}$ is the displacement field between Lagrangian and Eulerian coordinates, into an additional propagator from Eulerian to redshift space,

$$K_{\text{RSD}}(\mathbf{s}, \mathbf{x}; t) = Z \exp \left[\frac{i}{\hbar} \frac{1}{2} \frac{((\mathbf{s} - \mathbf{x}) \cdot \hat{\mathbf{e}}_{\text{LOS}})^2}{f(t)D_+(t)} \right] \quad (1.23)$$

where Z is an appropriately chosen normalization. It makes sense to calculate fields in redshift space because that's where observations take place.

⁶With the approximation of a distant observer where line of sight skewers are parallel making the redshift space distortions coincide with a Cartesian axis

Step seven

Combine the propagators into a single one,

$$\widehat{\mathbf{K}}(\mathbf{k}; t) := \exp \left[-\frac{i\hbar}{2} \left(\mathbf{k}^2 + f(t) (\mathbf{k} \cdot \hat{\mathbf{e}}_{\text{LOS}})^2 \right) D_+(t) \right] \quad (1.24)$$

Step eight

Calculate the fractionally transmitted flux from the optical depth τ with

$$F = e^{-\tau}, \quad (1.25)$$

where τ , in Eulerian space, is given by

$$\tau(\mathbf{x}) := \kappa \rho(\mathbf{x})^\beta. \quad (1.26)$$

Here κ and β are heuristic parameters determined by the physical state of IGM.

Step nine

Build a wave function to map the optical depth to redshift space conserving τ ⁷,

$$\chi_0(\mathbf{x}) := \kappa^{1/2} \rho^{(\beta-1)/2}(\mathbf{x}) \psi(\mathbf{x}). \quad (1.27)$$

Subsequently propagate this wave function to redshift space while conserving τ ,

$$\chi(\mathbf{s}) := \int d^3x \mathbf{K}_{\text{RSD}}(\mathbf{s}, \mathbf{x}; t) \chi_0(\mathbf{x}). \quad (1.28)$$

Step ten

Finally, calculate the three-dimensional quasar flux field F in redshift space with,

$$F(\mathbf{s}) = \exp(-\chi \bar{\chi}). \quad (1.29)$$

Each flux field used in the evaluation covers a volume of $64h^{-1} \text{ Mpc} \times 64h^{-1} \text{ Mpc} \times 64h^{-1} \text{ Mpc}$ corresponding to a box size of 64^3 pixels. The number of line of sights is chosen in a way that yields an average perpendicular line of sight distance of $2.5h^{-1} \text{ Mpc}$ and quantity of data along a skewer is chosen to yield an average distance along a line of sight between data points of around $2h^{-1} \text{ Mpc}$ both in line with [LKW⁺18b, HLA⁺21, LHC21]. No extra noise is added to the data. In total, around 8% of data within the volume is known from which the complete flux field reconstructions are made in redshift space.

⁷The optical depth has to be conserved during the mapping from physical to redshift space [Sel12].

1.4. Lyman-alpha forest

The entire section 1.4 uses information from [GP65, CO17, BFW⁺01, CWKH98, Lyn71, BG71, HNY97, SIK⁺13, LKW⁺18b, HLA⁺21]. The Lyman-alpha forest traces the neutral hydrogen density [GP65] and appears in spectra of very distant quasars and galaxies. Visually it is characterized by an unusually high number of absorption lines, caused by the Lyman alpha electron transition in neutral hydrogen clouds at high redshifts. This transition occurs when a photon with a wavelength of 1216 ångström collides with an electron in the ground state of a hydrogen atom, moving it to the first excited state [CO17]. When this electron returns from the higher level back to the ground state it releases a photon with the same energy that originally propelled it to the higher energy state. As the newly released photon is travelling towards the spectrometer, it gets redshifted along the way making the absorption lines appear at wavelengths in the spectrum corresponding to the redshifts where neutral hydrogen clouds absorbed or scattered the photons. The number of absorption lines in the Lyman-alpha forest increases with redshift until around redshift six [BFW⁺01]. At this point there is so much neutral hydrogen in the intergalactic medium that the forest turns into a trough. This feature is called the Gunn-Peterson trough [GP65] and shows the end of the reionization of the universe.

The discovery

The Lyman-alpha forest was discovered in 1970 during an observation of the quasar 4C 05.34, the farthest ever observed object at the time ($z = 2.877$), by astronomer Roger Lynds [Lyn71]. Noticing the unexpectedly high quantity of absorption lines in the spectrum shown in Figure 1.2, he correctly suggested that most of the absorption lines are caused by the same Lyman-alpha transition occurring at different redshifts. Additionally Lynds is also credited with coining the term "Lyman-alpha forest".

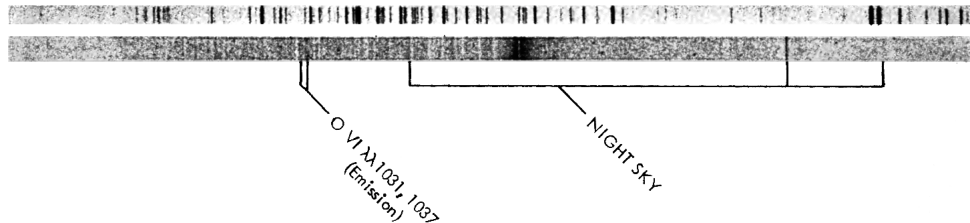


Figure 1.2.: Spectrogram showing the many absorption lines in 4C 05.34. The comparison spectrum is He+Ar+Ne. The section labelled night sky contains features caused by airglow. Additionally Lynds also labelled the O VI emission. O VI is an important tracer for growth of metals in stellar nucleosynthesis. The Lyman-alpha line is located in the center. This figure is taken from the original Lyman-alpha forest paper by Lynds [Lyn71].

Further observations by researchers John Bahcall and Samuel Goldsmith substantiated the presence of unusually numerous absorption lines in 4C 05.34 and later observations of other high-redshift quasars and galaxies continued to display Lyman-alpha forest features [BG71].

1.4.1. Fluctuating Gunn-Peterson approximation

The Fluctuating Gunn-Peterson Approximation (FGPA) [GP65] is an analytic approximation relating the matter overdensity and optical depth according to a power-law with the assumption of an equilibrium between optically thin photoionization and collisional recombination of residual neutral hydrogen in the intergalactic medium, $\tau \propto (1 + \delta)^\beta$, where the slope of the power law β is determined by the physical state of the IGM and is related to the slope of the temperature-density relationship. Simulations have shown that with the FGPA it is possible to reproduce the power spectrum and flux probability density function [HNY97].

1.4.2. Surveys

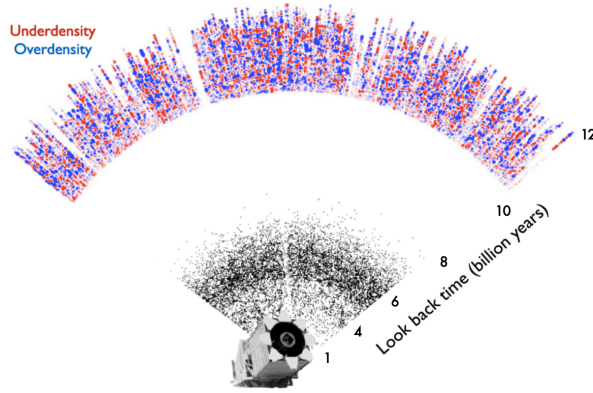


Figure 1.3.: BOSS survey visualization of the under- and overdense regions. The outer blue and red ring marks the Lyman-alpha forest where the sharp cut-off is caused by limits of the spectrograph. This figure is taken from [FR18].

BOSS and CLAMATO surveys

The Baryon Oscillation Spectroscopic survey [SIK⁺13] was designed to measure the expansion rate of the universe by mapping spatial distributions of luminous red galaxies and quasars to determine the characteristic scale which is imprinted by baryon acoustic oscillations in the early universe. From Lyman-alpha forest measurements, it was possible to measure the three-dimensional structure of the universe. These kinds of surveys can be used to investigate large-scale structures, but also small-scale clustering, dark matter properties and the shape of primordial spectrum of fluctuation from inflation. Figure 1.3 shows a visualization of the under- and overdense regions. The outer blue and red ring marks the Lyman-alpha forest where the sharp cut-off is caused by limits of the spectrograph. The COSMOS Lyman-Alpha Mapping And Tomography Observations (CLAMATO) [LKW⁺18b, HLA⁺21] is a survey from 2016 and was conducted with the LRIS spectrograph on the Keck I telescope. In total it observed 240 galaxies and quasars with redshifts between $z = 2.17$ and $z = 3$ probing the foreground Lyman-alpha forest absorption between $z = 2.05$ and $z = 2.55$ over an observational footprint of only 0.157 degrees squared. Reduction finally yielded spectra from 437 objects. Figure 1.4 shows a three-dimensional CLAMATO Lyman-alpha forest map. The red areas correspond to high densities and the blue one correspond to low densities. The redshift of a protocluster and an overdensity are also marked.

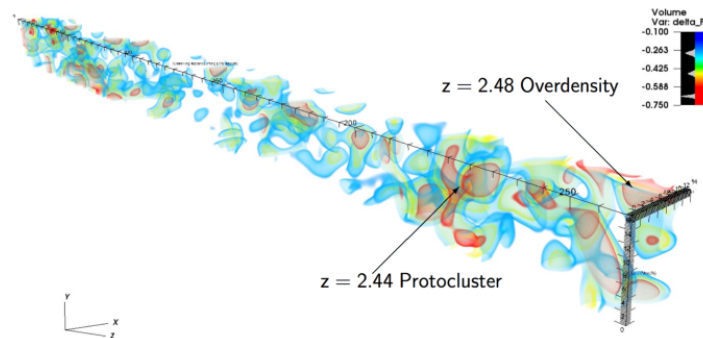


Figure 1.4.: Three-dimensional CLAMATO Lyman-alpha forest tomography. The red areas correspond to high densities and the blue one correspond to low densities. The redshift of a protocluster and an overdensity are also marked. This figures is taken from [LKW⁺18a].

2. Reconstruction Methods

The goal is to reconstruct the three-dimensional underlying flux field in redshift space, given data located only on the line of sight skewers penetrating the field between the location of observation and the high redshift quasar or galaxy. Figure 2.1 is a visual representation of the 3D reconstruction objective going from a massively down-sampled set of data points to a fully reconstructed field.

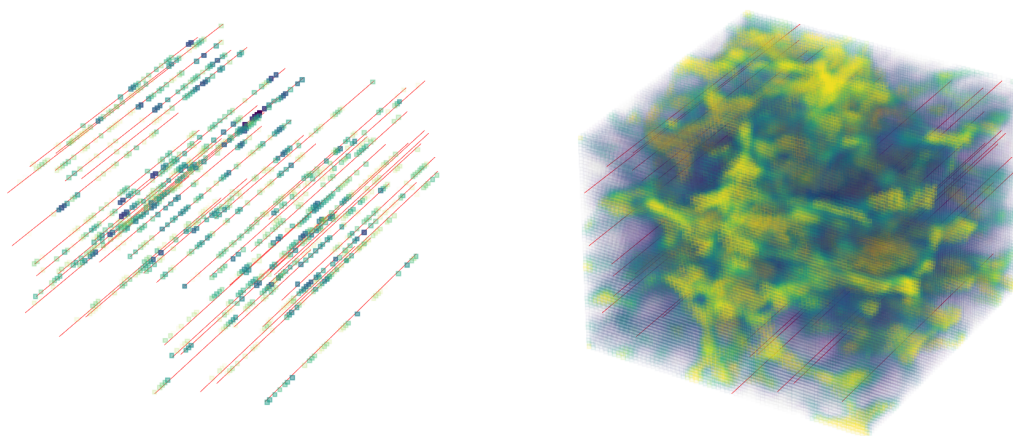


Figure 2.1.: Visual representation of the reconstruction objective. The figure on the right shows the "true" simulated density field generated with the methods described in chapter 1. The figure on the left shows the massively down-sampled field from which the reconstruction is to be made. The red lines mark the line of sight skewers and in the down-sampled field, the green, blue and yellow pixels represent measurement data with yellow colors corresponding to a higher density, i.e. low flux and the blue colors corresponding to lower densities or high flux.

Mathematically, the process of going from the down-sampled measurements on the left to the full reconstruction on the right Figure 2.1 can be expressed as follows,

$$\mathbf{d} = \mathbf{M}\mathbf{s}, \quad (2.1)$$

where $\mathbf{d}$ is the result of the measurement matrix $\mathbf{M}$ down-sampling the true field $\mathbf{s}$. This type of problem is also called an undetermined inverse problem. In such a case, direct inversion isn't possible since there is an infinite number of solutions $\mathbf{s}$ that can satisfy equation (2.1). The purpose of the following sections is to provide an overview of the different methods for solving the problem presented in equation (2.1) starting with the Wiener filter (WF), followed by optimized reconstruction with constraints on absorption (ORCA) and the two novel methods for Lyman-alpha field reconstructions, compressed sensing and machine learning. The appendix section A goes into detail on convexity, an important requirement for the formulation of an appropriate loss function for our problem, appendix section B describes when a solution is considered to be optimal. Appendix section C contains the python implementation codes.

2.1. Notation

Before proceeding with the description of methods, it's useful to introduce the most important notation which is kept consistent throughout this entire chapter.

Quantity	Description
N	Number of given data points
M	Number of total pixels within the reconstruction volume
$\mathbf{d}$	Given data, $\in \mathbb{R}^N$
$\mathbf{M}$	Measurement matrix sub-sampling the true field, $\in \mathbb{R}^{N \times M}$
$\mathbf{s}$	Reconstruction map, solution of equation (2.1), $\in \mathbb{R}^M$
$\mathbf{d}_s$	Measured signal within a data point, $\in \mathbb{R}^N$
$\mathbf{n}$	Noise $\in \mathbb{R}^N$, uncorrelated with itself and uncorrelated with the signal
$\mathbf{L}$	The matrix that when applied to $\mathbf{d}$ yields $\mathbf{s}$
ϵ	Minimal expected error
$\mathbf{S}_{MN}$	Map-data covariance matrix
$\mathbf{S}_{NN}$	Data-data covariance matrix
σ^2	amplitude of Gaussian covariance matrix
$\Delta r_{\parallel}$	Distance between two data points along a line of sight
$\Delta r_{\perp}$	Distance between two data points perpendicular to a line of sight
$L_{\parallel}$	Parallel covariance length
$L_{\perp}$	Perpendicular covariance length
$\mathbf{G}_{smoothing}$	Gaussian smoothing operator
$\mathbf{N}$	Diagonal covariance matrix with identical elements
α	Empirically determined upper limit in the second ORCA clipping term
$k_{1,2,3}$	Weighting parameters of ORCA loss function terms
λ	Regularizer weight
Ψ	Universal orthogonal basis transformation matrix
$\tilde{\mathbf{s}}$	Reconstruction map in an alternative basis, $\tilde{\mathbf{s}} = \Psi^T \mathbf{s}$
η	Step size in gradient descent
$\mathbf{z}_n$	Random noise vector, input for cGAN
G	Generator in the neural network
D	Discriminator in the neural network
$\mathcal{L}_{cGAN}$	cGAN objective

2.2. Wiener filter

The Wiener Filter provides an analytic way of solving equation (2.1) with the addition of uncorrelated noise. Assuming a Gaussian prior model which determines the covariance matrix form, the Wiener Filter provides a minimal error solution. In such a case it can be shown that the Wiener filter matches the Bayesian estimator which maximizes the posterior probability. For the full derivation and details, please refer to [ZHFL94]. The derivation in this thesis closely follows the one in [SWLH15]. Assuming the subsampled available data $\mathbf{d}$ consists of the signal $\mathbf{d}_s$ and noise¹ $\mathbf{n}$,

$$\mathbf{d} = \mathbf{d}_s + \mathbf{n}, \quad (2.2)$$

The objective is to calculate a linear estimate of the signal, $\mathbf{s} = \mathbf{L}\mathbf{d}$, where the expected error is minimal.

$$\begin{aligned} \epsilon &= E[|\mathbf{d}_s - \mathbf{s}|^2] \\ &= \text{tr}(E[\mathbf{d}_s \mathbf{d}_s^T] - E[\mathbf{d}_s \mathbf{s}] - E[\mathbf{s} \mathbf{d}_s^T] + E[\mathbf{s} \mathbf{s}^T]), \end{aligned} \quad (2.3)$$

Simplifying each term further while keeping in mind that noise is assumed to be uncorrelated with itself and uncorrelated with the signal,

$$\begin{aligned} E[\mathbf{s}_s \mathbf{s}^T] &= E[\mathbf{s}_s (\mathbf{L}\mathbf{d})^T] = E[\mathbf{s}_s \mathbf{d}^T \mathbf{L}^T] \\ &= E[\mathbf{s}_s (\mathbf{d}_s^T + \mathbf{n}^T) \mathbf{L}^T] \\ &= (E[\mathbf{s}_s \mathbf{d}_s^T] + E[\mathbf{s}_s \mathbf{n}^T]) \mathbf{L}^T = \mathbf{S}_{MN} \mathbf{L}^T \\ E[\mathbf{s} \mathbf{s}_m^T] &= E[(\mathbf{L}\mathbf{d}) \mathbf{d}_s^T] \\ &= E[\mathbf{L}(\mathbf{d}_s + \mathbf{n}) \mathbf{d}_s^T] \\ &= \mathbf{L}(E[\mathbf{d}_s \mathbf{d}_s^T] + E[\mathbf{n} \mathbf{d}_s^T]) = \mathbf{L} \mathbf{S}_{pm} \\ E[\mathbf{s} \mathbf{s}^T] &= \mathbf{L} E[\mathbf{d} \mathbf{d}^T] \mathbf{L}^T \\ &= \mathbf{L}(E[\mathbf{d}_s \mathbf{d}_s^T] + E[\mathbf{d}_s \mathbf{n}^T] + E[\mathbf{n} \mathbf{d}_s^T] + E[\mathbf{n} \mathbf{n}^T]) \mathbf{L}^T \\ &= \mathbf{L}(\mathbf{S}_{NN} + \mathbf{N}) \mathbf{L}^T \end{aligned} \quad (2.4)$$

$\mathbf{S}_{NN}$ and $\mathbf{S}_{MN}$ are the data-data and map-data covariance matrices. The error becomes,

$$\begin{aligned} \epsilon &= \text{tr} \mathbf{S}_{MM} - \text{tr}(\mathbf{S}_{pm} \mathbf{L}^T) - \text{tr}(\mathbf{L} \mathbf{S}_{MN}) + \text{tr}(\mathbf{L}(\mathbf{S}_{NN} + \mathbf{N}) \mathbf{L}^T) \\ &= \text{tr} \mathbf{S}_{MM} - 2 \text{tr}(\mathbf{L} \mathbf{S}_{MN}) + \text{tr}(\mathbf{L}(\mathbf{S}_{NN} + \mathbf{N}) \mathbf{L}^T), \end{aligned} \quad (2.5)$$

and the derivative of the error is,

$$\frac{\partial \epsilon}{\partial \mathbf{L}} = -2 \mathbf{S}_{MN} + 2(\mathbf{S}_{NN} + \mathbf{N})^T \mathbf{L}^T. \quad (2.6)$$

Setting this to zero and rearranging gives the Wiener Filter equation used in this master's thesis as formulated by [SWLH15],

$$\mathbf{s} = \mathbf{L}\mathbf{d} = \mathbf{S}_{MN}(\mathbf{S}_{NN} + \mathbf{N})^{-1} \mathbf{d}. \quad (2.7)$$

Here, the covariance matrices $\mathbf{S}_{NN}$ and $\mathbf{S}_{MN}$ denote the pixel-pixel and map-pixel covariance matrices which are both of Gaussian form,

$$\mathbf{S}_{NN} = \mathbf{S}_{MN} = \sigma_A^2 \exp \left[-\frac{(\Delta r_{\parallel})^2}{2L_{\parallel}^2} \right] \exp \left[-\frac{(\Delta r_{\perp})^2}{2L_{\perp}^2} \right], \quad (2.8)$$

where σ_A^2 is the amplitude and $L_{\perp}^2$ and $L_{\parallel}^2$ are the perpendicular and parallel covariance lengths. My two and three dimensional Wiener filter Python implementations are in appendix section C. For three-dimensional reconstructions I used Dachshund [Sta16], a Wiener filter implementation iteratively solving the linear system of equations in equation (2.7).

¹This reconstruction method requires a noise term in order to function. The noise is uncorrelated and equal for each line of sight, the latter property is not necessary.

2.3. Optimized reconstruction with constraints on absorption

Optimized Reconstruction with Constraints on Absorption (ORCA) [LHC21] is a flux reconstruction method developed to avoid intensive computation by minimizing a loss function. For this purpose, the authors of [LHC21] employ the limited memory Broyden Fletcher Goldfarb Shanno algorithm (L-BFGS)².

The Loss function to be minimised is

$$\begin{aligned} \mathcal{L} = & k_1(\mathbf{G}_{smoothing}(\mathbf{s}) - \mathbf{s})^T(\mathbf{G}_{smoothing}(\mathbf{s}) - \mathbf{s}) + (\mathbf{M}\mathbf{s} - \mathbf{d})^T\mathbf{N}^{-1}(\mathbf{M}\mathbf{s} - \mathbf{d}) \\ & + k_2 \sum clip(\mathbf{s}, 1, +\infty) + k_3 \sum clip(\mathbf{s}, 0, \alpha), \end{aligned} \quad (2.9)$$

where $\mathbf{G}_{smoothing}(\mathbf{s})$ is a Gaussian smoothing operator smoothing the output field $\mathbf{s}$ and $\mathbf{d}$ is the given, sub-sampled data. In my implementation, $\mathbf{M}\mathbf{s}$ is sub-sampling the field $\mathbf{s}$ to the measurement locations and $\mathbf{N}$ is the diagonal noise covariance matrix with identical entries [LHC21]. Similarly to the Wiener Filter, this reconstruction method also requires a noise term in order to function. In this thesis, I use the same uncorrelated noise for each skewer which is why the noise covariance matrix ends up being diagonal with identical entries, an extension to non-constant noise, however, is straightforward. The purpose of the clipping terms is to prevent results in an invalid flux range and keeping values from going too far above α with the second one having the effect of accounting for corrections in the probability density function. According to [LHC21], α is an empirical value, depending on the mean flux and smoothing scale. The idea behind this loss function is to reconstruct the data within the given locations with the second term as good as possible while concurrently smoothing the field with the first term. The weighting of all terms can be adjusted through modifications of $k_{1,2,3}$. This is just a brief overview of the ORCA method, for more information please refer to [LHC21].

Since the results in my thesis do not perfectly coincide with the results in [LHC21] it is unclear whether my implementation matches the one in [LHC21]. In [LHC21] the authors speak of an annealing procedure in the Gaussian smoothing term, which my version does not contain. However, the results achieved with my implementation are still very good, especially in terms of computational expense. Henceforth, mentioning ORCA implies my ORCA-based implementation and not exactly what is used in [LHC21]. The reason for including this reconstruction method is solely as a comparison to the new ones.

²The L-BFGS algorithm is an iterative quasi-Newton method for solving unconstrained nonlinear optimization problems [Fle13,BLNZ95,LHC21]. In a nutshell, the descent direction is determined through gradient preconditioning with curvature information. The way in which this is achieved is through gradual improvements of the loss function's Hessian matrix approximation, only by evaluating gradients with a generalized secant method [DJS96, LN89].

2.4. Compressed sensing

The sections about compressed sensing, Lasso regression and iterative minimization techniques is primarily based on [HK70, Tib96, BK22, Tib15, FR13, Rud16, BT09]. Compressed sensing is a method for reconstructing high-dimensional signals $\mathbf{s}$ from massively downsampled measurements $\mathbf{d}$ where the solution in a certain basis $\tilde{\mathbf{s}}$ is sparse, i.e. has many zero or close to zero entries, so the goal is to solve,

$$\begin{aligned}\mathbf{d} &= \mathbf{M}\mathbf{s} \\ &= \mathbf{M}\Psi\tilde{\mathbf{s}},\end{aligned}\tag{2.10}$$

for $\mathbf{s}$, where $\mathbf{s} = \Psi\tilde{\mathbf{s}}$ and $\mathbf{M}$ is a mask matrix, describing the measurement process and Ψ represents a universal orthogonal basis [BK22, FR13]. The system of equations in equation (2.10) is an underdetermined inverse problem, meaning that there is an infinite number of solutions for $\mathbf{s}$ that could solve it, however, there is only one sparsest solution in an alternative basis, which is the one that is desired. The whole idea behind this is that real images are naturally sparse in certain basis [BK22]. Structure evolution in the early phases of the universe started from a Gaussian field, which is maximally non-sparse in, for instance, Fourier space. As time progressed and large-scale structures, such as the ones found in the Lyman-alpha-forest started to evolve, sparsity in the Fourier space also increased and suddenly the idea of using compressed sensing to decipher an increasingly sparse field in Fourier becomes attractive. This idea is not new, [LSLP16, RLJ⁺22] have used algorithms based on wavelet sparsity priors for gravitational lensing applications. The linear regression technique for finding sparse solutions to equation (2.10) is called *Lasso regression* [Tib96].

Lasso regression

Lasso regression [Tib96] is a type of linear regression used for solving linear problems like the one in equation (2.1). Finding a linear relationship between $\mathbf{M}$ and $\mathbf{d}$ so that $\mathbf{d} = \mathbf{M}\mathbf{s}$, can, for instance, be achieved by minimizing [Tib96],

$$\frac{1}{N}\|\mathbf{M}\mathbf{s} - \mathbf{d}\|_2^2 + \lambda\|\mathbf{s}\|_1.\tag{2.11}$$

The first smooth term is the basic least-squares regression and the non-differentiable term scaled by a factor λ is called a *regularizer*. The purpose of a regularizer is to modify the magnitude of the solution components. The regularizer in equation (2.11) is the l_1 -norm. The reason why the l_1 -regularized regression yields sparse solutions becomes clear for the two dimensional case as seen in Figure 2.2 where the left figure shows the diamond-shaped contours of the constant l_1 -norm, the right figure shows the circular contours of the l_2 -norm. The solutions of the first term are located along the solid line. Evidently, the l_1 minimization gives a sparser solution because one of the elements of $\mathbf{s}$ is zero. This also applies for higher dimensions [Tib96, UGS18]. At first glance there is a minor problem with Lasso if a minimization technique is used that assumes differentiability which is that $\|\mathbf{s}\|_1$ is not differentiable so subgradient methods are employed [Tib15].

Gradient methods for minimization

Before describing the subgradient method used for minimizing the Lasso loss, it's important to understand gradient descent first. Gradient descent [Rud16] initializes a $\mathbf{s}_0$ and updates $\mathbf{s}$ according to,

$$\mathbf{s}_{t+1} = \mathbf{s}_t - \eta\nabla f(\mathbf{s}_t),\tag{2.12}$$

where η is the step size, t is the iteration number. This updating algorithm causes $\mathbf{s}$ to move in a direction where the function locally looks the steepest. The algorithm utilizing the proximal operator is called *proximal gradient algorithm* [Tib15] and is defined as,

$$\mathbf{s}_{t+1} = \text{prox}_{\eta h}(\mathbf{s}_t - \eta\nabla f(\mathbf{s}_t)).\tag{2.13}$$

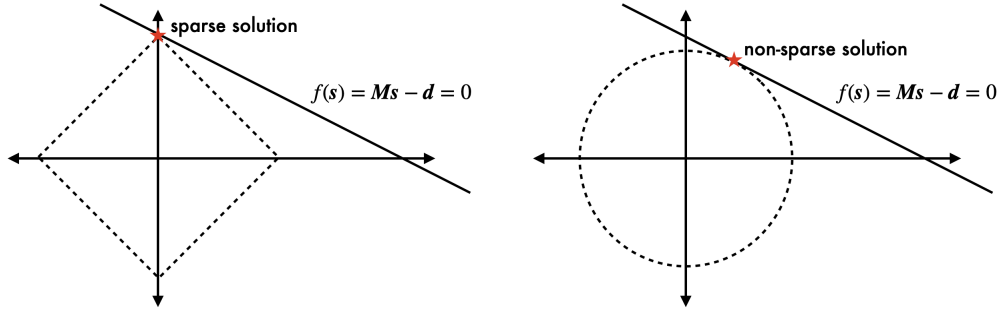


Figure 2.2.: The left figure shows the diamond-shaped contours of the constant l_1 -norm, the right figure shows the circular contours of the l_2 -norm. The solutions of the first term are located along the solid line. Evidently, the l_1 -minimization gives a sparser solution because one of the elements of $\mathbf{s}$ is zero. This also applies for higher dimensions.

The primary motivation for developing this kind of proximal operator was minimizing composite functions which consist of a Lipschitz convex function $g(\mathbf{s})$ and an easy to calculate proximal operator $h(\mathbf{s})$ [Car20d],

$$\min : f(\mathbf{s}) = g(\mathbf{s}) + h(\mathbf{s}) \quad (2.14)$$

the easy to calculate proximal operator in the Lasso regression is the l_1 -norm,

$$h(\mathbf{s}) = \|\mathbf{s}\|_1 = \sum |s_i|. \quad (2.15)$$

For this function, the proximal operator is [Car20d],

$$\text{prox}_{\eta h} = \operatorname{argmin}_{\mathbf{u}} : \|\mathbf{u}\|_1 + \frac{1}{2\eta} \|\mathbf{u} - \mathbf{s}\|_2^2, \quad (2.16)$$

and $\mathbf{u}$ is an optimal solution if,

$$0 \in \partial \|\mathbf{u}\|_1 + \frac{1}{\eta} (\mathbf{u} - \mathbf{s}), \quad (2.17)$$

where ∂ denotes the sub-differential. The previous equation is equivalent to,

$$(\mathbf{s} - \mathbf{u}) \in \eta \partial \|\mathbf{u}\|_1. \quad (2.18)$$

where a point $\mathbf{z}$ is an element of $\partial \|\mathbf{u}\|_1$ if,

$$\mathbf{z} \in \partial \|\mathbf{u}\|_1 \text{ if } z_i = \begin{cases} \operatorname{sign}(u_i) & u_i \neq 0 \\ \in [-1, 1] & u_i = 0 \end{cases}, \quad (2.19)$$

so for $(\mathbf{s} - \mathbf{u})$, this means,

$$(\mathbf{s} - \mathbf{u}) \in \eta \partial \|\mathbf{u}\|_1 \rightarrow \mathbf{u} : u_i = \begin{cases} s_i - \eta & s_i \geq \eta \\ 0 & |s_i| \leq \eta \\ s_i + \eta & s_i \leq -\eta \end{cases}, \quad (2.20)$$

This just means that if a value s_i , which is an element of $\mathbf{s}$, is greater than the value η , then the value of s_i gets reduced by η , if a value s_i is between η and $-\eta$, then s_i gets reduced to zero and finally if s_i is less than the value $-\eta$, then the value of s_i gets increased by η . Sometimes this operator is also called the *shrinkage* or *soft-thresholding operator* [Tib15, Car20d].

Iterative Shrinkage Thresholding Algorithm (ISTA)

This entire section closely follows [BT09, Car20e]. The *iterative shrinkage thresholding algorithm* utilizes the proximal gradient method for solving the Lasso regression with faster convergence rates than the subgradient method ($O(t^{-1/2})$ for the subgradient method, where t is the number of iterations compared to $O(t^{-1})$ for the proximal gradient method [BT09, Car20e]). The solution $\tilde{\mathbf{s}} \in \mathbb{R}^M$ is seen only through linear measurements,

$$\mathbf{d} = \mathbf{A}\tilde{\mathbf{s}}, \quad (2.21)$$

Here, $\mathbf{A} = \mathbf{M}\Psi$, where Ψ represent a universal basis transformation matrix. The function to be minimised is,

$$f(\tilde{\mathbf{s}}) = g(\tilde{\mathbf{s}}) + h(\tilde{\mathbf{s}}) = \|\mathbf{A}\tilde{\mathbf{s}} - \mathbf{d}\|_2^2 + \lambda\|\tilde{\mathbf{s}}\|_1. \quad (2.22)$$

Applying regular gradient descent (equation (2.12)) to $g(\tilde{\mathbf{s}})$ and using the proximal gradient algorithm as defined in equation (2.13), where $\eta = \frac{1}{\beta}$ and g is β -smooth³ we can write,

$$\begin{aligned} \tilde{\mathbf{s}}_{t+1} &= \operatorname{argmin}_{\tilde{\mathbf{s}}} : \lambda\|\tilde{\mathbf{s}}\|_1 + \frac{\beta}{2}\|\tilde{\mathbf{s}} - (\tilde{\mathbf{s}}_t - \frac{1}{\beta}\nabla g(\tilde{\mathbf{s}}_t))\|_2^2 \\ &= \operatorname{argmin}_{\tilde{\mathbf{s}}} : \frac{\lambda}{\beta}\|\tilde{\mathbf{s}}\|_1 + \frac{1}{2}\|\tilde{\mathbf{s}} - (\tilde{\mathbf{s}}_t - \frac{1}{\beta}\nabla g(\tilde{\mathbf{s}}_t))\|_2^2 \\ &= \operatorname{prox}_{\lambda/\beta, l1}(\tilde{\mathbf{s}}_t - \eta\nabla g(\tilde{\mathbf{s}}_t)). \end{aligned} \quad (2.23)$$

Computing the gradient of $g(\tilde{\mathbf{s}})$ yields,

$$\tilde{\mathbf{s}}_{t+1} = \operatorname{prox}_{\lambda/\beta, l1}(\tilde{\mathbf{s}}_t - 2\eta\mathbf{A}^T(\mathbf{A}\tilde{\mathbf{s}}_t - \mathbf{d})), \quad (2.24)$$

and applying this to the compressed sensing use case,

$$\begin{aligned} \mathbf{s}_{t+1} &= \operatorname{prox}_{\lambda/\beta, l1}(\tilde{\mathbf{s}}_t - 2\eta\mathbf{A}^T(\mathbf{M}\Psi\tilde{\mathbf{s}}_t - \mathbf{d})) \\ &= \operatorname{prox}_{\lambda/\beta, l1}(\Psi^T\mathbf{s}_t - 2\eta\Psi^T\mathbf{M}^T(\mathbf{M}\Psi\tilde{\mathbf{s}}_t - \mathbf{d})) \\ &= \operatorname{prox}_{\lambda/\beta, l1}(\Psi^T\mathbf{s}_t - 2\eta\Psi^T\mathbf{M}^T(\mathbf{M}\mathbf{s}_t - \mathbf{d})). \end{aligned} \quad (2.25)$$

Here, Ψ , $\mathbf{M}$ and $\mathbf{d}$ are the universal orthogonal transformation matrix, measurement mask and subsampled image from Equation 2.10. The implementation recipe starts by calculating $\mathbf{M}^T(\mathbf{M}\mathbf{s}_t - \mathbf{d})$ after which the forward transformation Ψ^T and subsequently the soft-thresholding operator is applied. Finally, $\tilde{\mathbf{s}}$ is inversely transformed back to pixel space and the process can be repeated for the next iteration step. The basis transformations are alternating between Fourier space and wavelet space in each iteration step. The compressed sensing code used in this thesis is based on [Kli20].

³The gradient of the function is smooth with Lipschitz constant β .

Similarities

Finally, I am briefly going to highlight the similarities between the different reconstruction methods starting with the compressed sensing loss function,

$$\|\mathbf{M}\Psi\tilde{\mathbf{s}} - \mathbf{d}\|_2^2 + \lambda\|\tilde{\mathbf{s}}\|_1. \quad (2.26)$$

The Bayesian posterior probability density function for the special case of Gaussian random fields which has been shown to coincide with the Wiener filter minimum error estimate [ZHFL94], is given by

$$P(\mathbf{s}|\mathbf{d}) \propto P(\mathbf{s})P(\mathbf{d}|\mathbf{s}), \quad (2.27)$$

where $\mathbf{d} = \mathbf{M}\mathbf{s} + \mathbf{n}$. $\mathbf{M}$ is the measurement matrix and $\mathbf{n}$ is a noise term. Computing the posterior yields,

$$P(\mathbf{s}|\mathbf{d}) \propto \exp \left[-\frac{1}{2} \left(\mathbf{s}^T \mathbf{S}_{MM}^{-1} \mathbf{s} + (\mathbf{d} - \mathbf{M}\mathbf{s})^T \mathbf{N}^{-1} (\mathbf{d} - \mathbf{M}\mathbf{s}) \right) \right]. \quad (2.28)$$

$\mathbf{S}_{MM}$ is the map-map covariance matrix (for more details refer to [ZHFL94]). By writing equation (2.28) in norm notation,

$$\frac{1}{N^2} \|\mathbf{d} - \mathbf{M}\mathbf{s}\|_2^2 + \mathbf{s}^T \mathbf{S}_{MM}^{-1} \mathbf{s}, \quad (2.29)$$

the similarities with the other reconstruction methods become clear where the second term in equation (2.29) acts as kind of a regularizer. Minimizing (2.29) essentially provides the same result as solving equation (2.7). ORCA in norm notation,

$$\mathcal{L} = \|\mathbf{M}\mathbf{s} - \mathbf{d}\|_2^2 + k_1 \|\mathbf{G}_{smoothing}(\mathbf{s}) - \mathbf{s}\|_2^2 + k_2 \|\text{clip}(\mathbf{s}, 1, +\infty)\|_1 + k_3 \|\text{clip}(\mathbf{s}, 0, \alpha)\|_1, \quad (2.30)$$

also highlights the resemblance, which is characterized by an identical least squares regression term followed by some kind of regularizer multiplied by a scaling factor,

$$\|\mathbf{M}\mathbf{s} - \mathbf{d}\|_2^2 + \text{regularizer} \quad (2.31)$$

The only exception in the resemblance is that the compressed sensing loss function is formulated in an alternative orthogonal basis.

2.5. Machine Learning

A different way in which equation (2.1) can be solved for Lyman-alpha-forest flux reconstructions is with machine learning. "The machine described is designed to operate as human beings seem to" [Sol57], is a succinct illustration of the general idea behind how machine learning methods work, the details, however, are going to be presented in this section starting with the motivation behind using this approach, followed by a deep dive into generative adversarial networks and finally the exact neural network architecture used in this thesis with which the reconstructions are produced. The entire subsection 2.5.1 is primarily based on [AZH⁺21, ON15, BB19, NH10, SHK⁺14, GPAM⁺14, IZZE17, RFB15, Rud16].

The general idea

Since it's fair to think of three-dimensional Lyman-alpha forest flux field reconstruction as way of "translating" sparsely sampled input fields into corresponding full map output fields, the idea was to use a neural network to learn a mapping from the discrete set of data $\mathbf{d}$ to the fully reconstructed field $\mathbf{s}$. The training data set is produced using the PPT formalism described in chapter 1, consisting of five hundred realizations of a Lyman-alpha-forest-like flux field in redshift space with a size of 64^3 pixels in 3D and 64^2 pixels in 2D. In order to understand the architecture of the neural network employed for this task, it's important to outline what a neural network is, how training works and what the functions within a neural network do.

2.5.1. Neural networks, training and activation functions

Neural networks (NNs) [AZH⁺21] are computing systems inspired by biological neural networks in animal brains, consisting of connected nodes called *neurons*. These neurons receive, process and send signals, in the form of real numbers to other neurons connected to it. Usually, neurons are organized in layers, where each layer performs a specific transformation of their inputs and the neuron outputs are calculated with *activation functions*, which are non-linear, allowing NNs to learn complex patterns. Most commonly, neural networks are trained with a type of gradient descent. The choice of the type of gradient descent depends on the application.

In pix2pix, the neural network is trained with ADAM [KB14], a variant of stochastic gradient descent. In a nutshell, stochastic gradient descent minimizes the loss function in a way where the weights of the nodes in the neural network w are updated for each training example according to [Rud16],

$$w_{t+1} := w_t - \eta \nabla \mathcal{L}(w_t, t_i), \quad (2.32)$$

where η is the step size and t_i is a training example from the training set.

The layer transformations used for my machine learning implementation are convolution [BB19], batch normalization, dropout [SHK⁺14], ReLU [NH10] and leaky ReLU just like in [IZZE17]. For more detailed information, please refer to the aforementioned references.

Convolution

Essentially, a convolution [BB19] computes the scalar product of a filter sized input patch and the filter itself, where the filter itself is also trainable. The purpose of this operation is extraction of regularities and patterns contained within the data. This process is systematically repeated across the entire input dataset where the stride length, which is a measure of how far the shift between two individual filter sized patches is, is set before initialization of the training. In my implementation the stride is chosen in way that causes downsampling of the input by a factor of two which can also be seen in Figure 2.4. The patterns and regularities are subsequently stored in a feature map.

Batch normalization

After the feature map is created, it can be normalized using a batch normalization layer [STIM18] which standardizes the data so that the mean is zero and standard deviation is one. In simple terms, batch norm prevents changes in the distribution of layer inputs which negatively impacts training. Accelerated convergence and easier initializations of training weights are also among the many reasons why batch norms are employed.

Rectified linear activation and leaky rectified linear activation

The rectified linear activation function or ReLU [NH10] is a function that sets negative values to zero and otherwise outputs everything directly, i.e. $f_{\text{ReLU}}(x) = \max\{0, x\}$. The benefits associated with ReLU usage are computational simplicity, representational sparsity⁴ and linear properties⁵ while concurrently being non-linear, or piecewise linear to be precise [GBB11]. Leaky Rectified Linear Unit, or Leaky ReLU, is a type of activation function based on a ReLU, but with a small slope for negative values instead of a flat slope. This type of function is used to counteract the effect of gradients becoming too sparse which can interfere with learning. The slope coefficient is determined before training.

Dropout

A dropout [SHK⁺14] layer randomly sets nodes to zero with a frequency which can be specified to reduce overfitting. Figure 2.3 shows a visual representation of a generic example of an artificial neural network architecture.

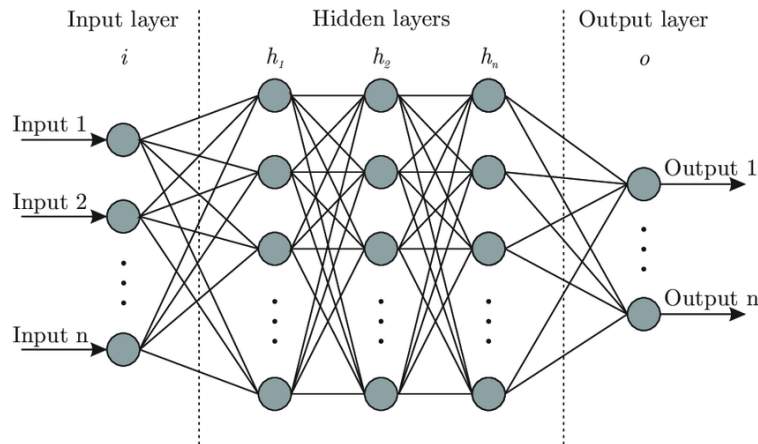


Figure 2.3.: Generic example of an artificial neural network architecture with input layers displayed on the left, hidden layers in the middle and output layers on the right. The filled in circles represent the nodes and the connections between the nodes are shown as black lines. The number of nodes within a hidden layer as well as the number of in- and outputs is greatly varied and depends on the concrete machine learning task. This figure is taken from [BGF17].

The reason why I chose to reuse these activation functions for cosmic web reconstruction applications is because, as demonstrated in [IZZE17], they work very well for any kind of two-dimensional image-to-image translation problems and there is no reason why the applicability should be limited to two dimensions. In fact, other areas of research have also used three-dimensional versions of pix2pix [CAE20].

⁴Desirable quality because the output can be exactly zero, simplifying the model and accelerating learning.

⁵Linear properties are beneficial because they prevent non-saturating gradients which would interfere with learning otherwise.

2.5.2. Conditional generative adversarial networks

Generative adversarial networks (GANs) [GPAM⁺14] are machine learning models used to make predictions by learning a mapping from a random noise vector $\mathbf{z}_n$ which seeds the generative process to an output vector $\mathbf{s}$,

$$G : \mathbf{z}_n \rightarrow \mathbf{s}. \quad (2.33)$$

There are two models embedded within the architecture of a GAN, the generator G and the discriminator D . In contrast to the WF, which has been shown to maximize the a-posteriori estimate [ZHFL94] for Gaussian fields, G generates samples following an implicitly defined conditional probability distribution $p_g(\mathbf{s}|\mathbf{d})$ through a maximum likelihood process with the goal of choosing parameters in a way so that the likelihood⁶ of training data is maximized [Goo16]. Performing this operation in log space simplifies the derivatives, since the product over examples is substituted by a sum according to,

$$\begin{aligned} \mathbf{w}^* &= \arg \max_{\mathbf{w}} \prod_{i=1}^m p_{model}(\mathbf{w}, t_i) \\ &= \arg \max_{\mathbf{w}} \log \prod_{i=1}^m p_{model}(\mathbf{w}, t_i) \\ &= \arg \max_{\mathbf{w}} \sum_{i=1}^m \log p_{model}(\mathbf{w}, t_i). \end{aligned} \quad (2.34)$$

here, $\mathbf{w}$ are the parameters which parameterize the probability distribution, t_i is a training example from a dataset consisting of m training examples.

The other architectural constituent of a GAN is the discriminator D which is a classification model employed during training with the purpose of encouraging the generator to produce outputs that are difficult to distinguish from true datasets by detecting unconvincing generator realizations. The way in which this process takes place is through discrimination of sample data and corresponding generator updates, which, after many training steps, leads to convergence towards $D^*(\mathbf{s}) = \frac{p_d(\mathbf{s})}{p_d(\mathbf{s}) + p_g(\mathbf{s})} = \frac{1}{2}$, where p_d is the probability distribution of true data [GPAM⁺14]. In practice, this produces a realization with real looking structures at scales that in reality cannot be resolved. In other words these structures are regarded as "fake" by the discriminator. In pix2pix, the generator uses a "U-Net"-based architecture [RFB15], and the discriminator uses a convolutional "PatchGAN" classifier punishing structures on image patch scales [IZZE17].

Conditional generative adversarial networks (cGANs) [IZZE17] on the other hand make predictions by learning a mapping from a dataset $\mathbf{d}$ and a random noise vector $\mathbf{z}_n$, to $\mathbf{s}$,

$$G : \{\mathbf{d}, \mathbf{z}_n\} \rightarrow \mathbf{s}. \quad (2.35)$$

The objective of the cGAN is,

$$\mathcal{L}_{cGAN}(G, D) = E_{\mathbf{d}, \mathbf{s}}[\log D(\mathbf{d}, \mathbf{s})] + E_{\mathbf{d}, \mathbf{s}}[\log(1 - D(\mathbf{d}, G(\mathbf{d}, \mathbf{z}_n)))]. \quad (2.36)$$

G is minimizing this objective and the adversarial D is maximizing it [GPAM⁺14],

$$(G^*, D^*) = \arg \min_G \max_D \mathcal{L}_{cGAN}(G, D), \quad (2.37)$$

Additionally, in pix2pix [IZZE17] it has been found that adding an l_2 - or an l_1 -loss term to the GAN objective improves reconstruction quality where l_1 produces less blurry results making the final objective,

$$(G^*, D^*) = \arg \min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_l(G), \quad (2.38)$$

⁶The likelihood refers to the weights.

where the l_1 -term is defined as,

$$\mathcal{L}_{l_1}(G) = E_{\mathbf{d}, \mathbf{s}, \mathbf{z}_n} [\|\mathbf{s} - G(\mathbf{d}, \mathbf{z}_n)\|_1]. \quad (2.39)$$

In application scenarios equation (2.36) sometimes has insufficient gradient for the generator to learn appropriately leading to incorrect rejections from the Discriminator [GPAM⁺14]. The outcome is saturation of $\log(1 - D(\mathbf{d}, G(\mathbf{d}, \mathbf{z}_n)))$. Instead, the generator can be trained to maximize $\log(D(\mathbf{d}, G(\mathbf{d}, \mathbf{z}_n)))$ which results in steeper gradients in the early learning phase.

It's important to emphasize that a conditional GAN solves equation (2.1) in a completely different way than the previously discussed methods, the big difference being the creative approach to artificially fabricating structures on scales that in reality cannot be resolved. This effect can be circumvented by achieving highly stochastic outputs, representing the complete entropy of conditional distributions they model, which my implementation doesn't produce. While it is unclear why activating dropout layers during testing doesn't achieve high stochasticity, using an alternative loss function [PTVP21] has been proven to work and may remedy the issue. An easy and alternative method of using a U-Net with only l_2 -loss, which doesn't require stochastic outputs⁷, is also included in the results in chapter four. The expected behaviour for this approach is a realization with blurrier visual characteristics and lower spatial coherence on small scales.

2.5.3. Network architecture

The network architecture used is based on U-Net, which is a type of encoder-decoder network utilising skip connections between layers [RFB15]. The input and output in our case have the same underlying structure making skip connections a convenient way for transferring information from one layer to another. The skip connections are added in between each layer i and layer $j - i$, where j is the total number of layers. All data at layer i is concatenated with data at layer $j - i$ at each connection. The encoder and decoder contain convolutions down- and upsampling by a factor of two respectively [IZZE17]. All the structural information provided here is clearly visualized in Figure 2.4.

If we use Ck for a Convolution-BatchNorm-ReLU layer with k , $4 \times 4 \times 4$ spatial filters applied with a stride 2 and CDk for a Convolution-BatchNorm-Dropout-ReLU layer with a dropout rate of fifty percent then we can concisely summarize the encoder and decoder architecture as follows:

$$\begin{aligned} \text{encoder: } & C64 - C128 - C256 - C512 - C512 - C512 \\ \text{decoder: } & CD512 - C512 - C256 - C128 - C64 \end{aligned} \quad (2.40)$$

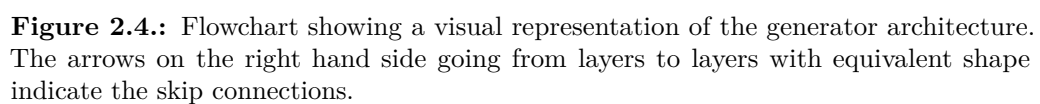
Following the last layer, the output is once again convolved for mapping to the number of output channels, after which a sigmoid function is applied. The only exception to the notation introduced above is that batch normalization is not applied to the first $C64$ layer in the encoder, and for a batch size of one, there is also no batch norm in the bottleneck [IZZE17]. The ReLU functions in the encode are leaky and have a slope of 0.3, whereas the ReLUs in the decoder are regular not leaky ReLU functions.

The discriminator architecture is:

$$\text{Discriminator: } C64 - C128 - C256 - C512 \quad (2.41)$$

Following the last layer, the output is once again convolved for mapping to a one-dimensional output, after which a Sigmoid function is applied. In the discriminator the Batch-normalization is also not applied to the first $C64$ layer. The ReLU functions are leaky and have slope of 0.3 [IZZE17].

⁷Only the expected error between true and reconstructed field is minimized.



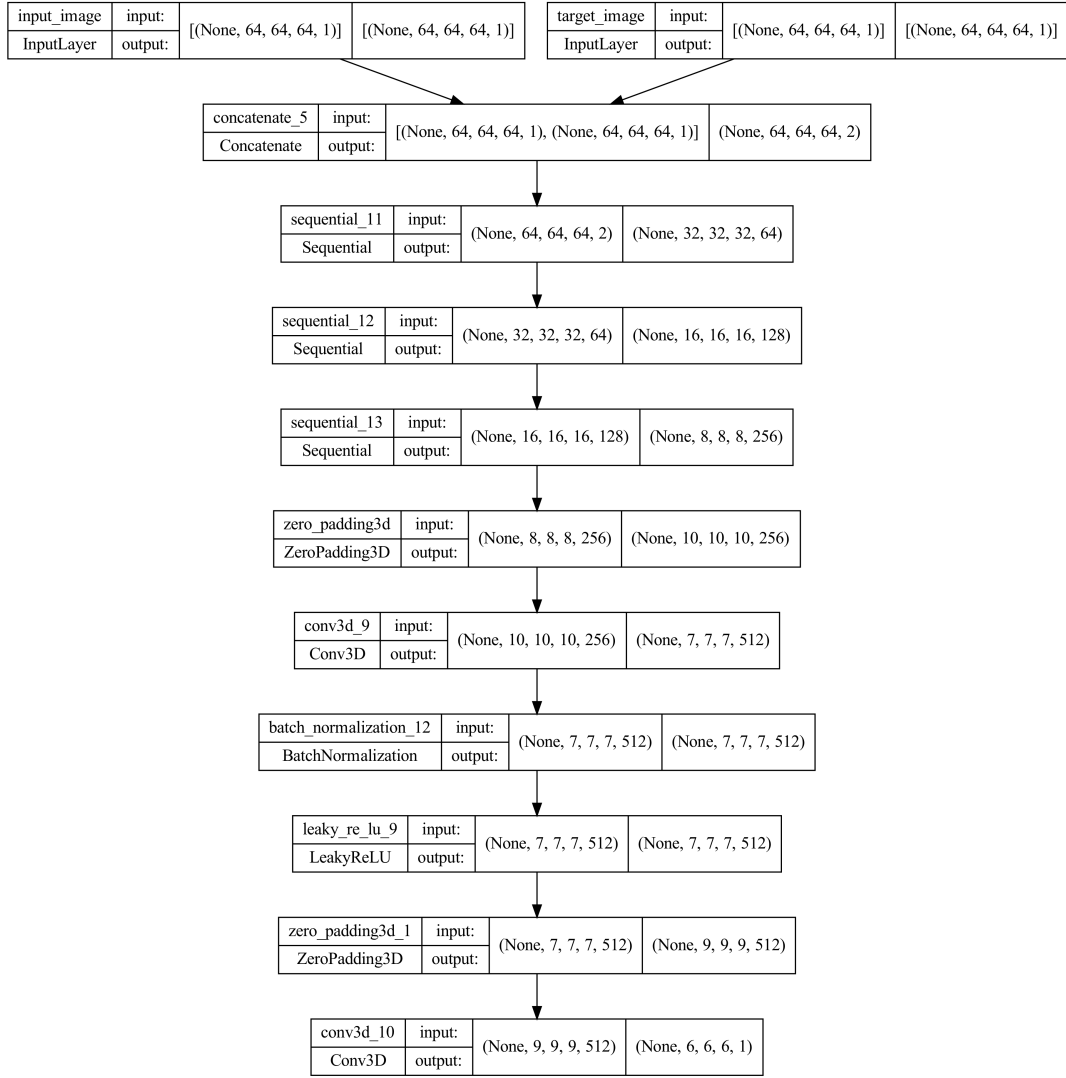


Figure 2.5.: Flowchart showing a visual representation of the discriminator architecture.

$$\text{U-Net decoder: } CD512 - C1024 - C1024 - C512 - C256 - C128 \quad (2.42)$$

Figure 2.4 and Figure 2.5 show visual representations of the full generator and discriminator architectures including encoding, decoding and skip connections [IZZ17].

3. Results

This chapter is going to discuss the reconstruction results achieved using the methods described in chapter 2 starting with an illustration of the shift from a Gaussian to a non-Gaussian regime followed by a thorough analysis of optimal Wiener filter parameters. The section 3.2 is going to focus on the time evolution of large scale structures, only analysing two dimensional face on slices of the simulated and reconstructed Lyman-alpha forest flux fields. Finally, the section 3.3 reveals the full three-dimensional reconstructions from sparsely sampled data located on line of sight skewers and a computational cost analysis. Throughout this chapter the evaluation of reconstruction quality is assessed in terms of,

- residual standard deviation calculations between true and reconstructed fields,
- power spectrum reproduction,
- probability density function reproduction, and
- spatial coherence,

where each displayed result is an average of ten realization evaluations.

The shift to non-Gaussianity

In order to demonstrate the shift from a Gaussian to a non-Gaussian regime Figure 3.8 shows the flux probability density function (PDF) for early and late times. The late times correspond to the Lyman-alpha forest as it can be observed at present times and the early times correspond to D_+ values as low as $D_+ = 0.03628$, at typically observed Lyman-alpha forest redshifts of $z \approx 2 - 3$.

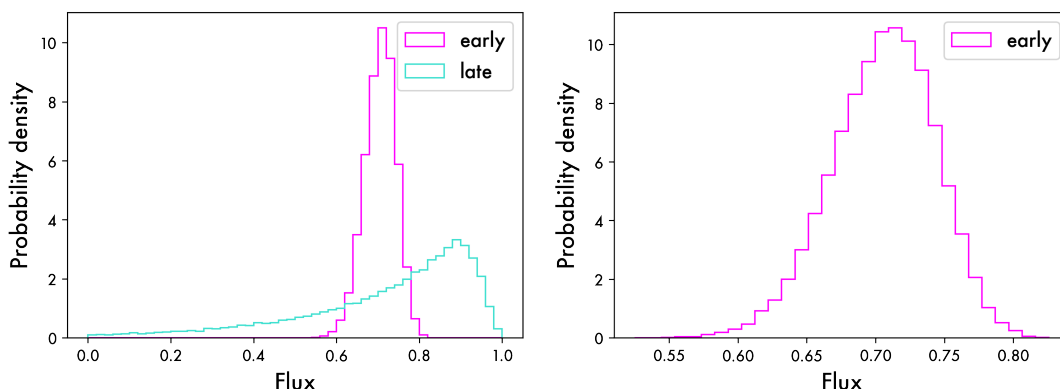


Figure 3.1.: Early and late universe flux probability density functions (PDFs). Clearly the PDF labelled ‘early’ is much closer to a Gaussian distribution than the blue one labelled ‘late’. This is exactly the reason why the Wiener filter is suboptimal for Lyman-alpha forest reconstructions.

It's not obvious that the flux PDF is Gaussian for a Gaussian density field, in this case however, the fluctuating Gunn Peterson approximation discussed in the previous section produces an optical depth in a range that doesn't significantly impact the flux PDF shape, compared to the density PDF, the flux PDF actually becomes slightly more Gaussian for this specific case as can be seen in Figure 3.2.

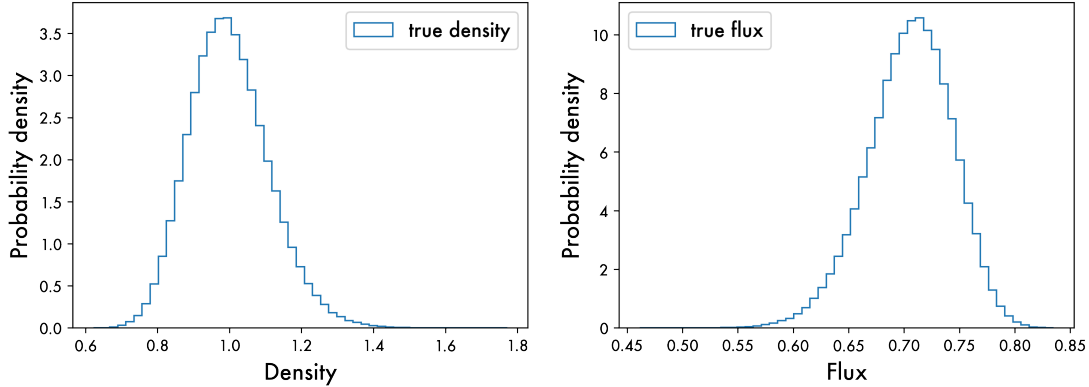


Figure 3.2.: True density and true flux PDFs for early times.

3.1. Wiener filter parameters

Before applying the Wiener filter, it is important to determine the three free parameters embedded in the Wiener filter equation for which the reconstruction becomes optimal in terms of a minimal error estimate. In this section the analysis is done for 2D reconstructions going from an early Gaussian to a late non-Gaussian structure of the universe. This is achieved by plotting 3D parameter surfaces for ten individual realizations, averaging them into one surface as shown in Figure 3.3 and taking note of the values where the surface reaches minimal residual standard deviation between true and reconstructed flux field (henceforth, all residual standard deviations are understood to be between true and reconstructed flux fields).

Figure 3.3 shows the Wiener filter parameter analysis for a Gaussian flux distribution of the early universe in the first column and for a late non Gaussian distribution in the middle column. The figures in row one and four display the same plot from different perspectives. The same applies for the figures in row two and three, the ones in row three and four show the plot face on and the ones in row one and two show it from the side, rotated clockwise ninety degrees. The red lines indicate the locations where the residual standard deviation is minimal, i.e. where the Wiener filter reaches maximum reconstruction quality. The plots in row one and four show residual standard deviation as a function of correlation length and amplitude, whereas the ones in row two and three show residual standard deviation as a function of correlation length and noise. The same analysis can be done for a late non-Gaussian universe and is displayed in the right column in Figure 3.3. In three dimensions $L_{\perp}$ is usually chosen to be on the order of mean line of sight distance [LHC21] which is sufficiently close to the result in my analysis. Table 3.1 summarizes the optimal parameters used for individual reconstructions. The ORCA parameters do not have the same effect as the ones in the Wiener filter, i.e. changing the noise term to an extremely low value simply causes the smoothing and clipping terms to lose weight almost completely, which is why they are chosen according to [LHC21] across the board for early as well as late times.

Table 3.1.: Table summarizing the optimal parameters for Wiener filtering and ORCA. The ORCA parameters do not have the same effect as the ones in the Wiener filter which is why they are chosen according to [LHC21] across the board for early as well as late times. The hyperparameter in compressed sensing is the $l1$ -term scaling factor λ which is varied as a function of iteration step.

Parameter	correlation length	amplitude	noise	k_1	k_2	k_3	α
2D WF early	0.35	0.05	0.0075	/	/	/	/
2D WF late	0.07	0.05	0.06	/	/	/	/
3D WF late	$(1/n_{los})^{1/2} = 0,039$	0.05	0.04	/	/	/	/
ORCA	$(1/n_{los})^{1/2} = 0.039$	/	0.2	5	0.3	0.025	0.55

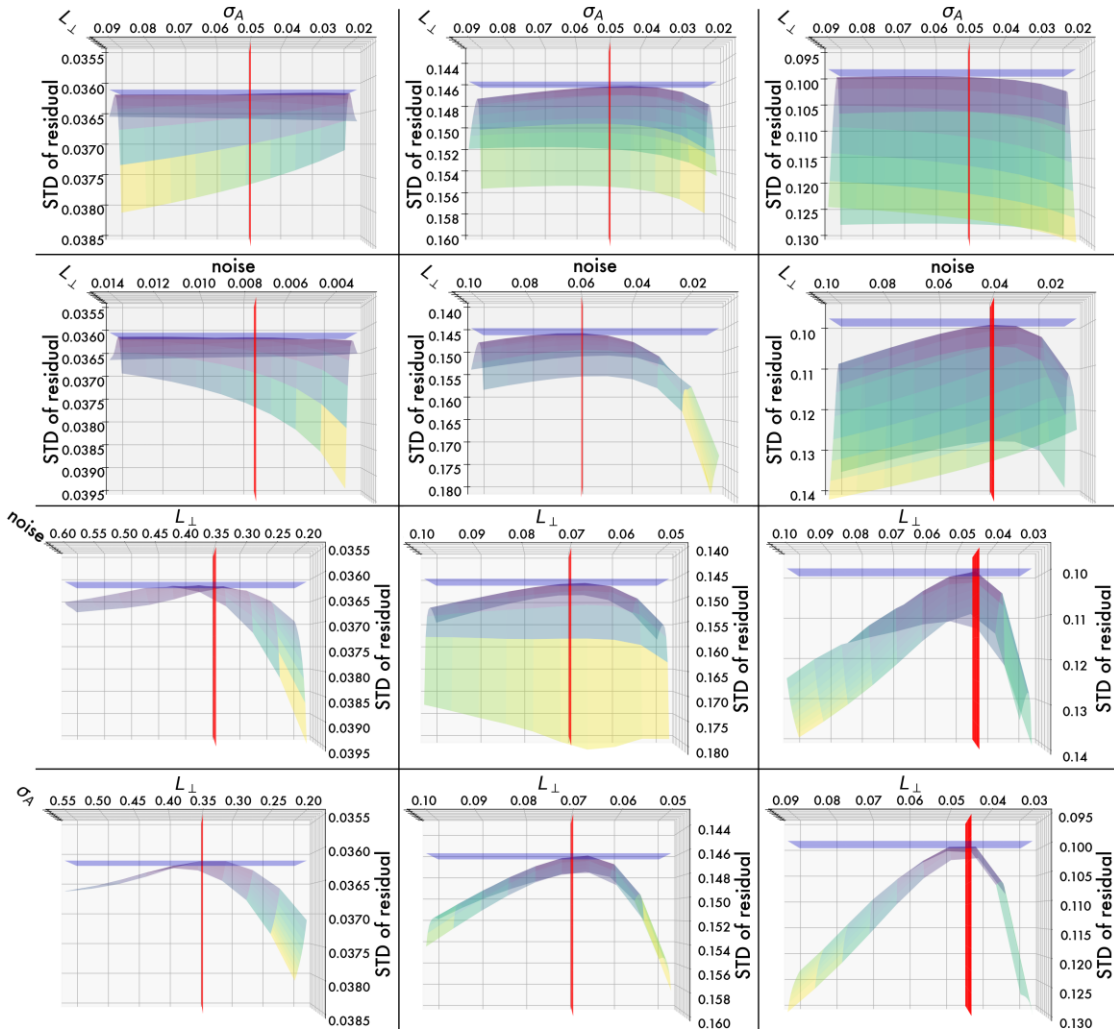


Figure 3.3.: Wiener filter parameter analysis for Gaussian and non-Gaussian flux distributions of the early and late universe for two and three dimensional reconstructions. The red lines indicate the correct parameter values where the residual standard deviation is minimal, i.e. where the Wiener filter reaches maximum reconstruction quality. The plots in row two and three show residual standard deviation as a function of correlation length and noise, whereas the ones in row one and four show residual standard deviation as a function of correlation length and amplitude. The optimal parameter values in the three dimensional case are different from the two dimensional case since quantity of line of sights dictates which correlation length, noise and amplitude are optimal.

3.2. 2D early and late universe Lyman-alpha flux reconstructions

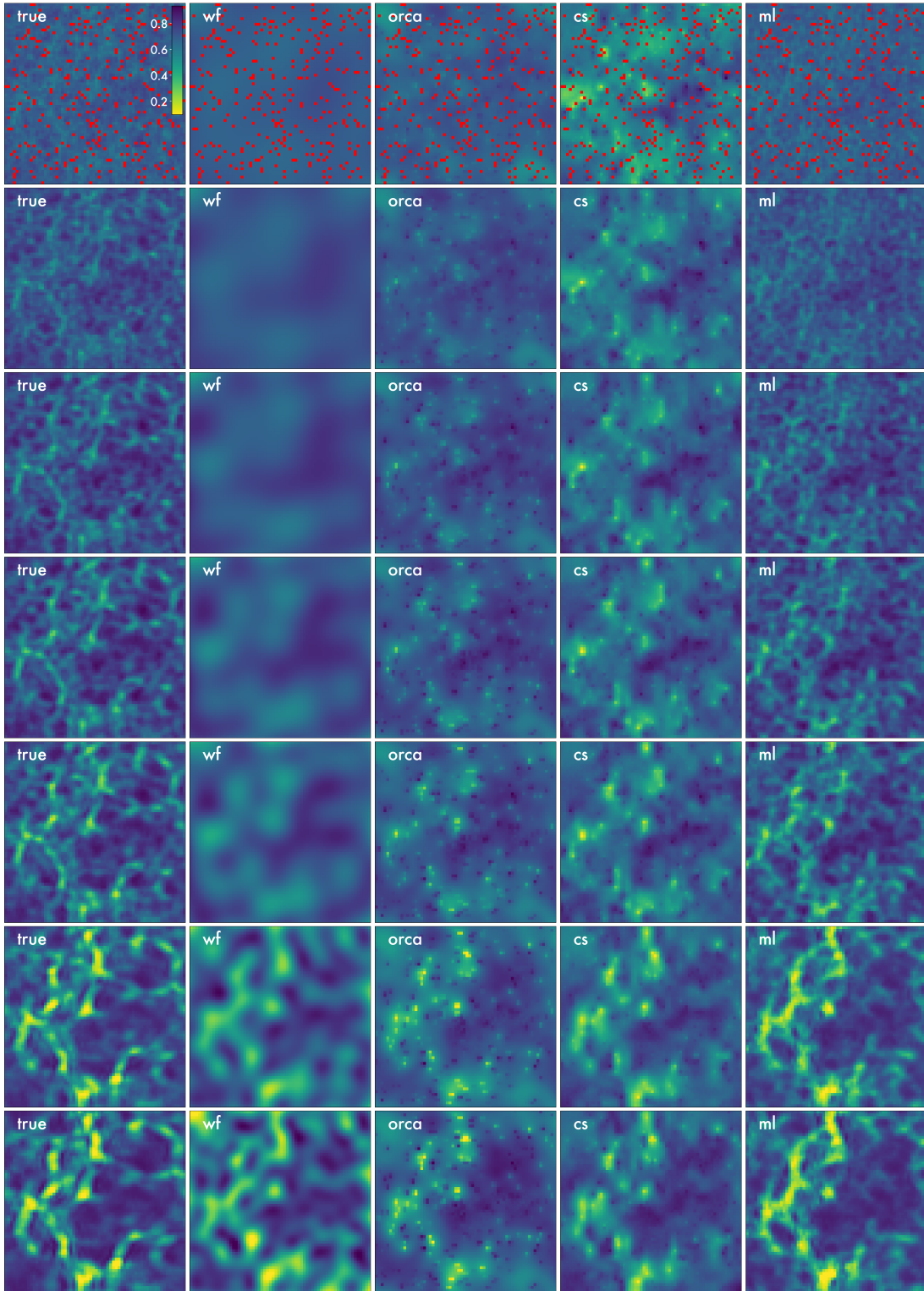


Figure 3.4.: 2D true fields and corresponding Wiener filter, ORCA, compressed sensing and machine learning reconstructions ranging from early Gaussian to late non Gaussian flux fields displayed along the columns. The red dots in the first row represent the 327 given measurement points from which the reconstruction is made.

Visually, each reconstruction method produces vastly different results as seen in Figure 3.4. The Wiener filter can generally be described as blurry, whereas the machine learning approach, which is also conditioned on the growth factor, yields the most authentic looking result, however, often guessing what the field could look like given the data its provided. ORCA and compressed sensing reconstructions can generally be described as somewhat discontinuous, though compressed sensing seems less discontinuous than ORCA. As expected, for early times compressed sensing is the worst reconstruction method. The reason for this is that, in Fourier and Wavelet space, Gaussian fields are maximally non-sparse, making them the literal antithesis of what compressed sensing is made to reconstruct.

3.2.1. Residual standard deviation

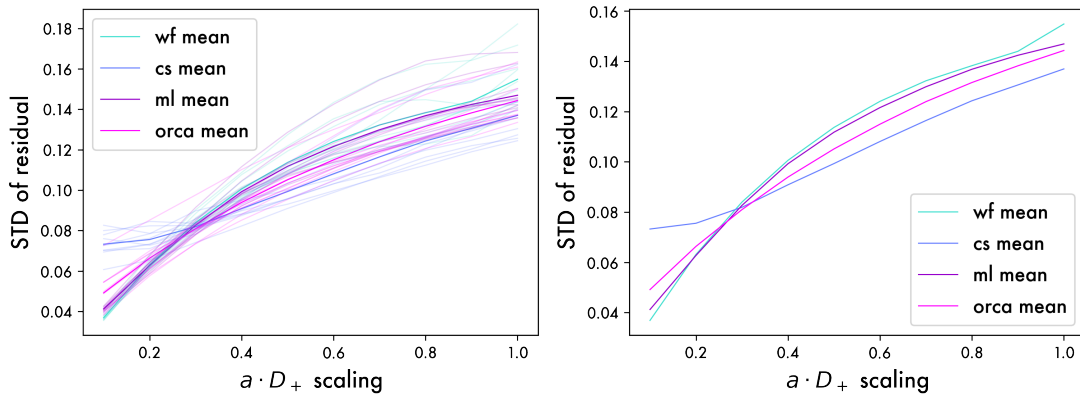


Figure 3.5.: Residual standard deviations for different D_+ values ranging from early to late times in the universe where $a = 0.3628$. The STD calculation is done on ten different fields, the plot on the right shows the averages of all results for each reconstruction method shown on the left. The Wiener filter parameters are adjusted individually for each D_+ scaling step.

In Figure 3.5 it quickly becomes apparent that the Wiener filter delivers on its promise of being optimal for Gaussian fields in terms of a minimal error reconstruction. At late times, however, compressed sensing appears to yield the best result in this regard.

3.2.2. Power- and cross spectra

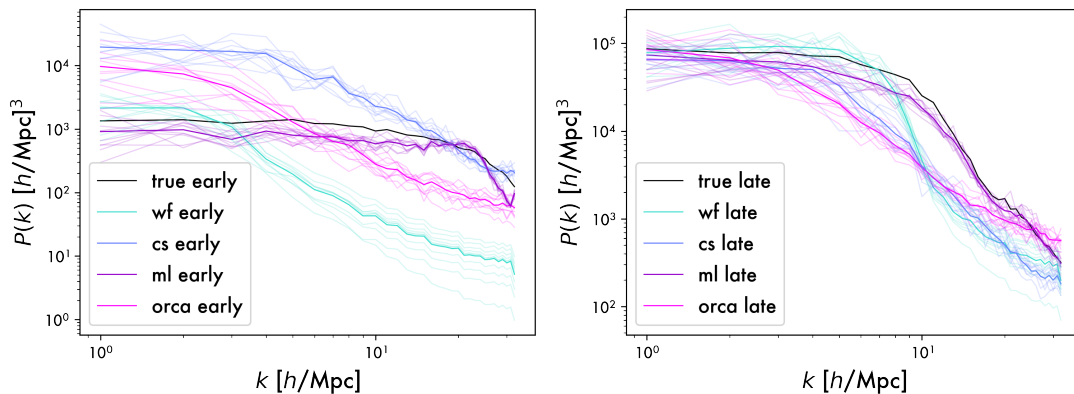


Figure 3.6.: The plots show the power spectra for each reconstruction method for an early Gaussian flux distribution (left) and the same for a late non Gaussian distribution (right). For early and late times machine learning seems to reproduce the power spectrum best.

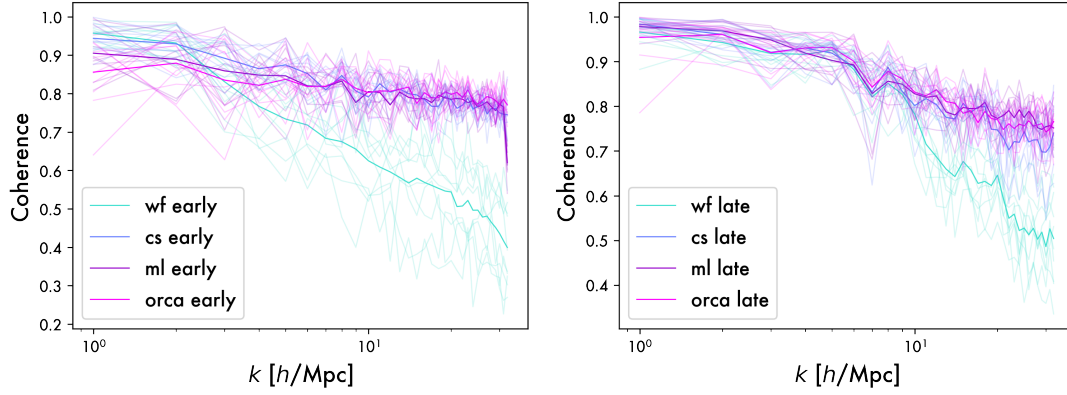


Figure 3.7.: The left plot shows the cross spectra for each reconstruction method for an early Gaussian Flux distribution and the one the right shows the same for a late non Gaussian distribution. Only for very large scales and early times is spatial coherence exceptionally good in the Wiener filter reconstruction, due to the blurry nature of the reconstruction, however, coherence drops significantly towards smaller scales.

The true power spectrum is most accurately reproduced by the machine learning method, for early as well as late times equally, which can be seen in Figure 3.6. In their respective reconstructions ORCA and compressed sensing are adding fictional structures at the measurement locations which at early times aren't present resulting in an incorrect power spectrum. At late times, the Wiener filter follows the power spectrum very closely for large scales, on mid- to small scales, however, the power spectrum deviates significantly. Compressed Sensing and ORCA reproduce the power spectrum best at small and large scales, on medium scales, the deviation appears to be largest compared to the other two methods. The cross spectra in Figure 3.7 really highlight an important characteristic of the Wiener filter, high accuracy on large scales and low accuracy on small scales and it is indeed the blurriness that is present in the reconstructions which is causing some loss of information on these scales. Compressed sensing, ORCA and machine learning reconstructions are almost equally good with the machine learning approach narrowly beating the other methods by a small margin for medium to small scales. An additional comparison between the full cGAN+l1-loss and a version only utilizing the Euclidean loss in Figure 3.9 and Figure 3.10 also clearly demonstrates the benefits of a discriminator, improvement of PDF, power- and cross-spectra without much impact on residual STD. In all regards except for residual STD for very early times, l_2 still outperforms the Wiener filter, because the prior is implicitly given through training data. This is a simple way of utilizing machine learning for three-dimensional tomography without the problem of fictional structure fabrication at scales that in reality cannot be resolved.

3.2.3. Probability density functions

In terms of the flux probability density functions shown in Figure 3.8 compressed sensing clearly fails in the early Gaussian case for the same aforementioned reasons. ORCA and machine learning reproduce the almost perfectly Gaussian PDF more closely than any of the other methods. For late times where the true PDF evidently becomes highly non Gaussian, the machine learning approach delivers the best reconstruction with only a very slight shift of the curve's peak. An important characteristic of the ORCA and Wiener filter approach is that some values do not fall into the limits of a valid flux range making density field calculations impossible at those points without artificially limiting the range by clipping values lower than zero and larger than one.

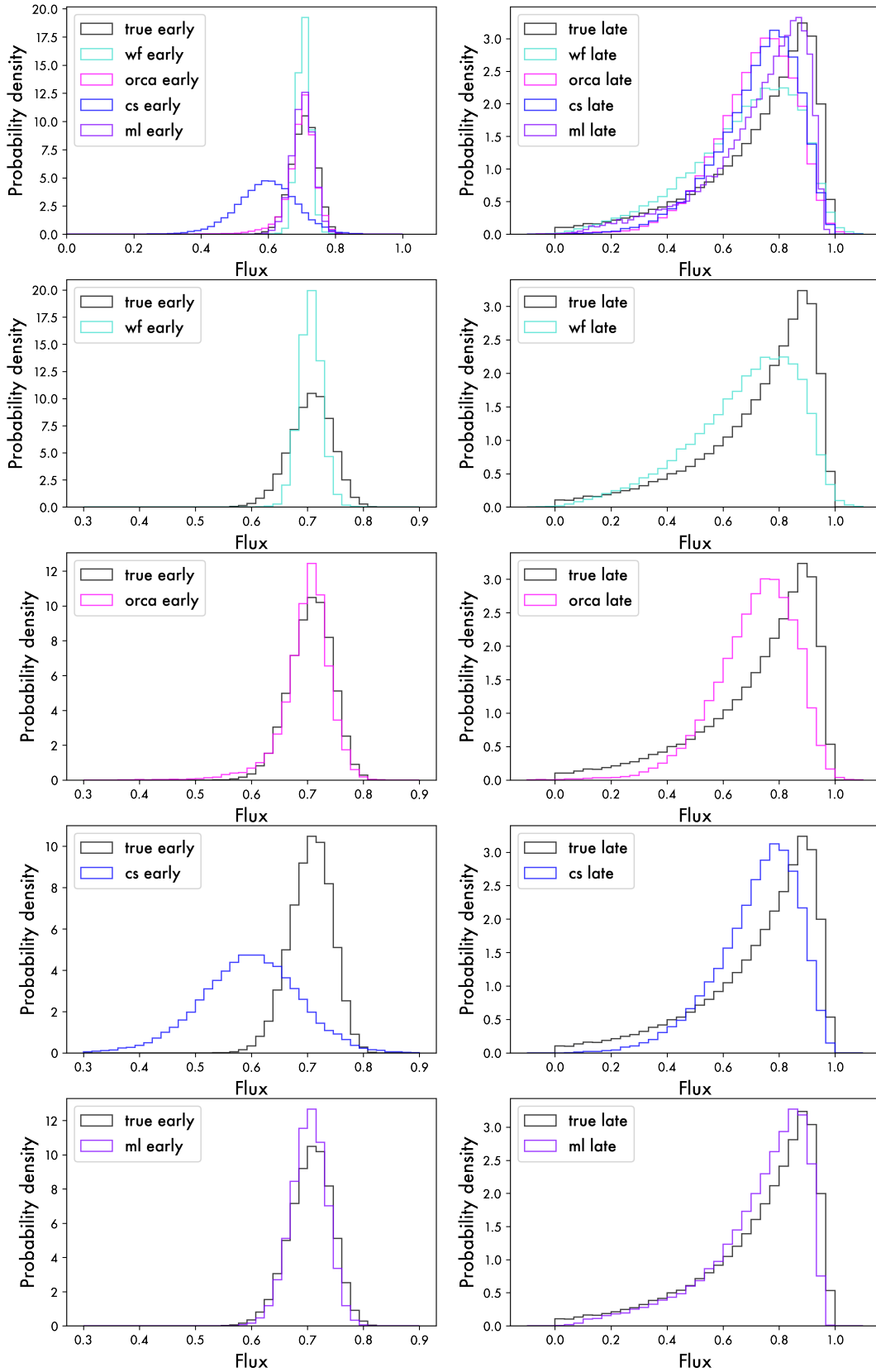


Figure 3.8.: Early and late universe flux PDFs. Compressed sensing clearly fails for early times. ORCA and machine learning reproduce the almost perfectly Gaussian PDF more closely than any of the early methods. For late times where the true PDF evidently becomes highly non Gaussian, the machine learning approach, which is also conditioned on the growth factor, delivers the best reconstruction with only a very slight shift of the curve's peak. Some values in the WF and ORCA reconstructions do not fall within the limits of a valid flux range.

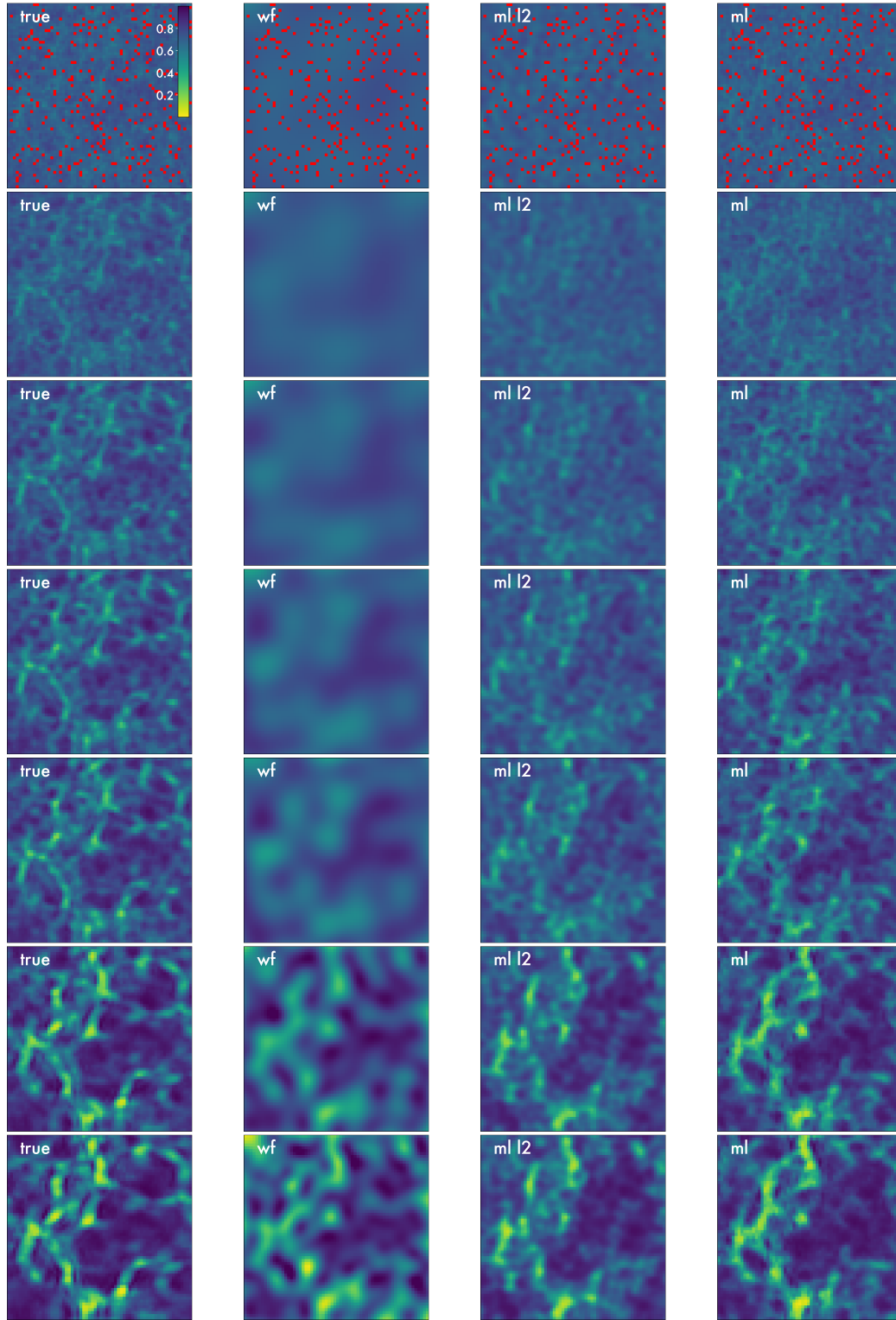


Figure 3.9.: Two-dimensional true fields and corresponding Wiener filter, and machine learning reconstructions ranging from early Gaussian to late non Gaussian flux fields displayed along the columns. The red dots in the first row represent the 327 given measurement points from which the reconstruction is made. Column three shows the machine learning reconstruction only utilizing Euclidean norm loss, whereas column four shows the reconstruction utilizing cGAN+ l_1 -loss. As expected, visually, l_2 is blurrier compared to the full cGAN+ l_1 -loss because there is no discriminator being trained to identify results that look unconvincing. However, in all reconstructions, l_2 is not as blurry as the Wiener filter, possibly because prior is implicitly given through training data.

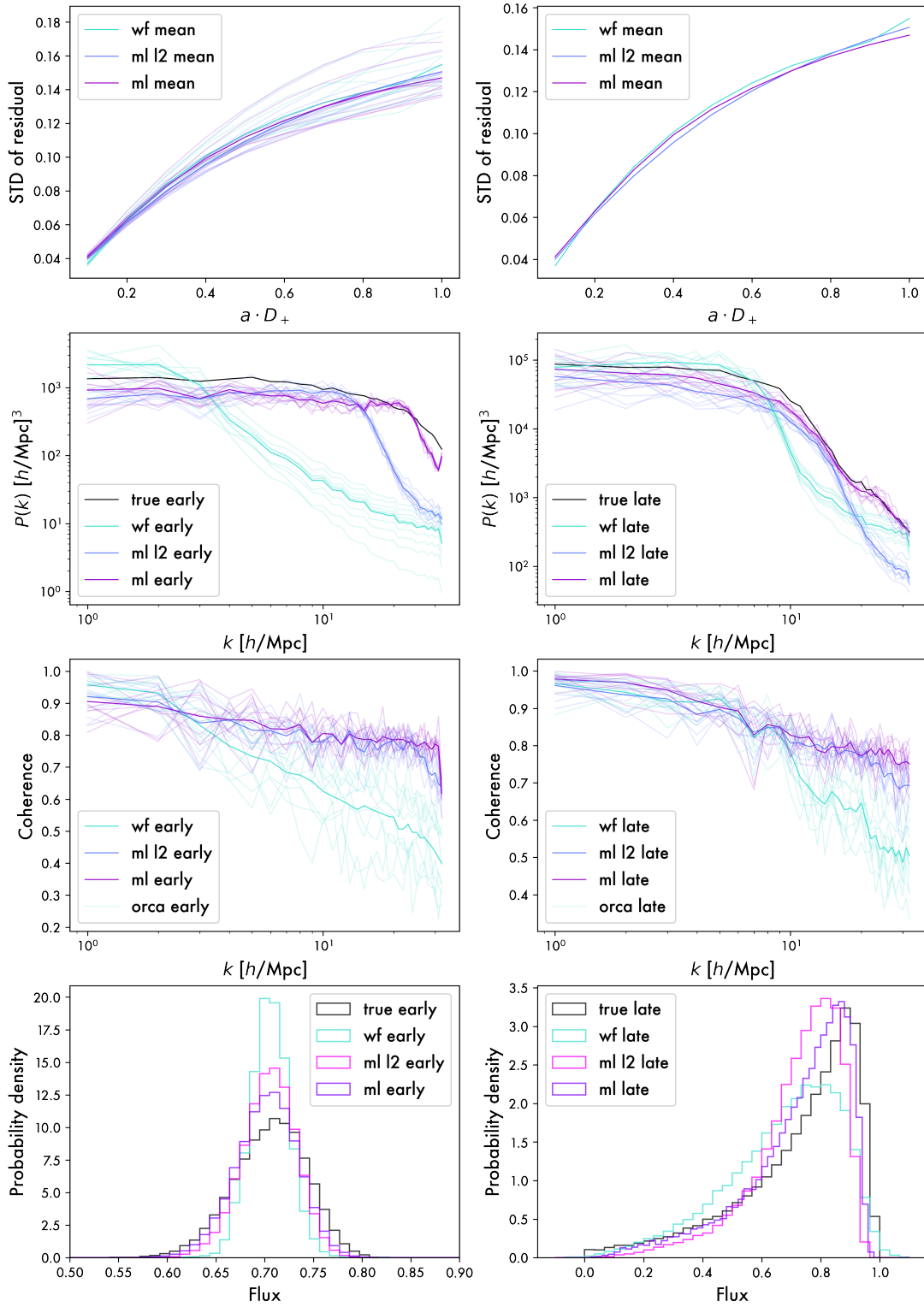


Figure 3.10.: STD of residual, power-spectra, cross-spectra and probability density functions comparing full cGAN+l1-loss with only l2-loss.

3.3. 3D late universe Lyman-alpha flux and density reconstructions

Now, I will discuss the three-dimensional case which is the concrete case that is relevant for potential future real applications on survey datasets. Adding the additional dimension results in available data being aligned along the line of sight skewers as opposed to being randomly spread out within the reconstruction volume. In the previous two-dimensional reconstructions the points were randomly distributed, similar to the way in which Lyman-alpha line of sight skewers would be distributed face on. For the power- and cross spectra, the probability density functions and the structure classification evaluations I averaged the results of ten realizations. As we are going to see, many of the characteristic features and qualities of each reconstruction method seen in the previous section are also valid for three-dimensional reconstructions.

In addition to the full machine learning reconstruction from a discrete set of data points, a different way of using this method is also included. In Figure 3.11 and Figure 3.12 the images labelled ml_{enh} are created by feeding the Wiener Filter reconstruction into the neural network during training as the input field. Visually, the Wiener filter field is greatly enhanced using this method, however, lack of knowledge of true data leads to some negative effects with regard to density field calculations and structure classification which will become obvious during the course of this chapter.

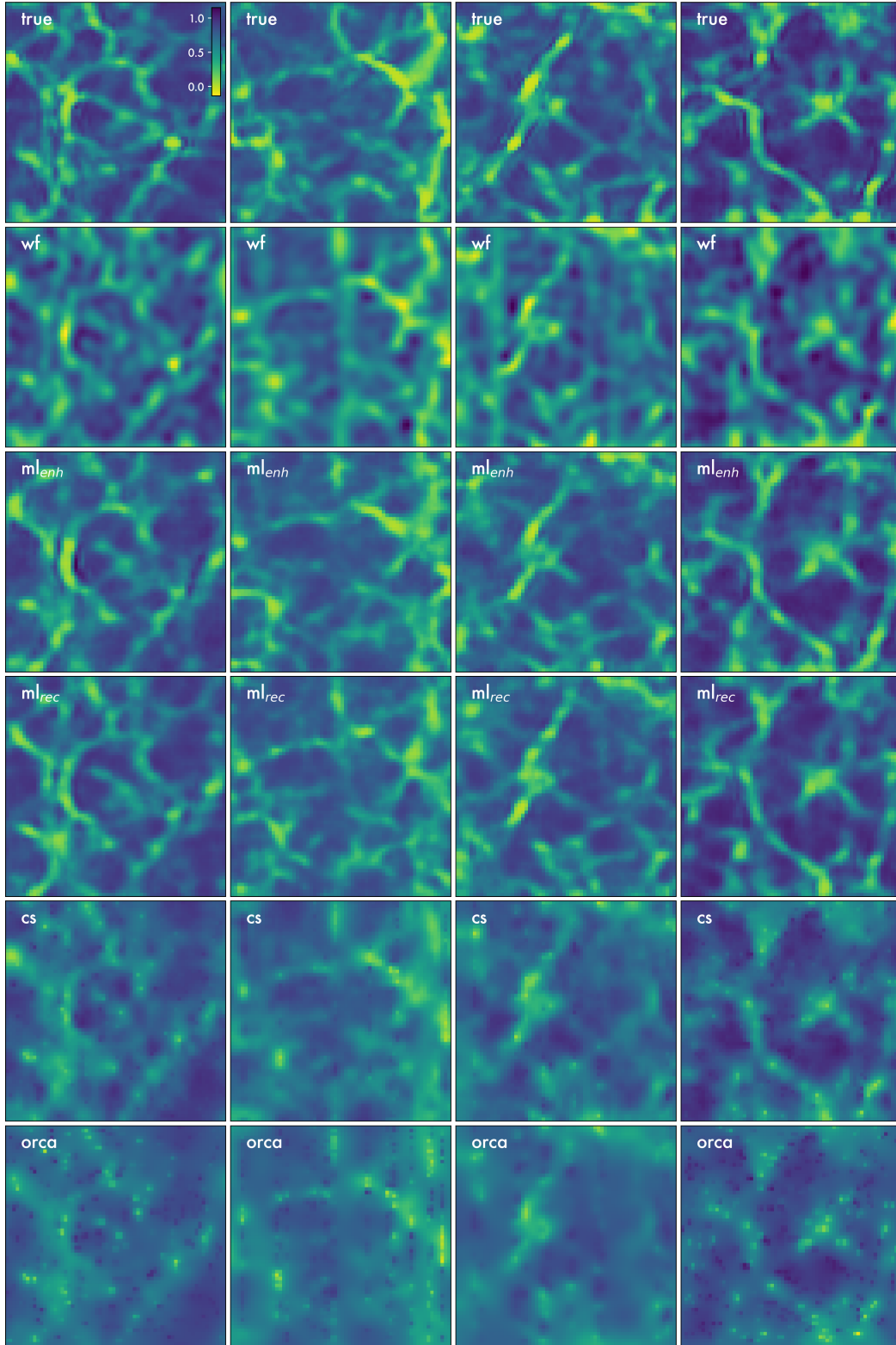


Figure 3.11.: 3D Ly- α flux field slices and corresponding WF, ORCA, compressed sensing and machine learning reconstruction slices shown face on (column one and four), from the side (column two), and from the top (third column). The reconstruction denoted as ml_{enh} enhances the Wiener Filter image with machine learning without ever seeing true data.

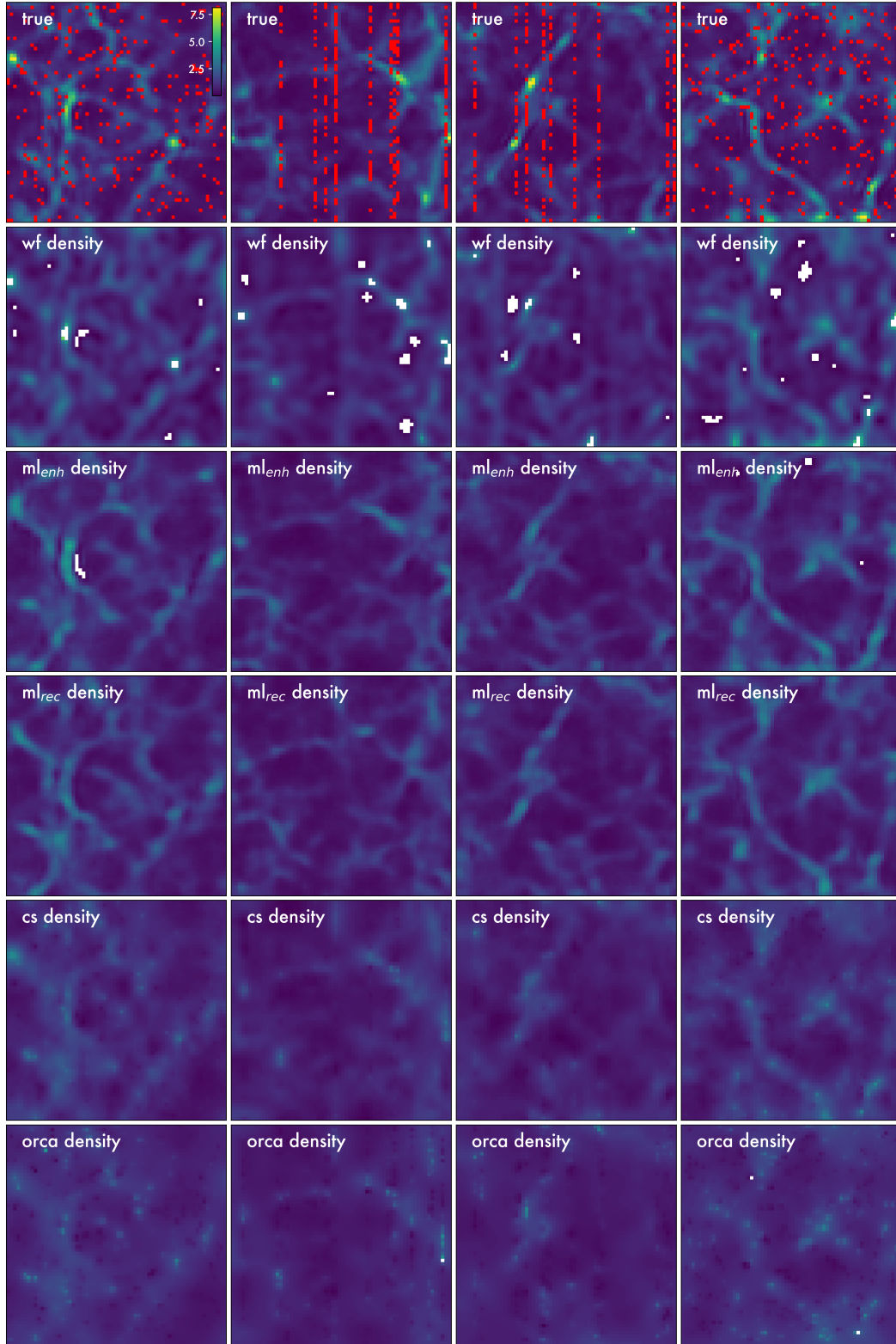


Figure 3.12.: 3D Ly- α density field slices and corresponding WF, ORCA, compressed sensing and machine learning density field slices calculated from the flux reconstructions from three different perspectives, face on (column one and four), from the side (column two), and from the top (third column). The white marks show areas where the flux wasn't in a valid range and the red ones show the points of given true data from which the reconstructions are made.

3.3.1. Power- and cross spectra

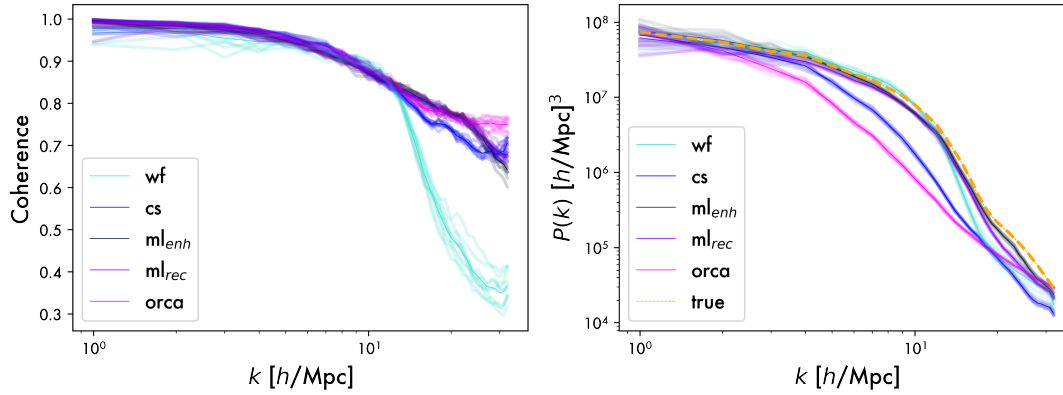


Figure 3.13.: This plot shows the cross spectra for each reconstruction method for a late non Gaussian Flux distribution on the left and the one on the right shows the corresponding power spectra.

The power and cross spectra displayed in Figure 3.13 give further insight into spatial correlation of the reconstructions with the true field. The plot on the left hand side shows cross spectra for each reconstruction method for a late non-Gaussian flux distribution and the one the right shows the corresponding power spectra. For large scales spatial coherence is good for the Wiener filter field, due to the blurry nature of the reconstruction, however, coherence drops significantly towards smaller scales. Compressed sensing, ORCA and machine learning reconstructions are almost equal in terms of coherence. The reason why ORCA produces high coherence on very small scales may be related to the visual discontinuity observed in the reconstruction, where, for example having a very bright single pixel increases the amplitude of a certain mode. Close inspection of the power spectra clearly shows that the true spectrum is best reproduced with the machine learning methods. From the flux field, the density field is estimated according to the fluctuating Gunn-Peterson approximation,

$$\delta_{rec} = \left(\frac{-\ln(F)}{0.35} \right)^{(1/1.56)}. \quad (3.1)$$

3.3.2. Probability density functions

In the flux probability density functions shown in Figure 3.14, similar patterns are observable as in the case of late non Gaussian 2D PDFs in Figure 3.8. The machine learning reconstruction has direct access to this information during the training phase which is evident in the quality of PDF reproduction. Just like in the 2D case, it also becomes apparent that some values in the Wiener filter and ORCA reconstructions do not fall within the limits of a valid flux range making density field calculations impossible at those points without artificial clipping. In Figure 3.12, the points where density calculation is impossible are represented with white pixels.

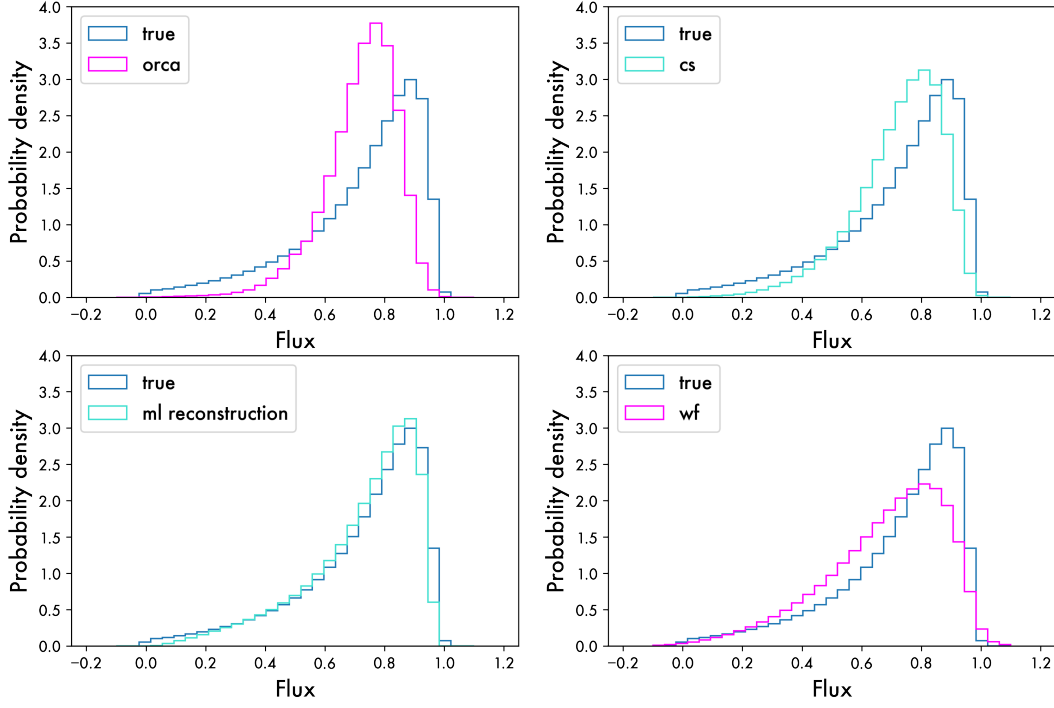


Figure 3.14.: Late non Gaussian universe 3D flux PDFs. The machine learning approach delivers the best reconstruction with only a very slight deviation from the true PDF. Some values in the WF and ORCA reconstructions do not fall within the limits of a valid flux range.

3.3.3. Structure classification

The structure classification method used in this evaluation is the Tidal Shear Tensor [FRHG⁺09, HPCD07] in which the deformation tensor¹,

$$D_{ij} = \frac{\partial^2 \Phi}{\partial x_i \partial x_j}, \quad (3.2)$$

where Φ is the underlying gravitational potential, is used to decide whether a structure is classified as a node, sheet, filament or void by calculating the deformation tensor's eigenvectors. Numerically, this is done in Fourier space using,

$$\tilde{D}_{ij} = \frac{k_i k_j}{k^2} \tilde{\delta}, \quad (3.3)$$

and inverse Fourier transforming $\tilde{D}_{ij}$ to get D_{ij} where the density field is denoted as $\tilde{\delta}$. The deformation tensor's eigenvectors determine the direction of net flow depending on whether, one, two or three eigenvalues are above a threshold λ . Zero eigenvalues above threshold classify a point as a void, one eigenvalue above λ classifies it as a sheet, two eigenvalues above λ classify it as a filament and three values above λ classify it as a node. However, the relationship between the density field $\tilde{\delta}$ and flux field is reversed, i.e. high flux corresponds to low density and vice versa meaning that, for instance, three eigenvalues above the threshold value λ classify the point as a void instead of node. λ is selected according to [HZK21] in a way that makes the voids occupy 22% of the total volume and before the calculations, Gaussian smoothing is applied to all fields with a kernel of $2h^{-1}$ Mpc. The accuracy of cosmic web classification can additionally be investigated with

¹the Hessian of the underlying gravitational potential Φ

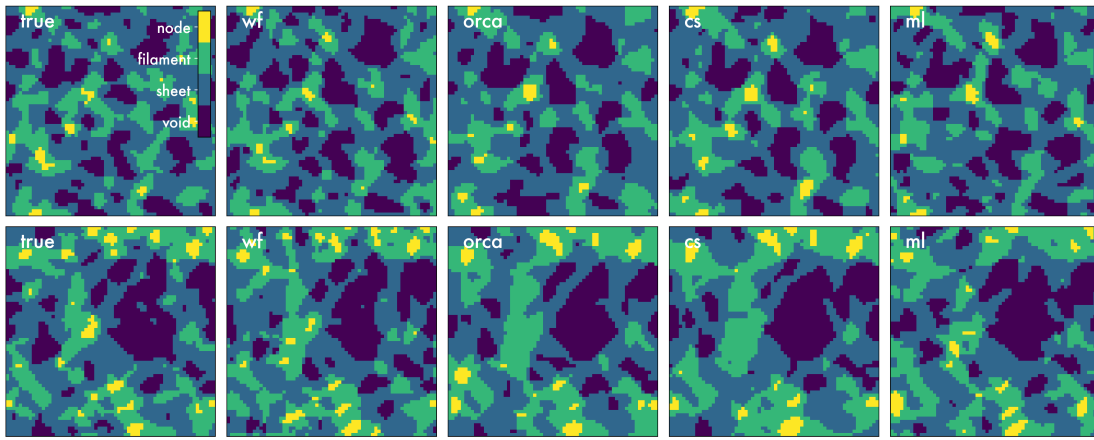


Figure 3.15.: Typical examples for slices of 3D segmentation maps. The corresponding confusion matrices in Figure 3.16 reflect the similar nature of these figures.

confusion matrices in Figure 3.16, where the fractions of correctly identified structures are displayed on the diagonal in percent whereas the off-diagonal elements show the fraction of misidentified structures, also in percent. Overall, compressed sensing is best at identifying nodes, filaments and sheets and machine learning is best at identifying voids. Generally speaking, all methods are similarly good in this regard. The code used for calculating the deformation tensor is based on [Akr17].

Wiener filter					ORCA				
	node	filament	sheet	void		node	filament	sheet	void
node	61.4	36.4	2.2	0.0	node	50.2	43.4	6.3	0.1
filament	10.8	73.3	15.7	0.2	filament	9.8	67.7	22.0	0.6
sheet	1.6	25.9	68.3	4.2	sheet	1.0	24.3	69.5	5.2
void	0.3	6.8	40.8	52.1	void	0.1	3.9	39.9	56.1

Compressed sensing					Machine learning reconstruction				
	node	filament	sheet	void		node	filament	sheet	void
node	62.2	34.1	3.6	0.0	node	50.7	43.8	5.5	0.1
filament	10.9	74.0	14.8	0.3	filament	10.6	70.0	19.1	0.4
sheet	0.9	25.1	70.8	3.2	sheet	1.2	23.5	70.1	5.2
void	0.1	3.4	39.6	57.0	void	0.1	3.6	36.6	59.7

Figure 3.16.: Confusion matrices showing the fractions of correctly identified structures on the diagonal in percent. The off-diagonal elements show the fractions of misidentified structures, also in percent. Overall, compressed sensing is best at identifying nodes, filaments and sheets and machine learning is best at identifying voids. Generally speaking, all methods are similarly good in this regard.

Table 3.2.: Table listing the fraction of structures identified as nodes, filaments, sheets and voids in percent for each reconstruction technique. Before the calculations Gaussian smoothing is applied to all fields with a kernel of $2h^{-1}$ Mpc. The threshold λ is chosen according to [LHC21].

Method	voids	sheets	filaments	nodes
true	22.0	49.5	25.3	3.1
wf	19.3	48.5	28.4	3.7
cs	20.4	49.2	26.9	3.4
ORCA	21.2	50.0	25.7	3.1
ml _{rec}	22.3	48.7	25.6	3.3
ml _{enh}	21.8	49.3	25.5	3.3

3.3.4. Computational cost analysis

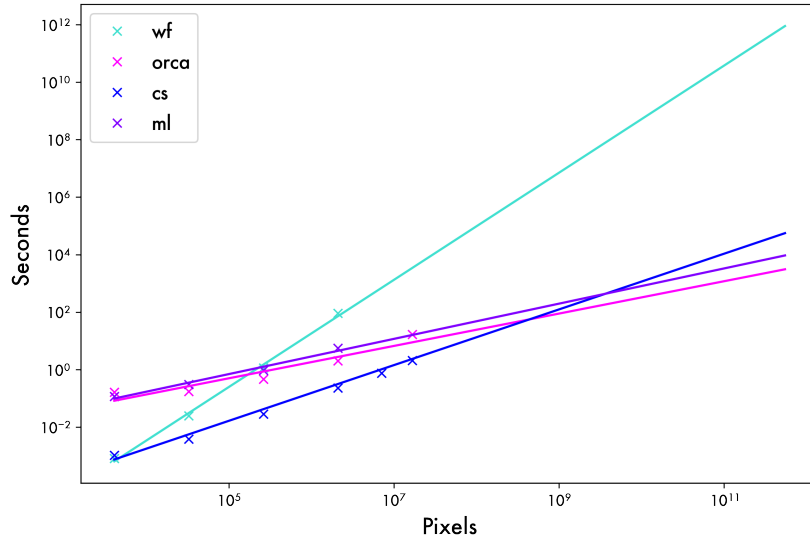


Figure 3.17.: Computational cost analysis. This log-log plot shows the time scaling in seconds for one iteration/training step as a function of pixel number. The Wiener filter equations are solved iteratively with dachshund whereas ORCA and the machine learning approaches greatly benefit from GPU acceleration. The slope of the fit is what really determines time scaling.

The final essential step in the evaluation of results is analysis of computational expense for each reconstruction technique and which hardware capabilities, if any, can be exploited. All of the coding, training and evaluation in this thesis was done on a 13 inch M1 Macbook Pro with 16 Gb RAM. My version of ORCA is written in Python within Tensorflow [ABC⁺16], an open source software library which enables programs to run on a computer's graphics processing unit dramatically decreasing running time as a consequence. The machine learning approach is also written within Tensorflow, where GPU utilization is also exploited. Compressed Sensing and dachshund on the other hand run fully on the CPU (not parallelized). In line with findings in [LHC21], this technique is around three times faster for a volume with 100^3 pixels, because of my significantly weaker GPU, however, I wasn't able to achieve a 10-100 fold speed increase as reported in [LHC21]. It's important to note that with increasing resolution, the number of required iterations/training steps may also increase which is why the figure shows the required time per iteration/training step in seconds. Finding the exact relationship was beyond the scope of this thesis.

3.3.5. Summary

All reconstruction techniques appear to have their benefits in certain aspects, however, only for the case of early universe Gaussian random field reconstructions does the Wiener Filter outperform all other methods on very large scales. In terms of reproduction of the power spectrum and PDF, machine learning is clearly ahead, primarily because it has direct access to these statistics during training. ORCA produces results that are spatially the most coherent with the true field and compressed sensing and machine learning outperform the Wiener filter by a similar margin. The drawback associated with the cGAN based method as discussed in chapter two is artificial fabrication of structures on scales that in reality cannot be resolved. However, the comparison between the full cGAN+ l_1 -loss and a version utilizing only Euclidean loss also clearly demonstrates the benefits of a discriminator, improvement of PDF, power- and cross-spectra without much impact on residual STD. In all regards except for residual STD for very early times, l_2 still outperforms the Wiener filter, which proves to be a simple way of utilizing machine learning for three-dimensional tomography without the problem of fictional structure fabrication as in the cGAN. For cosmic web classification, compressed sensing seems to be the best choice overall although machine learning leads to void identification with greatest accuracy. Table 3.3 summarizes some important quantities comparing residual standard deviations of true and reconstructed fields and the time scaling of their evaluation.

Table 3.3.: Table summarizing some important quantities with regard to reconstruction quality and computational cost. Residual standard deviation in data refers to the given data from which the reconstruction is made. The fraction of correctly identified structures is scaled by volume, meaning that nodes, for instance, have least impact on this quantity since they occupy the least amount of volume within the reconstructed field. The runtime scaling is apparently dominated by a constant overhead in compressed sensing, ORCA and machine learning since the values presented in this table are sub-linear.

Method	Residual STD in data	Residual STD	Correctly identified structures [%]	time scaling
wf	0.073	0.139	66.34	$\propto p^{1.86}$
cs	0.018	0.119	68.55	$\propto p^{0.97}$
ORCA	0.089	0.137	65.62	$\propto p^{0.56}$
ml _{rec}	0.08	0.127	67.12	$\propto p^{0.61}$
ml _{enh}	0.098	0.137	65.25	$\propto p^{0.61}$

4. Discussion and Future Perspectives

During the course of this thesis I successfully developed and applied two novel three-dimensional reconstruction techniques for cosmic web reconstruction purposes. One of these techniques is a machine learning method based on the conditional generative adversarial network "pix2pix" and the other is a signal reconstruction method utilizing the iterative shrinkage thresholding algorithm for solving underdetermined linear systems for sparse solutions in orthogonal basis. Additionally I attempted to replicate the work done by Li, Horowitz and Cai in [LHC21] and although it is unclear whether my implementation exactly matches their method, the results are very promising, especially in terms of reconstruction quality compared to computational expense for datasets larger than $\approx 10^{10}$ pixels. The machine learning method produces the most compelling results visually with high spatial coherence and reproduction of Lyman-alpha forest statistics such as power spectrum and probability density function, however, Lyman-alpha statistics can only be learned if full Lyman-alpha maps are available beforehand meaning that this approach goes hand in hand with simulations, otherwise no training dataset is available. If there are errors in assumptions for simulations, those errors are going to be propagated into the reconstruction. Compressed sensing and ORCA don't have this problem to such an extent. The only assumptions being made are orthogonal basis solution sparsity in compressed sensing and a Gaussian smoothing operator in the ORCA loss function. The Wiener Filter also contains covariance matrices of Gaussian form in its equations. The Gaussian elements contained within the latter two methods have the effect of smoothing the reconstruction in a way where sub covariance length structures cannot be resolved whereas machine learning attempts to guess those structures based on data it receives during training so caution is advised for usage even though there is no negative impact during structure identification, for instance. In the context of Lyman-alpha tomography, the drawback associated with the cGAN based method as discussed in chapter two is artificial fabrication of structures on scales that in reality cannot be resolved. However, the comparison between the full cGAN+ l_1 -loss and a version utilizing only Euclidean loss also clearly demonstrated the benefits of a discriminator, improvement of PDF, power- and cross-spectra without much impact on residual STD. l_2 U-Net outperforms the Wiener filter in all regards except for residual STD for very early times, which has proven to be a simple way of using machine learning for three-dimensional tomography.

The machine learning approach also opens up an entirely new world containing a variety of ways in which it can be used for scientific purposes. One way, which is also teased in this thesis, is image enhancing by feeding, for example, compressed sensing reconstructions into the neural network in addition to the real data in a second channel. This method might dramatically increase the reconstruction quality visually, with all of the benefits that compressed sensing has with regard to cosmic web classification. By utilizing the fast iterative shrinkage thresholding algorithm [BT09], the faster variant of the algorithm used in this thesis which converges with $O(t^{-2})$, the reconstructed fields could also be produced much quicker.

Overall I have shown that, if Lyman-alpha skewer data is known, and simulation data is good, machine learning is a viable option for Lyman-alpha cosmic web tomography especially regarding spatial coherence, PDF and power spectrum reproduction, and void classification in the cosmic web. Additionally, I have also demonstrated that my three-dimensional implementation of compressed sensing is best in terms of residual standard deviation between true and reconstructed field and yields best overall results at classifying structures, particularly nodes, filaments and sheets. Overall, the novel methods outperform the iterative Dachshund implementation of the Wiener filter.

In total my work can be summarized as follows.

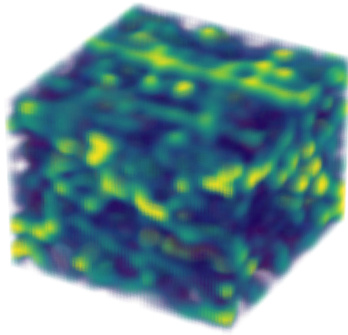
- I have developed two new methods based on work done in [Kli20] and [IZZE17] for conducting three-dimensional Lyman-alpha forest tomography and have shown through thorough testing that these techniques present a viable alternative to the previously existing methods.
- My compressed sensing, ORCA and machine learning implementations are capable of reproducing Lyman-alpha statistics such as the probability density function and power spectrum to a greater degree with higher spatial coherence.
- Cosmic web classification is greatly improved by compressed sensing overall and particularly for voids through machine learning (up to 8% more accurate void identification compared to Wiener filtering).
- All Wiener filter alternatives reconstruct the cosmic web at least as good as the Wiener filter at lesser computational expense for large datasets ($> 10^6$ pixels).

Some improvements are still to be made.

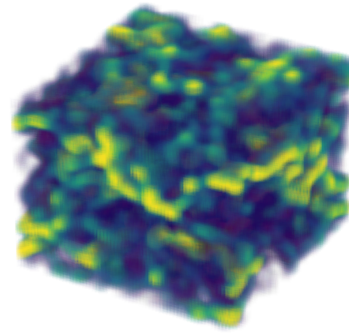
- Adding realistic individual signal-to-noise ratios to each individual line of sight skewer like in [LHC21] would be a great addition to the mock survey set-up.
- The machine learning method doesn't produce a highly stochastic output that can cover the full entropy of the conditional distributions it models. Testing different machine learning techniques such as explicit model learning of a gradient log-probability distribution with efficient sampling via annealing as was done by the authors of [RLJ⁺22] seems to be the right approach for solving the aforementioned issue.
- Further testing regarding the machine learning loss function is required since this is an aspect that has fallen short due to time constraints.
- The compressed sensing method can further be sped up by implementing the so called fast iterative shrinkage algorithm [BT09].
- Inserting the compressed sensing reconstruction as a second channel in addition to the true data as the input for the neural network might be a simple way to increase the machine learning reconstruction quality.
- Unfortunately, time constraints have also prevented a more thorough investigation of time scaling on an actually powerful GPU to test the limits of each method, computationally.

I'm ending this thesis with a visually enhanced three-dimensional rendering of the density field reconstructions for each reconstruction technique in Figure 4.1.

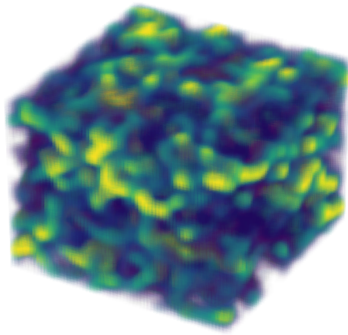
Wiener filter



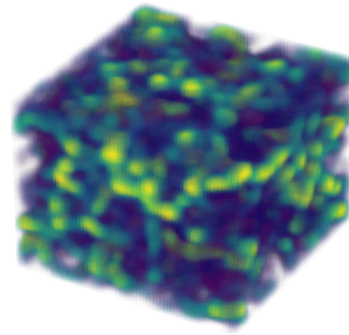
True density



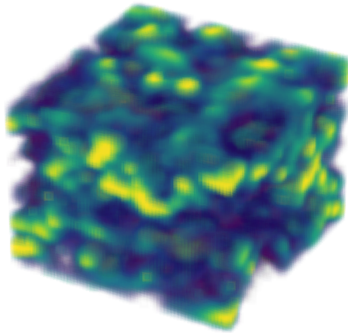
Machine learning WF enhancing



Machine learning



Compressed sensing



ORCA

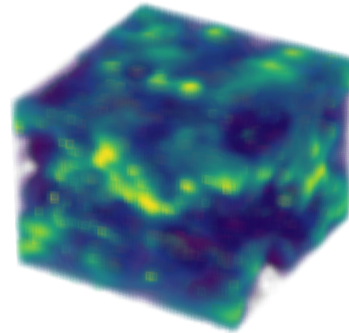


Figure 4.1.: Visually enhanced three-dimensional renderings of density field reconstructions for each reconstruction technique. Yellow colors indicate high density areas, whereas blue colors mark areas where density is low.

List of Figures

1.1. Comparison: Zeldovich and PPT	10
1.2. First ever Lyman-alpha forest spectrum	12
1.3. BOSS survey	13
1.4. CLAMATO survey	13
2.1. Visualization of 3D reconstruction objective	15
2.2. Sparse solutions from l_1 -norm.	20
2.3. Artificial neural network architecture	24
2.4. Architecture flowchart: generator	27
2.5. Architecture flowchart: discriminator	28
3.1. Gaussian and non Gaussian flux PDF	29
3.2. Gaussian and non Gaussian density PDF	30
3.3. WF parameter analysis for the 3D case	31
3.4. 2D reconstruction results comparison	32
3.5. 2D STD of residual comparison	33
3.6. 2D power spectrum comparison	33
3.7. 2D cross spectrum comparison	34
3.8. Early and late universe flux PDFs	35
3.9. 2D l_2 flux field reconstruction results comparison	36
3.10. 2D l_2 statistics comparisons	37
3.11. 3D flux reconstruction results comparison	39
3.12. 3D density reconstruction results comparison	40
3.13. 3D cross and power spectra	41
3.14. 3D flux PDFS	42
3.15. 3D segmentation maps	43
3.16. Confusion matrices	43
3.17. Computational cost analysis	44
4.1. 3D density field renderings	49

Bibliography

- [ABC⁺16] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. {TensorFlow}: a system for {Large-Scale} machine learning. In 12th USENIX symposium on operating systems design and implementation (OSDI 16), pages 265–283, 2016.
- [Akr17] Akrolewski. Igm tomography eigenvectors. https://github.com/akrolewski/IGM_tomography_eigenvectors, 2017. Accessed: 22-09-2022.
- [AZH⁺21] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A Fadhel, M. Al-Amidie, and L. Farhan. Review of deep learning: Concepts, cnn architectures, challenges, applications, future directions. Journal of big Data, 8(1):1–74, 2021.
- [BB19] T. Bezdan and N. Bacanin. Convolutional neural network layers and architectures. pages 445–451, 01 2019.
- [BCGS02] F. Bernardeau, S. Colombi, E. Gaztanaga, and R. Scoccimarro. Large-scale structure of the universe and cosmological perturbation theory. Physics reports, 367(1-3):1–248, 2002.
- [BCHJ94] F. Bouchet, S. Colombi, E. Hivon, and R. Juszkiewicz. Perturbative lagrangian approach to gravitational instability. arXiv preprint astro-ph/9406013, 1994.
- [BFW⁺01] R. H. Becker, X. Fan, R. L. White, M. A. Strauss, V. K. Narayanan, R. H. Lupton, J. E. Gunn, J. Annis, N. A. Bahcall, J. Brinkmann, et al. Evidence for reionization at $z \sim 6$: Detection of a gunn-peterson trough in $az = 6.28$ quasar. The Astronomical Journal, 122(6):2850, 2001.
- [BG71] J. N. Bahcall and S. Goldsmith. On the absorption-line spectrum of 4c 05.34. The Astrophysical Journal, 170:17, 1971.
- [BGF17] F. Bre, J. Gimenez, and V. Fachinotti. Prediction of wind pressure coefficients on building surfaces using artificial neural networks. Energy and Buildings, 158, 2017.
- [BK22] S. L. Brunton and J. N. Kutz. Data-driven science and engineering: Machine learning, dynamical systems, and control. Cambridge University Press, 2022.
- [BLNZ95] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited memory algorithm for bound constrained optimization. SIAM Journal on scientific computing, 16(5):1190–1208, 1995.
- [Bra85] R. H. Brandenberger. Quantum field theory methods and inflationary universe models. Reviews of Modern Physics, 57(1):1–60, 1985.
- [BT09] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. SIAM journal on imaging sciences, 2(1):183–202, 2009.

- [CAE20] M. Domenico Cirillo, D. Abramian, and A. Eklund. Vox2vox: 3d-gan for brain tumour segmentation. In International MICCAI Brainlesion Workshop, pages 274–284. Springer, 2020.
- [Car20a] C. Caramanis. 2.1 convex sets: Definitions. <https://youtu.be/hNRtGmrgr2g>, 2020. Accessed: 26-07-2022.
- [Car20b] C. Caramanis. 2.3 convex functions. <https://youtu.be/FNiJJL-1xVw>, 2020. Accessed: 26-07-2022.
- [Car20c] C. Caramanis. 2.5 optimality conditions for convex optimization. <https://youtu.be/oS4kCcF0JGg>, 2020. Accessed: 26-07-2022.
- [Car20d] C. Caramanis. 5.1 proximal and projected gradient descent. https://youtu.be/m73Fy_rHV0A, 2020. Accessed: 26-07-2022.
- [Car20e] C. Caramanis. 5.4 ista and fista. <https://youtu.be/JRerBpNggN0>, 2020. Accessed: 26-07-2022.
- [CO17] B. W. Carroll and D. A. Ostlie. An introduction to modern astrophysics. Cambridge University Press, 2017.
- [CWKH98] R. A. C. Croft, D. H. Weinberg, N. Katz, and L. Hernquist. Recovery of the power spectrum of mass fluctuations from observations of the $\text{Ly}\alpha$ forest. The Astrophysical Journal, 495(1):44–62, 1998.
- [DEFW85] M. Davis, G. Efstathiou, C. S. Frenk, and S. D. M. White. The evolution of large-scale structure in a universe dominated by cold dark matter. The Astrophysical Journal, 292:371–394, 1985.
- [Dir05] P. A. M. Dirac. The lagrangian in quantum mechanics. In Feynman’s Thesis—A New Approach To Quantum Theory, pages 111–119. World Scientific, 2005.
- [DJS96] J. E. Dennis Jr and R. B. Schnabel. Numerical methods for unconstrained optimization and nonlinear equations. SIAM, 1996.
- [DMD⁺22] C. T. Donnan, D. J. McLeod, J. S. Dunlop, R. J. McLure, A. C. Carnall, R. Begley, F. Cullen, M. L. Hamadouche, R. A. A. Bowler, H. J. McCracken, et al. The evolution of the galaxy uv luminosity function at redshifts $z \sim 8-15$ from deep jwst and ground-based near-infrared imaging. arXiv preprint arXiv:2207.12356, 2022.
- [Dod03] S. Dodelson. Modern cosmology. Elsevier, 2003.
- [Dor70] A. G. Doroshkevich. Spatial structure of perturbations and origin of galactic rotation in fluctuation theory. Astrophysics, 6(4):320–330, 1970.
- [Fey48] R. P. Feynman. Space-time approach to non-relativistic quantum mechanics. Rev. Mod. Phys., 20:367–387, 1948.
- [Fle13] R. Fletcher. Practical methods of optimization. John Wiley & Sons, 2013.
- [FR13] S. Foucart and H. Rauhut. A Mathematical Introduction to Compressive Sensing. 2013.
- [FR18] A. Font-Ribera. Cosmology with the lyman-alpha forest: challenges and opportunities. https://cosmology.lbl.gov/talks/FontRibera_18.pdf, 2018. Accessed: 04-08-2022.

- [FRHG⁺09] J. E. Forero-Romero, Y. Hoffman, S. Gottlöber, A. Klypin, and G. Yepes. A dynamical classification of the cosmic web. Monthly Notices of the Royal Astronomical Society, 396(3):1815–1824, 2009.
- [Fry84] J. N. Fry. The galaxy correlation hierarchy in perturbation theory. The Astrophysical Journal, 279:499–510, 1984.
- [GBB11] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In G. Gordon, D. Dunson, and M. Dudík, editors, Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, volume 15 of Proceedings of Machine Learning Research, pages 315–323, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR.
- [GGRW86] M. H. Goroff, B. Grinstein, S. Rey, and M. B. Wise. Coupling of modes of cosmological mass density fluctuations. The Astrophysical Journal, 311:6–14, 1986.
- [Goo16] Ian Goodfellow. Nips 2016 tutorial: Generative adversarial networks. arXiv preprint arXiv:1701.00160, 2016.
- [GP65] J. E. Gunn and B. A. Peterson. On the Density of Neutral Hydrogen in Intergalactic Space. The Astrophysical Journal, 142:1633–1636, 1965.
- [GP82] A. H. Guth and S. Y. Pi. Fluctuations in the New Inflationary Universe. Physical Review Letters, 49(15):1110–1113, 1982.
- [GPAM⁺14] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks, 2014.
- [Gut81] A. H. Guth. Inflationary universe: A possible solution to the horizon and flatness problems. Physical Review D, 23(2):347–356, 1981.
- [Har70] E. R. Harrison. Fluctuations at the threshold of classical cosmology. Physical review D, 1(10):2726, 1970.
- [HK70] A. E. Hoerl and R. W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. Technometrics, 12(1):55–67, 1970.
- [HLA⁺21] B. Horowitz, K. Lee, M. Ata, T. Müller, A. Krolewski, J. X. Prochaska, J. F. Hennawi, M. White, D. Schlegel, R. M. Rich, et al. Second data release of the cosmos lyman-alpha mapping and tomographic observation: The first 3d maps of the large-scale cosmic web at $2.05 < z < 2.55$. arXiv preprint arXiv:2109.09660, 2021.
- [HNY97] L. Hui, Y. G. Nickolay, and Z. Yu. The statistics of density peaks and the column density distribution of the $\text{Ly}\alpha$ forest. The Astrophysical Journal, 486(2):599–622, 1997.
- [HPCD07] O. Hahn, C. Porciani, C. M. Carollo, and A. Dekel. Properties of dark matter haloes in clusters, filaments, sheets and voids. Monthly Notices of the Royal Astronomical Society, 375(2):489–499, 2007.
- [HR21] O. Hahn and C. Rampf. Cosmic structures: Theory, statistics and numerics. https://moodle.univie.ac.at/pluginfile.php/15370770/course/section/2182703/cosmic_structures180621.pdf, 2021. Accessed: 17-08-2022.
- [HSVdW13] J. Hidding, S. F. Shandarin, and R. Van de Weygaert. The zel'dovich approximation: key to understanding cosmic web complexity. Monthly Notices of the Royal Astronomical Society, 437(4):3442–3472, dec 2013.

- [HZK21] B. Horowitz, K. Zhang, B. Lee, and R. Kooistra. TARDIS. II. synergistic density reconstruction from $\text{Ly}\alpha$ forest and spectroscopic galaxy surveys with applications to protoclusters and the cosmic web. The Astrophysical Journal, 906(2):110, 2021.
- [IZZE17] P. Isola, J. Zhu, T. Zhou, and A. Efros. Image-to-image translation with conditional adversarial networks. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 1125–1134, 2017.
- [Jeo10] D. Jeong. Cosmology with high redshift ($z > 1$) galaxy surveys. PhD thesis, University of Texas at Austin, 2010.
- [JMW95] B. Jain, H. J. Mo, and Simon D. M. White. The evolution of correlation functions and power spectra in gravitational clustering. Monthly Notices of the Royal Astronomical Society, 276(1):L25–L29, 1995.
- [KB14] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.
- [Kli20] M. Kliesch. Cs image recovery demo. https://github.com/MartKl/CS_image_recovery_demo, 2020. Accessed: 22-09-2022.
- [LHC21] Z. Li, B. Horowitz, and Z. Cai. Improved $\text{Ly}\alpha$ tomography using optimized reconstruction with constraints on absorption (orca). The Astrophysical Journal, 916(1):20, 2021.
- [LKW⁺18a] K. Lee, A. Krolewski, M. White, D. Schlegel, P. E. Nugent, J. F. Hennawi, T. Müller, R. Pan, J. X. Prochaska, A. Font-Ribera, et al. 3d tomographic map of the redshift 2.2 - 2.5 $\text{Ly}\alpha$ forest in the cosmos field from our 2014-2015 data. <https://clamato.lbl.gov/photos/>, 2018. Accessed: 04-08-2022.
- [LKW⁺18b] K. Lee, A. Krolewski, M. White, D. Schlegel, P. E. Nugent, J. F. Hennawi, T. Müller, R. Pan, J. X. Prochaska, A. Font-Ribera, et al. First data release of the cosmos $\text{Ly}\alpha$ mapping and tomography observations: 3d $\text{Ly}\alpha$ forest tomography at $2.05 < z < 2.55$. The Astrophysical Journal Supplement Series, 237(2):31, 2018.
- [LN89] D. C. Liu and J. Nocedal. On the limited memory bfgs method for large scale optimization. Mathematical programming, 45(1):503–528, 1989.
- [LSLP16] F. Lanusse, J. Starck, A. Leonard, and S. Pires. High resolution weak lensing mass mapping combining shear and flexion. Astronomy & Astrophysics, 591:A2, 2016.
- [Lyn71] R. Lynds. The absorption-line spectrum of 4c 05.34. The Astrophysical Journal, 164:L73, 1971.
- [MSS92] N. Makino, M. Sasaki, and Y. Suto. Analytic approach to the perturbative expansion of nonlinear gravitational fluctuations in cosmological density and velocity fields. Physical Review D, 46(2):585, 1992.
- [Muk05] V. Mukhanov. Physical foundations of cosmology. Cambridge university press, 2005.
- [MVdBW10] H. Mo, F. Van den Bosch, and S. White. Galaxy formation and evolution. Cambridge University Press, 2010.
- [NH10] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In International Conference for Machine Learning, 2010.

- [ON15] K. O’Shea and R. Nash. An introduction to convolutional neural networks. arXiv preprint arXiv:1511.08458, 2015.
- [Pee81] P. J. E. Peebles. Primeval adiabatic perturbations-constraints from the mass distribution. The Astrophysical Journal, 248:885–897, 1981.
- [PHJL20] N. Porqueres, O. Hahn, J. Jasche, and G. Lavaux. A hierarchical field-level inference approach to reconstruction from sparse lyman- α forest data. Astronomy & Astrophysics, 642:A139, 2020.
- [PTVP21] W Pan, C Torres-Verdín, and M. J. Pyrcz. Stochastic pix2pix: a new machine learning method for geophysical and well conditioning of rule-based channel reservoir models. Natural Resources Research, 30(2):1319–1345, 2021.
- [PW65] A. A. Penzias and R. W. Wilson. A measurement of excess antenna temperature at 4080 mc/s. The Astrophysical Journal, 142:419–421, 1965.
- [PY70] P. J. E. Peebles and J. Yu. Primeval adiabatic perturbation in an expanding universe. The Astrophysical Journal, 162:815, 1970.
- [RAB⁺04] S. Rawlings, F. B. Abdalla, S. L. Bridle, C. A. Blake, C. M. Baugh, L. J. Greenhill, and J. M. Van Der Hulst. Galaxy evolution, cosmology and dark energy with the square kilometer array. New Astronomy Reviews, 48(11-12):1013–1027, 2004.
- [RFB15] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In International Conference on Medical image computing and computer-assisted intervention, pages 234–241. Springer, 2015.
- [RLJ⁺22] B. Remy, F. Lanusse, N. Jeffrey, J. Starck, K. Osato, and T. Schrabback. Probabilistic mass mapping with neural score estimation. arXiv preprint arXiv:2201.05561, 2022.
- [Rud16] S. Ruder. An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747, 2016.
- [Ryd17] B. Ryden. Introduction to cosmology. Cambridge University Press, 2017.
- [SBK⁺92] G. F. Smoot, C. L. Bennett, A. Kogut, E. L. Wright, J. Aymon, N. W. Boggess, E. S. Cheng, G. de Amici, S. Gulkis, M. G. Hauser, G. Hinshaw, P. D. Jackson, M. Janssen, E. Kaita, T. Kelsall, P. Keegstra, C. Lineweaver, K. Loewenstein, P. Lubin, J. Mather, S. S. Meyer, S. H. Moseley, T. Murdock, L. Rokke, R. F. Silverberg, L. Tenorio, R. Weiss, and D. T. Wilkinson. Structure in the COBE Differential Microwave Radiometer First-Year Maps. The Astrophysical Journal, 396:L1, 1992.
- [Sch06] P. Schneider. Extragalactic astronomy and cosmology: an introduction, volume 146. Springer, 2006.
- [Sel12] Uroš Seljak. Bias, redshift space distortions and primordial nongaussianity of nonlinear transformations: application to ly- α forest. Journal of Cosmology and Astroparticle Physics, 2012(03):004, 2012.
- [SHK⁺14] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1):1929–1958, 2014.
- [SIK⁺13] A. Slosar, V. Iršič, D. Kirkby, S. Bailey, T. Delubac, J. Rich, É. Aubourg, J. E. Bautista, V. Bhardwaj, M. Blomqvist, et al. Measurement of baryon

- acoustic oscillations in the lyman- α forest fluctuations in boss data release 9. Journal of Cosmology and Astroparticle Physics, 2013(04):026, 2013.
- [Sol57] R. J. Solomonoff. An inductive inference machine. In IRE Convention Record, Section on Information Theory, volume 2, pages 56–62. Institute of Radio Engineers New York, 1957.
- [SS91] Y. Suto and M. Sasaki. Quasilinear theory of cosmological self-gravitating systems. Physical Review Letters, 66(3):264, 1991.
- [Sta16] C. W. Stark. Dachshund. <https://github.com/caseywstark/dachshund>, 2016. Accessed: 22-09-2022.
- [STIM18] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry. How does batch normalization help optimization? Advances in neural information processing systems, 31, 2018.
- [SWLH15] C. W. Stark, M. White, K. Lee, and J. F. Hennawi. Protocluster discovery in tomographic ly α forest flux maps. Monthly Notices of the Royal Astronomical Society, 453(1):311–327, 2015.
- [The21] T. Theuns. Physical cosmology. <http://icc.dur.ac.uk/~tt/Lectures/UA/L4/cosmology.pdf>, 2021. Accessed: 22-09-2022.
- [Tib96] R. Tibshirani. Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society: Series B (Methodological), 58(1):267–288, 1996.
- [Tib15] R. Tibshirani. Convex optimization lecture 8. <https://www.stat.cmu.edu/~ryantibs/convexopt-F15/scribes/08-prox-grad-scribed.pdf>, 2015. Accessed: 26-09-2022.
- [UGS18] K. Usman, H. Gunawan, and A. Suksmono. Compressive sensing reconstruction algorithm using l1-norm minimization via l2-norm minimization. International Journal on Electrical Engineering and Informatics, 10:37–50, 2018.
- [URGH19] Cora Uhlemann, Cornelius Rampf, Mateja Gosenca, and Oliver Hahn. Semiclassical path to cosmic large-scale structure. Physical Review D, 99(8):083524, 2019.
- [Vis83] E. T. Vishniac. Why weakly non-linear effects are small in a zero-pressure cosmology. Monthly Notices of the Royal Astronomical Society, 203(2):345–349, 1983.
- [Zel70] Y. B. Zeldovich. Gravitational instability: An approximate theory for large density perturbations. Astronomy and astrophysics, 5:84–89, 1970.
- [Zel72] Y. B. Zeldovich. A hypothesis, unifying the structure and the entropy of the Universe. mnras, 160:1P, 1972.
- [ZHFL94] S. Zaroubi, Y. Hoffman, KB. Fisher, and O. Lahav. Wiener reconstruction of the large scale structure. arXiv preprint astro-ph/9410080, 1994.

Appendix

A. Convex Sets and Functions

Def: A set $C \subseteq \mathbb{R}^n$, is convex if,

$$\lambda x_1 + (1 - \lambda)x_2 \in C \quad (\text{A.1})$$

is true $\forall x_1, x_2 \in C$ and $\forall \lambda \in [0, 1]$ [Car20a].

Similarly a function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is convex if,

$$f(\lambda x_1 + (1 - \lambda)x_2) \leq \lambda f(x_1) + (1 - \lambda)f(x_2), \quad (\text{A.2})$$

is true $\forall x_1, x_2 \in C$ and $\forall \lambda \in [0, 1]$ [Car20b].

Additionally, the linear approximation of a convex function at a point is always smaller than the function value at that point, i.e. a differentiable function f is convex if,

$$f(\mathbf{x}_2) \geq f(\mathbf{x}_1) + (\nabla f(\mathbf{x}_1))^T(\mathbf{x}_2 - \mathbf{x}_1) \quad (\text{A.3})$$

is true [Car20b].

A twice differentiable function f is also convex if the Hessian,

$$\nabla^2 f(\mathbf{x}), \quad (\text{A.4})$$

is positive semi definite meaning that for all $\mathbf{z} \in \mathbb{R}^n$,

$$\mathbf{z}^T(\nabla^2 f(\mathbf{x}))\mathbf{z} \geq 0 \quad (\text{A.5})$$

is true, or in other words, all eigenvalues of the Hessian of f are greater or equal to zero [Car20b].

However, assuming that a function is not differentiable, it can still be convex if

$$\forall \mathbf{x}, \exists \mathbf{g} \text{ s.t. } f(\mathbf{x}_2) \geq f(\mathbf{x}_1) + \mathbf{g}^T(\mathbf{x}_2 - \mathbf{x}_1). \quad (\text{A.6})$$

In this case $\mathbf{g}$ is called a *subgradient* and the set of subgradients of f at a point $\mathbf{x}$, is called the *subdifferential* of f at $\mathbf{x}$ [Car20b].

B. Conditions for Convex Optimization

Classically, $\mathbf{s}$ is an optimal solution to a minimization problem if [Car20c],

$$\nabla f(\mathbf{s}) = 0. \tag{B.1}$$

In terms of convexity (appendix section A) in combination with equation (B.1) the definition of the optimal solution is [Car20c],

$$f(\mathbf{x}) \geq f(\mathbf{s}) + (\nabla f(\mathbf{s}))^T(\mathbf{x} - \mathbf{s}) \quad \forall \mathbf{x}. \tag{B.2}$$

If $\nabla f(\mathbf{s}) = 0$, then [Car20c],

$$f(\mathbf{x}) \geq f(\mathbf{s}) \quad \forall \mathbf{x}. \tag{B.3}$$

C. Python implementations

Python implementation of the Wiener Filter in two dimensions (full code)

Here, n represents the number of pixels along one axis. So the total image has a resolution of n^2 pixels. xd and yd are the coordinates of the points where data is given.

```

1 n          = 64
2 d          = true[xd,yd]
3 N          = n**2
4 M          = 327

```

At this point a grid is defined.

```

1 l          = 1.0
2 gridstep  = l/n
3 ii        = np.mgrid[0:n, 0:n]
4 xx        = gridstep * (ii)
5 x         = np.reshape(xx[0],N)
6 y         = np.reshape(xx[1],N)
7
8 xdgrid    = xd/n
9 ydgrid    = yd/n

```

The Wiener Filter are set:

```

1 L          = correlation_length
2 noise     = noise_val
3 a         = amplitude

```

Calculate the data-data covariance matrix efficiently by exploiting numpys broadcasting capabilities.

```

1 Cdd       = np.zeros((M,M))
2
3 dx        = np.reshape(xdgrid, (M,1)) - np.reshape(xdgrid, (1,M))
4 dy        = np.reshape(ydgrid, (M,1)) - np.reshape(ydgrid, (1,M))
5
6 Cdd       = a*np.exp( -(dx)**2-(dy)**2)/(2*(L**2)) )

```

Calculate noise covariance and solve linear system of equations instead of doing a full matrix inversion.

```

1 N_cov     = np.diag(np.ones(M))*noise**2
2
3 u         = np.linalg.solve(Cdd + N_cov,d)

```

Calculate the map-data covariance matrix.

```

1 Cmd       = np.zeros((N,M))
2
3 dx        = np.reshape(x, (N,1)) - np.reshape(xdgrid, (1,M))
4 dy        = np.reshape(y, (N,1)) - np.reshape(ydgrid, (1,M))
5
6 Cmd       = a*np.exp( -(dx)**2-(dy)**2)/(2*(L**2)) )
7
8 F         = np.matmul(Cmd,u)

```

Python Implementation of the Wiener Filter in three dimensions (full code)

The three dimensional version is analogous to the two dimensional one.

```

1 n          = 64
2 d          = true[xd,yd,zd]
3 N          = n**3
4 M          = 655*32

```

At this point a grid is defined.

```

1 l          = 1.0
2 gridstep   = l/n
3 ii         = np.mgrid[0:n, 0:n, 0:n]
4 xx         = gridstep * (ii)
5 x          = np.reshape(xx[0],N)
6 y          = np.reshape(xx[1],N)
7 z          = np.reshape(xx[2],N)
8
9
10 xdgrid     = xd/n
11 ydgrid     = yd/n
12 zdgrid     = zd/n

```

The Wiener Filter are set:

```

1 L          = correlation_length
2 noise      = noise_val
3 a          = amplitude

```

Calculate the data-data covariance matrix efficiently by exploiting numpys broadcasting capabilities.

```

1 Cdd        = np.zeros((M,M))
2
3 dx         = np.reshape(xdgrid, (M,1)) - np.reshape(xdgrid, (1,M))
4 dy         = np.reshape(ydgrid, (M,1)) - np.reshape(ydgrid, (1,M))
5 dz         = np.reshape(zdgrid, (M,1)) - np.reshape(zdgrid, (1,M))
6
7
8 Cdd        = a*np.exp( -(dx)**2-(dy)**2-(dz)**2)/(2*(L**2))

```

Calculate noise covariance and solve linear system of equations instead of doing a full matrix inversion.

```

1 N_cov      = np.diag(np.ones(M))*noise**2
2
3 u          = np.linalg.solve(Cdd + N_cov,d)

```

Calculate the map-data covariance matrix.

```

1 Cmd        = np.zeros((N,M))
2
3 dx         = np.reshape(x, (N,1)) - np.reshape(xdgrid, (1,M))
4 dy         = np.reshape(y, (N,1)) - np.reshape(ydgrid, (1,M))
5 dz         = np.reshape(z, (N,1)) - np.reshape(zdgrid, (1,M))
6
7
8 Cmd        = a*np.exp( -(dx)**2-(dy)**2-(dz)**2)/(2*(L**2))
9
10 F          = np.matmul(Cmd,u)

```

Python Implementation of ORCA in two dimensions (full code)

```

1 d_orca    = true[xd.astype(int),yd.astype(int)]
2
3 point_locs = np.stack((xd/n,yd/n),axis=1)
4
5 @tf.function
6 def loss(s):
7     x_ref_min    = [0., 0.]
8     x_ref_max    = [1, 1]
9
10    @tf.function
11    def tf_interp(y_ref):
12        y_ref      = tf.reshape(y_ref,[n,n])
13        interp      = tfp.math.batch_interp_regular_nd_grid(x=point_locs,
14        x_ref_min=x_ref_min, x_ref_max=x_ref_max, y_ref=y_ref, axis=-2)
15        return interp
16
17    @tf.function
18    def tf_gauss(A):
19        A = tf.cast(tf.reshape(A,[n,n]),tf.float32)
20        A = A.numpy()
21        Ag = gaussian_filter(A, sigma = 0.3*(n*L/2 - 1) + 0.8, mode='wrap')
22        return Ag
23
24    return tfp.math.value_and_gradient(
25    lambda s: 5*tf.math.reduce_sum(tf.math.squared_difference(tf_gauss(s),
26    tf.reshape(s,[n,n])))+
27    tf.math.reduce_sum(tf.math.squared_difference(tf_interp(s),d_orca)
28    /0.2**2)+
29    0.3 * tf.math.reduce_sum(tf.clip_by_value(s,1,np.inf))+
30    0.025 * tf.math.reduce_sum(tf.clip_by_value(s,0,0.55))
31    , s)
32
33 @tf.function
34 def opt_with_lbfgs():
35    return tfp.optimizer.lbfgs_minimize(loss, initial_position=F_init,
36    tolerance=1e-12, max_iterations=100, max_line_search_iterations=50)
37
38 mlf = run(opt_with_lbfgs)
39
40 orca = mlf.position

```

Python Implementation of ORCA in three dimensions (full code)

```

1 d_orca    = true[xd.astype(int),yd.astype(int),zd.astype(int)]
2
3 point_locs = np.stack((xd/n,yd/n,zd/n),axis=1)
4
5 @tf.function
6 def loss(s):
7     x_ref_min    = [0., 0., 0.]
8     x_ref_max    = [1, 1, 1]
9
10    @tf.function
11    def tf_interp(y_ref):
12        y_ref      = tf.reshape(y_ref,[n,n,n])
13        interp      = tfp.math.batch_interp_regular_nd_grid(x=point_locs,
14        x_ref_min=x_ref_min, x_ref_max=x_ref_max, y_ref=y_ref, axis=-3)
15        return interp
16
17    @tf.function
18    def tf_gauss(A):
19        A = tf.cast(tf.reshape(A,[n,n,n]),tf.float32)
20        A = A.numpy()
21        Ag = gaussian_filter(A, sigma = 0.3*(n*L/2 - 1) + 0.8, mode='wrap')
22        return Ag
23
24    return tfp.math.value_and_gradient(
25    lambda s: 5*tf.math.reduce_sum(tf.math.squared_difference(tf_gauss(s),
26    tf.reshape(s,[n,n,n])))+
27    tf.math.reduce_sum(tf.math.squared_difference(tf_interp(s),d_orca)
28    /0.2**2)+
29    0.3 * tf.math.reduce_sum(tf.clip_by_value(s,1,np.inf))+
30    0.025 * tf.math.reduce_sum(tf.clip_by_value(s,0,0.55))
31    , s)
32
33 @tf.function
34 def opt_with_lbfgs():
35     return tfp.optimizer.lbfgs_minimize(loss, initial_position=F_init,
36     tolerance=1e-12, max_iterations=100, max_line_search_iterations=50)
37
38 mlf = run(opt_with_lbfgs)
39
40 orca = mlf.position

```

Python Implementation of ISTA for Compressed Sensing in two dimensions (full code)

In the following section X_{orig} represent the original image that is to be reconstructed.

```
1 shape      = X_orig
2 D          = np.prod(shape)
3 X_noisy_raw = X_orig + np.random.normal(0, 10, size=shape)
```

Now, a function is defined to clip values in the RGB range [0, 255]

```
1 def clip_vals(x, low, high):
2     return np.clip(x, low, high)
3 X_noisy = clip_vals(X_noisy_raw, low=0, high=255)
```

Creating a subsampled image Y,

```
1 p          = 0.02
2 S          = int(p * D)
3 sub_inds   = np.random.choice(D, S, replace=False)
4
5 mask       = np.zeros(D, dtype=bool)
6 mask[sub_inds] = True
7
8 Y          = np.zeros(shape)
9 Y.flat[mask] = X_noisy.flat[mask]
```

where p is the fraction of pixels to be kept, D is the total number of pixels in the image and S is the total number of subsampled pixels.

```
1 class DCT:
2     def __init__(self, shape):
3         self.shape = shape
4     def forward(self, image):
5         image_t = fftpack.dct(fftpack.dct(image, norm='ortho', axis=0),
6                               norm='ortho', axis=1)
7         return image_t.reshape(-1)
8     def inverse(self, image_t):
9         image_t = image_t.reshape(self.shape)
10        return fftpack.idct(fftpack.idct(image_t, norm='ortho', axis=0),
11                             norm='ortho', axis=1)
```

In the previous section, the forward and inverse discrete cosine transform is defined.

```
1 class DWT:
2     def __init__(self, shape, wavelet='db4', level=3, do_weighting=True):
3         self.shape      = shape
4         self.wavelet     = wavelet
5         self.level      = level
6         self.mat_shapes = []
7
8         if do_weighting is False:
9             self.weights = np.ones(3 * self.level + 1)
10        else:
11            weights      = np.linspace(1, .2, self.level)
12            weights      = np.repeat(weights, 3)
13            weights      = np.hstack([[20], weights])
14            self.weights = weights
15        def forward(self, image):
16            coeffs = pywt.wavedec2(image,
17                                   wavelet=self.wavelet,
18                                   level=self.level)
19            mat_list = [coeffs[0]]
20            mat_list += [c_mat for c in coeffs[1:] for c_mat in c]
21            self.mat_shapes = list(map(np.shape, mat_list))
```

```

22
23     vec_list = [np.asarray(mat_list[j]).flatten() * self.weights[j] for
                j in range(3 * self.level + 1)]
24
25     return np.concatenate(vec_list)
26 def inverse(self, wavelet_vector):
27     if len(self.mat_shapes) == 0:
28         print("Call 'forward' first to set the shapes of coefficient
                matrices!")
29         return []
30
31     vec_shapes      = list(map(np.prod, self.mat_shapes))
32     split_indices   = np.cumsum(vec_shapes)
33     vec_list_weighted = np.split(wavelet_vector, split_indices)
34
35     vec_list = [vec_list_weighted[j] / self.weights[j]
                 for j in range(3 * self.level + 1)]
36
37
38     coeffs = [np.reshape(vec_list[0], self.mat_shapes[0])]
39             for j in range(self.level):
40         triple = vec_list[3 * j + 1:3 * (j + 1) + 1]
41         triple = [np.reshape(triple[i], self.mat_shapes[1 + 3 * j + i])
                   for i in range(3)]
42         coeffs = coeffs + [tuple(triple)]
43
44     return pywt.waverec2(coeffs, wavelet=self.wavelet)

```

The same is done for the discrete wavelet transform. The following part shows the actual ISTA algorithm.

```

1 def thresh(v, gamma):
2     return pywt.threshold(v, gamma, mode='soft')
3 def ista_step(Psi, Xt, Y, Gamma, mask, eta=1.0):
4     CXt      = np.zeros(Psi.shape)
5     CXt.flat[mask] = Xt.flat[mask]
6     grad      = 2 * (CXt - Y)
7     norm_grad  = linalg.norm(grad.flat)
8     Psi_T_update = Psi.forward(Xt - eta * grad)
9     theta_new   = thresh(Psi_T_update, Gamma)
10    support     = theta_new != 0
11    return Psi.inverse(theta_new), norm_grad, support

```

The final section contains iteration loop.

```

1 psi_op_dwt = DWT(shape)
2 psi_op_dct = DCT(shape)
3
4 N_it      = 5000
5 eta       = 0.75
6
7 gammas = np.linspace(25, 4, N_it, endpoint=True)
8 X_rec  = Y
9
10 for it in tqdm(range(N_it)):
11     X_rec, norm_grad, support = ista_step(Psi=psi_op_dwt, Xt=X_rec, Y=Y,
        Gamma=gammas[it], mask=mask, eta=eta)
12     X_rec = clip_vals(X_rec, low=0, high=255)
13     X_rec, norm_grad, support = ista_step(Psi=psi_op_dct, Xt=X_rec, Y=Y,
        Gamma=gammas[it], mask=mask, eta=eta)
14     X_rec = clip_vals(X_rec, low=0, high=255)

```


Python Implementation of ISTA for Compressed Sensing in three dimensions (full code)

For the three dimensional case, the only parts that change are the basis transformations.

```

1 class DCT:
2     def __init__(self, shape):
3         self.shape = shape
4     def forward(self, image):
5         image_t = fftpack.dct(fftpack.dct(fftpack.dct(image, norm='ortho',
6         axis=0), norm='ortho', axis=1), norm='ortho', axis=2)
7         return image_t.reshape(-1)
8     def inverse(self, image_t):
9         image_t = image_t.reshape(self.shape)
10        return fftpack.idct(fftpack.idct(fftpack.idct(image_t, norm='ortho',
11        , axis=0), norm='ortho', axis=1), norm='ortho', axis=2)
12
13 class DWT:
14     def __init__(self, shape, wavelet='db4', level=3, do_weighting=True):
15         self.shape = shape
16         self.wavelet = wavelet
17         self.level = level
18         self.mat_shapes = []
19         if do_weighting is False:
20             self.weights = np.ones(7 * self.level + 1)
21         else:
22             weights = np.linspace(1, .2, self.level)
23             weights = np.repeat(weights, 7)
24             weights = np.hstack([[20], weights]) # insert first
25         coefficient
26             self.weights = weights
27     def forward(self, image):
28         coeffs = pywt.wavedecn(image, wavelet=self.wavelet, level=self.
29         level)
30         mat_list = [coeffs[0]]
31         dict_arr = ["aad", "ada", "daa", "dda", "dad", "add", "ddd"]
32         for i in range(1, self.level + 1):
33             for j in dict_arr:
34                 mat_list.append(coeffs[i][j])
35             self.mat_shapes = list(map(np.shape, mat_list))
36             vec_list = [np.asarray(mat_list[j]).flatten() * self.weights[j] for
37             j in range(7 * self.level + 1)]
38             return np.concatenate(vec_list)
39
40     def inverse(self, wavelet_vector):
41         if len(self.mat_shapes) == 0:
42             print("Call 'forward' first to set the shapes of coefficient
43             matrices!")
44             return []
45         vec_shapes = list(map(np.prod, self.mat_shapes))
46         split_indices = np.cumsum(vec_shapes)
47         vec_list_weighted = np.split(wavelet_vector, split_indices)
48         vec_list = [vec_list_weighted[j] / self.weights[j] for j in range(7
49         * self.level + 1)]
50         dict_arr = ["aad", "ada", "daa", "dda", "dad", "add", "ddd"]
51         coeffs = [np.reshape(vec_list[0], self.mat_shapes[0])]
52         counter = 1
53         for i in range(1, self.level + 1):
54             coeffs.append(dict())
55             for j in dict_arr:
56                 coeffs[i][j] = np.reshape(vec_list[counter], self.
57                 mat_shapes[counter])
58             counter += 1
59         return pywt.waverecn(coeffs, wavelet=self.wavelet)

```

Python Implementation of pix2pix based cGAN within Tensorflow in two dimensions (full code)

The downsampling consists of a 2D convolution with a stride of two after with batch normalization is applied followed by a Leaky ReLU layer.

```

1 def downsample(filters, size, apply_batchnorm=True):
2     initializer = tf.random_normal_initializer(0., 0.02)
3
4     result = tf.keras.Sequential()
5     result.add(
6         tf.keras.layers.Conv2D(filters, size, strides=2, padding='same',
7                                 kernel_initializer=initializer, use_bias=False)
8     )
9
10    if apply_batchnorm:
11        result.add(tf.keras.layers.BatchNormalization())
12
13    result.add(tf.keras.layers.LeakyReLU())
14    return result

```

The upsampling consists of random noise field followed by a 2D deconvolution, batch normalization, dropout and ReLU layer.

```

1 def upsample(filters, size, apply_dropout=False):
2     initializer = tf.random_normal_initializer(0., 0.02)
3
4     result = tf.keras.Sequential()
5     result.add(
6         tf.keras.layers.Conv2DTranspose(filters, size, strides=2,
7                                           padding='same',
8                                           kernel_initializer=initializer,
9                                           use_bias=False))
10
11    result.add(tf.keras.layers.BatchNormalization())
12
13    if apply_dropout:
14        result.add(tf.keras.layers.Dropout(0.5))
15
16    result.add(tf.keras.layers.ReLU())
17    return result

```

The generator downsamples and upsamples the image after which the skip connections are established.

```

1 def Generator():
2     inputs = tf.keras.layers.Input(shape=[64, 64, 2])
3
4     down_stack = [
5         downsample(64, 4, apply_batchnorm=False),
6         downsample(128, 4),
7         downsample(256, 4),
8         downsample(512, 4),
9         downsample(512, 4),
10        downsample(512, 4),
11    ]
12    up_stack = [
13        upsample(512, 4, apply_dropout=True),
14        upsample(512, 4),
15        upsample(256, 4),
16        upsample(128, 4),
17        upsample(64, 4),
18    ]
19

```

```

20     initializer = tf.random_normal_initializer(0., 0.02)
21     last = tf.keras.layers.Conv2DTranspose(OUTPUT_CHANNELS, 4,
22                                           strides=2,
23                                           padding='same',
24                                           kernel_initializer=initializer,
25                                           activation='tanh')
26
27     x = inputs
28
29     skips = []
30     for down in down_stack:
31         x = down(x)
32         skips.append(x)
33
34     skips = reversed(skips[:-1])
35
36     for up, skip in zip(up_stack, skips):
37         x = up(x)
38         x = tf.keras.layers.Concatenate()([x, skip])
39
40     x = last(x)
41
42     return tf.keras.Model(inputs=inputs, outputs=x)
43
44 def generator_loss(disc_generated_output, gen_output, target):
45     gan_loss = loss_object(tf.ones_like(disc_generated_output),
46                             disc_generated_output)
47
48     l1_loss = tf.reduce_mean(tf.abs(target - gen_output))
49
50     total_gen_loss = gan_loss + (LAMBDA * l1_loss)
51
52     return total_gen_loss, gan_loss, l1_loss
53
54 def Discriminator():
55     initializer = tf.random_normal_initializer(0., 0.02)
56
57     inp = tf.keras.layers.Input(shape=[64, 64, 2], name='input_image')
58     tar = tf.keras.layers.Input(shape=[64, 64, 2], name='target_image')
59
60     x = tf.keras.layers.concatenate([inp, tar])
61
62     down1 = downsample(64, 4, False)(x)
63     down2 = downsample(128, 4)(down1)
64     down3 = downsample(256, 4)(down2)
65
66     zero_pad1 = tf.keras.layers.ZeroPadding2D()(down3)
67     conv = tf.keras.layers.Conv2D(512, 4, strides=1,
68                                   kernel_initializer=initializer,
69                                   use_bias=False)(zero_pad1)
70
71     batchnorm1 = tf.keras.layers.BatchNormalization()(conv)
72
73     leaky_relu = tf.keras.layers.LeakyReLU()(batchnorm1)
74
75     zero_pad2 = tf.keras.layers.ZeroPadding2D()(leaky_relu)
76
77     last = tf.keras.layers.Conv2D(1, 4, strides=1,
78                                   kernel_initializer=initializer)(zero_pad2)
79
80     return tf.keras.Model(inputs=[inp, tar], outputs=last)
81
82

```

```

83 def discriminator_loss(disc_real_output, disc_generated_output):
84     real_loss = loss_object(tf.ones_like(disc_real_output), disc_real_output)
85     generated_loss = loss_object(tf.zeros_like(disc_generated_output),
86                                     disc_generated_output)
87
88     total_disc_loss = real_loss + generated_loss
89     return total_disc_loss
90
91 @tf.function
92 def train_step(input_image, target, step):
93     with tf.GradientTape() as gen_tape, tf.GradientTape() as disc_tape:
94         gen_output = generator(input_image, training=True)
95
96         disc_real_output = discriminator([input_image, target], training=True)
97         disc_generated_output = discriminator([input_image, gen_output],
98                                             training=True)
99
100         gen_total_loss, gen_gan_loss, gen_l1_loss = generator_loss(
101             disc_generated_output, gen_output, target)
102         disc_loss = discriminator_loss(disc_real_output, disc_generated_output)
103
104         generator_gradients = gen_tape.gradient(gen_total_loss,
105                                             generator.trainable_variables)
106         discriminator_gradients = disc_tape.gradient(disc_loss,
107                                                     discriminator.
108                                                         trainable_variables)
109
110         generator_optimizer.apply_gradients(zip(generator_gradients,
111                                             generator.trainable_variables))
112         discriminator_optimizer.apply_gradients(zip(discriminator_gradients,
113                                                    discriminator.
114                                                        trainable_variables))

```

The fit function consists of the training steps and saving checkpoints.

```

1 def fit(train_ds, test_ds, steps):
2
3     train_step(input_image, target, step)
4
5     if (step + 1) % 5000 == 0:
6         checkpoint.save(file_prefix=checkpoint_prefix)

```

Finally the fit function is called and training is initialized.

```

1 fit(train_dataset, test_dataset, steps=45000)

```

The reason for two channels in the two dimensional case is an extra layer that contains the D_+ value corresponding to a time progression through the history of the universe. The network learns not only what a field should look like but also what a field should look like at a certain time in the past.

For the three dimensional "pix2pix"-based cGAN the convolutions and deconvolutions have to be three dimensional, the generator and discriminator are also modified accordingly.

[illegible]

```

58
59 x = inputs
60
61 skips = []
62 for down in down_stack:
63     x = down(x)
64     skips.append(x)
65
66 skips = reversed(skips[:-1])
67
68 for up, skip in zip(up_stack, skips):
69     x = up(x)
70     x = tf.keras.layers.Concatenate()([x, skip])
71
72 x = last(x)
73
74 return tf.keras.Model(inputs=inputs, outputs=x)
75
76 def Discriminator():
77     initializer = tf.random_normal_initializer(0., 0.02)
78
79     inp = tf.keras.layers.Input(shape=[64, 64, 64, 1], name='input_image')
80     tar = tf.keras.layers.Input(shape=[64, 64, 64, 1], name='target_image')
81
82     x = tf.keras.layers.concatenate([inp, tar])
83
84     down1 = downsample(64, 4, False)(x)
85     down2 = downsample(128, 4)(down1)
86     down3 = downsample(256, 4)(down2)
87
88     zero_pad1 = tf.keras.layers.ZeroPadding3D()(down3)
89     conv      = tf.keras.layers.Conv3D(512, 4, strides=1,
90                                         kernel_initializer=initializer,
91                                         use_bias=False)(zero_pad1)
92
93     batchnorm1 = tf.keras.layers.BatchNormalization(axis=2)(conv)
94
95     leaky_relu = tf.keras.layers.LeakyReLU()(batchnorm1)
96
97     zero_pad2 = tf.keras.layers.ZeroPadding3D()(leaky_relu)
98
99     last = tf.keras.layers.Conv3D(1, 4, strides=1,
100                                   kernel_initializer=initializer)(zero_pad2)
101
102     return tf.keras.Model(inputs=[inp, tar], outputs=last)

```


Danksagung

An dieser Stelle möchte ich mich bei Oliver Hahn und Florian List für ihre Unterstützung bei dieser Masterarbeit herzlich bedanken.