UNIVERSIDAD DE SONORA
División de Ciencias Exactas y Naturales
Departamento de Investigación en Física

PECULIAR VELOCITY ESTIMATION FROM SUNYAEV-ZEL’DOVICH EFFECT SIMULATED SIGNAL USING DEEP LEARNING

Submitted to the graduate degree program in Physics in partial fulfillment of the requirements for the Degree of Master of Science (Physics)

By
Edgar Martín Salazar Canizales

Director
Hume A. Feldman

Hermosillo, Sonora, 7th of August 2020

Sponsored by *Consejo Nacional de Ciencia y Tecnología* through the graduate studies scholarship program 2018-2020.

All figures and text were generated and compiled by the author with LaTeX free typsetting system.

Magneticum and SMAC are owned by Klaus Dolag at the *Max Planck Gesellschaft für Astrophysik*, Garching, Germany.

July 2020
Printed in Mexico

Obtained Products

This project was designed and initiated during the eight-week **KU-Sonora Summer Research Program in Global Perspective**[1] in the summer of 2019 at The University of Kansas in Lawrence, USA. Sponsored in jointly efforts by *Universidad de Sonora*, Sonora State Government, KU Provost Office, KU CLAS Dean Office and KU Department of Physics & Astronomy.

The present thesis is one of the products obtained from the collaborative work with Y. Wang[2] and N. S. Ramachandra[3,4]. An article derived from this collaboration titled "Peculiar Velocity Estimation from Kinematic SZ Effect using Deep Neural Networks" is ready for submission to the Monthly Notices of the Royal Astronomical Society at the elaboration date of this document.

[1] More at https://isp.ku.edu/sonora-research-program

[2] Department of Physics & Astronomy, University of Kansas, Lawrence, KS 66045, USA.

[3] High Energy Physics Division, Argonne National Laboratory, Lemont, IL 60439, USA.

[4] Kavli Institute for Cosmological Physics, University of Chicago, Chicago, IL 60637, USA.

To my parents, sisters, nana and Ari. Thank you for all of your support along the way.

Acknowledgments

To Prof. Feldman my thesis advisor at The University of Kansas, for letting this thesis to be my own work. To Y. Wang for allowing me to collaborate in a free environment whilst pushing me to develop my independence.

To the University of Kansas Physics Department for providing access to HPC facilities operated by, and the staff of, the University of Kansas Center for Research Computing during the realization of my project.

To the University of Kansas ISS office for being so attentive in providing human and material resources that allowed me to start my work on this project during 2019's summer internship. Margaret, Mindy and Aaron, your kindness and hospitality meant a big deal to me.

To CONACYT for funding my studies through the graduate studies scholarship program 2018-2020. The economic certainty and stability allowed me to become a full-time student and successfully obtain my Master of Science degree.

To the University of Sonora for being a second home during my bachelors and masters programs. I want to thank all my professors that devoted to my education in physics. Specially to professors García, Yeomans, Montoya and Rosas, who invested more time and effort believing in me. To Prof. Álvarez for being so attentive and helping me so much during my graduate studies, always looking for opportunities that enabled me to pursue a specialization in astrophysics.

To my beloved family. You who've always been my pillar, blessing me with your love and candour. You've never doubted me even when the lights went dimmer. Specially to Nana who's still looking out for me, and my sisters with whom I've shared so many fraternal moments. To Ariadna, my dear confident and companion. Thank you all for your endless patience and support. Let's keep it strong.

Edgar M. Salazar

Abstract

Cosmic Microwave Background photons are inverse Compton scattered by energetic electrons in galaxy clusters and distorted by its bulk motion in a process known as the Sunyaev-Zel'dovich (SZ) effect. Conventional methods that calculate cluster peculiar velocities require additional information like the thermal component (tSZ) or underlying electron density n_e to estimate each cluster's optical depth τ, however observational measurements contain large errors and biases. This work studies the feasibility of using deep learning regression algorithms for estimating individual cluster peculiar velocities from the simulated kinematic (kSZ) component signal, thus exempting the need of τ. This formalism is tested using the Magneticum cosmological hydrodynamical simulation. Both tSZ and kSZ, along with n_e maps were generated at $z = [1.04 - 1.32]$. Trials with a simple convolutional neural network yielded prediction error standard deviations as low as $\sigma_e = 11.019\,\mathrm{km\,s^{-1}}$; whilst both methods implementing τ's explicit calculation yielded standard deviations of $81.569\,\mathrm{km\,s^{-1}}$ using tSZ and $87.527\,\mathrm{km\,s^{-1}}$ for n_e. The neural network provided reliable predictions for studying LSS velocity fields using the pairwise velocity estimator v_{12}, in addition to improved correlation compared to conventional methods.

Keywords: *peculiar velocity*, *optical depth*, *kinematic Sunyaev-Zel'dovich effect*, *neural network*

Resumen

Fotones de la radiación cósmica de fondo sufren dispersión Compton inversa causada por electrones energéticos en cúmulos de galaxias, y distorsión por su movimiento de bulto en un proceso conocido como efecto Sunyaev-Zel'dovich (SZ). Métodos convencionales para calcular velocidades peculiares de cúmulos requieren información adicional como la componente térmica (tSZ) o la densidad de electrones subyacente n_e para estimar la profundidad óptica τ de cada cúmulo, sin embargo, las mediciones observacionales contienen grandes errores y sesgos. Este trabajo estudia la factibilidad de utilizar algoritmos de regresión por aprendizaje profundo para estimar velocidades peculiares de cúmulos individuales a partir de la señal simulada de la componente cinemática (kSZ), exceptuando así la necesidad de τ. Este formalismo es probado usando la simulación cosmológica hidrodinámica Magneticum. Ambos mapas tSZ y kSZ junto con n_e son generados a $z = [1.04 - 1.32]$. Pruebas con una red neuronal convolucional simple arrojaron desviaciones estándar del error de predicción tan bajas como $\sigma_e = 11.019\,\mathrm{km\,s^{-1}}$; mientras que ambos métodos implementando el cálculo explícito de τ resultaron con desviaciones estándar de $81.569\,\mathrm{km\,s^{-1}}$ utilizando tSZ y $87.527\,\mathrm{km\,s^{-1}}$ para n_e. La red neuronal proveyó predicciones confiables para estudiar campos de velocidades a gran escala (LSS) utilizando el estimador de velocidades por pares v_{12}, además de una correlación mejorada comparada con los métodos convencionales.

Palabras clave: *velocidad peculiar*, *profundidad óptica*, *efecto Sunyaev-Zel'dovich cinemático*, *red neuronal*

Contents

List of Figures

List of Tables

List of Symbols

x_j	Feature, attribute or characteristic
$\mathbf{X}$	Instance or example
$\hat{y}$	Target or label
y	Prediction
$\mathbb{X}$	Instances dataset
$\hat{\mathbb{Y}}$	Targets dataset
$\mathbb{Y}$	Predictions dataset
$\hat{f}(\mathbf{X})$	Objective function
$f(\mathbf{X}, \boldsymbol{\theta})$	Prediction function
θ	Parameter
$\boldsymbol{\theta}$	Parameter set
$\mathcal{L}(y, \hat{y})$	Loss function (per sample)
ϕ	Activation function
$z_{\mathbf{W}}$	Weighted sum for fixed $\mathbf{W}$
$J(\boldsymbol{\theta})$	Cost function
$\mathbf{E}[\alpha^n]$	Expectation value of the n-th moment of α
$\mathbb{B}$	Minibatch
$\mathbf{g}$	Minibatch gradient
$\mathbf{A}_\ell$	Output set from layer ℓ
$\mathbb{M}_{m\times n}$	Matrices of shape $m \times n$

List of Acronyms

ΛCDM Lambda-Cold-Dark-Matter.

AI Artificial Intelligence.

AN artificial neuron.

ANN artificial neural network.

BAO baryon acoustic oscillations.

BGD batch gradient descent.

CDM cold dark matter.

CMB Cosmic Microwave Background.

COBE Cosmic Background Explorer.

DL Deep Learning.

DNN deep neural network.

ESA European Space Agency.

FLRW Friedmann-Lemaître-Robertson-Walker.

GD gradient descent.

ICM intra-cluster medium.

kSZ kinematic Sunyaev-Zel'dovich.

LF laboratory frame of reference.

LSS large-scale strucure.

LTU linear threshold unit.

ML Machine Learning.

MLP multilayered perceptron.

RF electron rest frame.

SGD stocastic gradient descent.

SPH smoothed-particle hydrodynamics.

SZ Sunyaev-Zel'dovich effect.

tSZ thermal Sunyaev-Zel'dovich.

WMAP Wilkinson Microwave Anisotropy Probe.

Introduction

Cosmology, as a formal discipline, is concerned with the study of large scale properties of the universe as a whole. These are thought to be imprinted, for example, in the Cosmic Microwave Background (CMB) anisotropies; distortions produced by under- and over-densities of matter at a period when radiation and matter were coupled. The CMB radiation field permeates the universe and is deformed by effects such as: Doppler shift (earth's relative motion), gravitational lensing (mass over-densities), gravitational redshift (Sachs-Wolfe effect), bremsstrahlung and synchrotron radiation (from plasma clouds), inverse Compton scatter (Sunyaev-Zel'dovich effects), and other more. In this sense, anisotropies contain information about how the universe is and came to be.

In fact there's some avail in studying CMB anisotropies directly. Take the Sunyaev-Zel'dovich effect (SZ), for example, which describes the distortion process via inverse Compton scatter caused by electrons in galaxy clusters. Thermal Sunyaev-Zel'dovich (tSZ) effect is due to the random motion of hot electrons and its power spectrum (Zeldovich & Sunyaev, 1969, 1972) is sensitive to matter density fluctuations characterized by cosmological parameters σ_8 and Ω_m (Makiya et al., 2019). This makes it a powerful probe of the present day matter density. A kinematic Sunyaev-Zel'dovich (kSZ) effect is caused by the bulk motion of the cluster and its power spectrum is particularly useful to estimate galaxy cluster peculiar velocities at high redshift(Sunyaev & Zeldovich, 1980). The velocity field is studied through ensemble statistics such as bulk flows, velocity correlation functions, and the pairwise estimators. Although kSZ is far weaker than tSZ, making it difficult to locate at low frequencies, both effects can be separated using their different spectra. Furthermore, both signals are redshift independent. This is a very desirable and uncommon feature that makes SZ a valuable cosmological and cluster evolutionary diagnostic tool (Rephaeli, 1995).

The kSZ has been detected from CMB maps using pairwise momentum estimator to probe for large-scale structure (LSS) at around 10 to 1000 h^{-1} Mpc showing substantial errors on the bulk flow (Hand et al., 2012; Lavaux et al., 2013). Similar methods have detected kSZ in real and Fourier spaces; by cross-correlating kSZ temperature map with velocity fields; and others through CMB temperature dispersion (Planck Collaboration, 2016, 2018b; Schaan et al., 2016; Hill et al., 2016; Soergel et al., 2016; Calafut et al., 2017; Sugiyama et al., 2018).

In order to estimate peculiar velocities from kSZ, the optical depth of the cluster must be measured, however it contains large errors and biases that affect the estimation. For instance, Lindner et al. (2015) report an average uncertainty of 30% and Mittal et al. (2018) forecast 24% in observations. Additionally, using emission-weighted temperature in the optical depth measurement may cause bias (Diaferio et al., 2005). The optical depth also varies between simulation models with or without star formation and feedback (Flender et al., 2016, 2017). The signal and large errors in optical depth measurements make the kSZ peculiar velocity calculation imprecise.

Recent work has been done to calibrate methods using simulated SZ signal as a probe for cosmology and calculate peculiar velocities, using computational techniques to estimate and analyse optical depth and comptonization. For example, Soergel et al. (2018) have shown promising results in obtaining pairwise velocity statistics with kSZ by applying map filtering to the signals and relied on tSZ to estimate average optical depth. But then again, the core problem with using kSZ for estimating peculiar velocities for individual clusters is that the optical depth must be computed from observational data, which has very large errors, and/or coarse statistical analysis from additional signals that cause large dispersions.

An alternative method for estimating peculiar velocities directly from kSZ can be found in Deep Learning (DL) algorithms. These are designed to perform complex analyses in a data-driven manner instead of the explicit programming of the physical processes, allowing the development and calibration of models using simulation data. DL algorithms for classification and regression tasks, are merely a subset of Machine Learning (ML) methods that make Artificial Intelligence (AI) an active field of research. Some ML methods like Gaussian processes, decision trees, nearest neighbour algorithms and support vector machines have been used in astrophysical context. In the study of kSZ, these methods give the possibility to overlook optical depth estimation and compute peculiar velocities directly. Therefore, an automatized computer program can predict and improve the estimation of peculiar velocities for single galaxy clusters from simulated signal maps. However, model interpretability becomes difficult as the algorithm is numb to the physical situation.

The objective of the present work is to implement a DL algorithm to process simulated kSZ signal maps and correctly predict individual cluster peculiar velocities. In order to achieve this the following actions were taken:

- Review of the pertinent theory, derived and presented in orderly fashion to situate the reader within the boundaries and reach of both SZ and DL.

- Implement the DL technique called *supervised learning for regression analysis* using TensorFlow API libraries.
- Test the methodology using simulated signal samples extracted from the Magneticum Simulation and generated using a light-ray tracing code (SMAC) for both thermal and kinetic SZ effects.

In chapter 1 the reader will find a review of the current cosmological theory and its model known as ΛCDM. CMB radiation is presented as the experimental finding that supports the current model, and the SZ effect is studied as one of the most relevant effects known to produce anisotropies. This chapter lays out a background to study the physics and relevance of the SZ effect in CMB analysis; addresses the explicit derivation of Zeldovich & Sunyaev (1969) results starting from the basics of Compton scattering and the Boltzmann equation in the non-relativistic Thomson limit. At the end of the chapter, the relativistic limit is briefly discussed to show the sufficiency of the non-relativistic approach when working with kSZ measurements.

Chapter 2 provides the necessary concepts of Machine Learning that will allow a better comprehension of the field and its analysis tools, along with the introduction of a formal notation. It covers core concepts of Neural Networks, learning algorithms, and the particularities of convolutional neural networks. A notation section and glossary for this chapter can be found on pages x and 73 respectively.

Chapter 3 contains a general description of the Magneticum Simulation and the selection of kSZ signal maps. Extensively resorting to the notation introduced in chapter 2, the network design choice is discussed followed by the construction of tests meant for exploration purposes. Next, model cross-validation method for regression analysis and model accuracy parameters are described. Results for all selected DL training tests come after in section 3.5. The methodology for estimating optical depths studied by Soergel et al. (2018) is replicated (section 3.6) in order to estimate peculiar velocities and compare with DL training. Furthermore, the direct computation of optical depth is also performed from electron density maps extracted from the same simulation box. Section 3.7 is dedicated to compare all model's error distributions, and section 3.8 presents a pairwise velocity estimator as the final test for model predictability power.

Final remarks and conclusions on this work are found in page 62.

Chapter 1 – Cosmology

This first chapter introduces the reader to cosmology by briefly presenting the current standard model and how it describes the observable universe. A quick review of CMB observations manifest the relevance of this relic radiation in cosmology studies. Then, peculiar velocities of galaxy clusters are discussed as probe for cosmology parameter estimation. The last section on this chapter provides a detailed derivation of SZ effect equations that serve for peculiar velocity estimation. Focused on proving the validity of both SZ equations, a detailed walk-through starting from inverse Compton scatter and the Kompaneets equation is provided. Two concepts relevant for SZ signal analysis are introduced: comptonization parameter and optical depth. Finally, some arguments are presented to justify that working with a non-relativistic approach is sufficient to comprehend the implications and usefulness of SZ effect.

1.1 The Cosmological Standard Model

The prevalent cosmological theory explaining how the observable universe came to be is called *cosmological inflation*. It depends on two assumptions: the universality of physical laws[1] and the cosmological principle[2]. It describes how the universe expanded from an immeasurable hot, dense, nearly homogeneous mixture of photons and matter tightly coupled, to an expanding structure. Universe evolution has been theorized to have several stages: inflation, recombination, structure formation and expansion (see Fig. 1.1). The first stage is called cosmic inflation, where the universe expanded exponentially and it is believed to be the reason why the universe seems to be isotropic and flat. Although initial conditions of the early plasma are difficult to establish because density fluctuations are seeded by quantum fluctuations in the field driving inflation (Baumann, 2009). The small perturbations propagate through the plasma collisionally like acoustic waves producing under- and over-densities in the plasma itself with simultaneous changes in density of matter and radiation. Cold dark matter (CDM)[3] doesn't share in these pressure-induced oscillations, but does act gravitationally, either enhancing or negating the acoustic pattern for the photons and baryons (Hu &

[1] This is the underlying principle in the theory of general relativity which has already passed stringent tests.

[2] Can be derived from the Copernican principle and has been confirmed by CMB observations.

[3] Non-relativistic matter with little or null interaction with radiation and ordinary matter.

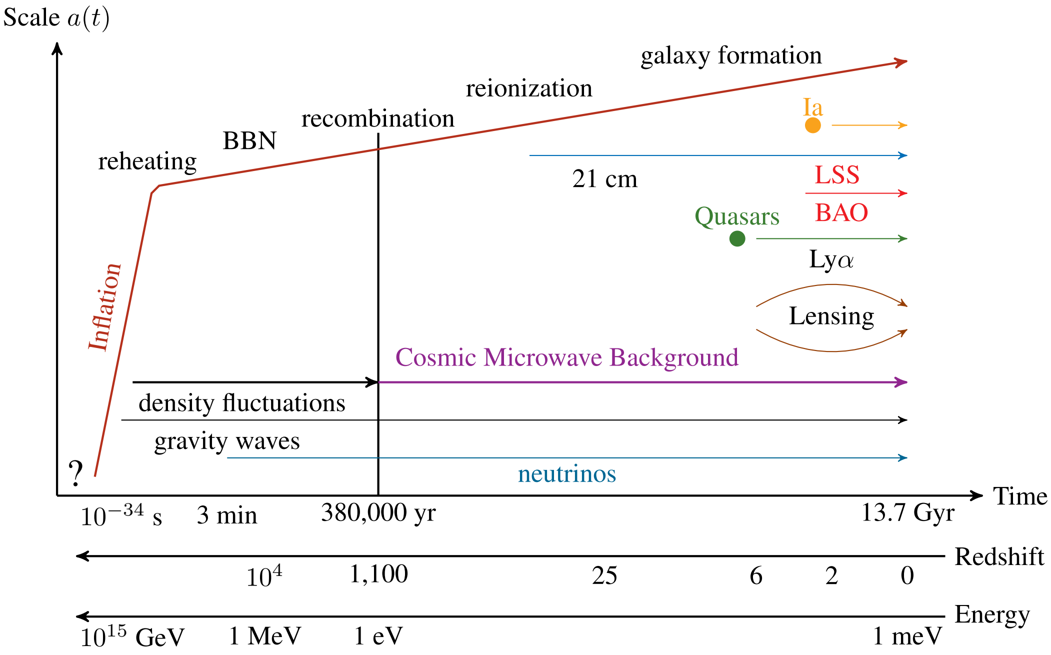


Figure 1.1: History of the universe. Key events and their associated time and energy scales (adapted Fig. 2 from Baumann, 2009). The question mark at the origin represents all unknown processes occurred at inflation. Acronyms: BBN (Big Bang Nucleosynthesis), LSS (Large-Scale Structure), BAO (Baryon Acoustic Oscillations), Lyα (Lyman-alpha), Ia (Type Ia supernovae), 21cm (hydrogen 21 cm-transition).

White, 2004). During inflation, all elementary particles and antiparticles were produced. At some point the process of baryogenesis took place producing an imbalance of matter and antimatter, leading to an excess of quarks and leptons over their antimatter counterparts. What led to baryogenesis is still to be determined. In fact, processes occurred during inflation are mostly speculative, e.g. string scale, grand unification, (super) symmetry breaking and baryogenesis.

Later came nucleosynthesis; a period of cooling and density decrease when plasma reached a point where quarks and gluons combined to form baryons. Electrons and baryons where able to stably recombine and form atoms, mostly in the form of neutral hydrogen. At recombination photons decouple from the baryons as the plasma becomes neutral, and perturbations no longer propagate as acoustic waves: the existing density pattern becomes “frozen”. This snapshot of the density fluctuations is preserved in the CMB anisotropies and the imprint of baryon acoustic oscillations (BAO) observable today in large-scale strucure (LSS) (Eisenstein & Hu, 1998). Recombination produces a largely neutral universe which is unobservable throughout most of the electromagnetic spectrum. During this era, CDM begins gravitational collapse in over-dense regions. Baryonic matter gravitationally collapses into these CDM halos, and so begins the formation of the first radiation sources such as stars. Radiation from these objects reionizes the intergalactic medium (Switzer, 2016), thus ending the *cosmic dark ages* and beginning the *cosmic dawn*.

All this took place around the first 400 million years since inflation started. During the final stages structure continues to grow and merge under the influence of gravity thus forming the vast cosmic web of dark matter density now observed (LSS). As the universe continues to expand, a negative pressure (thought to be some kind of *Dark Energy*) increasingly dominates over-opposing gravitational forces, and so the universe expands.

Einstein introduced the cosmological parameter Λ to the field equations in order to fit a static universe, but Alexander Friedman (1922) explored the idea of space curvature characterized by a radius of curvature k, where $k = 0$ is for flat space-time; $k = 1$ for a closed universe; and $k = -1$ for the opposite. He proposed a solution considering a homogeneous and isotropic space based on the Friedmann-Lemaître-Robertson-Walker (FLRW) metric as an exact solution. This metric assumes that the spacial coordinates are scaled by a dimensionless scale factor $a(t)$, which is related to the Hubble parameter (expansion rate) and cosmological redshift by $H(t) = \frac{\dot{a}(t)}{a(t)}$ and $a(t) = \frac{1}{1+z}$ (assuming $a(t_0) = 1$) respectively.

As described by General Relativity and the FLRW metric, the Lambda-Cold-Dark-Matter (ΛCDM) model is a parametrization of the *inflationary* theory and encompasses the existence of radiation, ordinary matter (baryonic), cold-dark matter, and a negative pressure or soft density of empty space (coded in the Λ parameter) which is responsible for the observed acceleration in the Hubble expansion. The standard ΛCDM model further considers that both contribution of radiation density and neutrino mass are negligible. Equation (1.1) gives the evolution of the Hubble parameter at time t as a result of density parameters Ω

$$H^2 = H_0^2 \left((\Omega_b + \Omega_c)\, a^{-3} + \Omega_k\, a^{-1} + \Omega_\Lambda\, a^{3(1+w)} \right) \tag{1.1}$$

where $\Omega_m \approx \Omega_b + \Omega_c$ is the total matter density of the universe (baryons and CDM); Ω_k is the curvature density parameter; Ω_Λ is the cosmological constant density for negligible neutrino mass ($w = -1$); $a(t)$ the scale parameter; and H_0 is the Hubble parameter at t_0. Its common practice to use the reduced Hubble constant h parameter to express the current expansion rate. Related to the Hubble constant by $H_0 = 100\ h\ \mathrm{km\,s^{-1}\,Mpc^{-1}}$, a value of $h = 0.7$ indicates that a galaxy located $1\,\mathrm{Mpc}$ away from the observer at t_0 recedes at $70\,\mathrm{km\,s^{-1}}$.

Table 1.1 contains the latest results (Hinshaw et al., 2013; Planck Collaboration, 2018a) for ΛCDM parameters. The first three specify the corresponding density parameters assuming the total density $\Omega = 1$. Curvature fluctuation amplitude measures the deviation of k from a flat space-time. In this model, field fluctuations are the root cause of the observed CMB anisotropies which are characterized by the primordial scalar perturbations as a power law relation $\propto k^{n_s - 1}$, thus n_s is a critical parameter that indicates the strength of CMB anisotropies on all angular scales. The reionization optical depth provides a measure of the line-of-sight free-electron opacity to CMB radiation. Assuming instantaneous and complete reionization at z_{reion}; it is computed as the integral of the electron density times the Thomson cross section over the geometrical path length computed between redshifts $z = 0$ (now) and z_{reion}. Lastly, the amplitude of matter density fluctuations parameter σ_8 tells how matter is distributed on scales around $8\ h^{-1}\ \mathrm{Mpc}$ and defined as the RMS of the $z = 0$ density perturbations.

Making the explicit correspondence between cosmological observables, such as temperature fluctuations (e.g. WMAP[4]) or galaxy density inferred in a galaxy survey (e.g. SDSS[5]), and LSS is crucial for constraining inflation predictions. More on CMB anisotropies is discussed in the next section.

Table 1.1: ΛCDM parameters from WMAP 9 and Planck PR3 derived from multiple sources. WMAP reports Ω_Λ as a fit parameter whilst Planck as derived, and the acoustic scale θ of BAO instead as a fit parameter.

Parameter	Name	WMAP 9	Planck PR3
$\Omega_b h^2$	Baryon density	$0.022\,23(33)$	$0.022\,42(14)$
$\Omega_c h^2$	Cold dark matter density	$0.1153(33)$	$0.119\,33(91)$
Ω_Λ	Dark energy density	$0.7135(95)$	$0.6889(56)$
Δ_k^2	Curvature fluctuation amplitude $[\ln(10^{10}\Delta_k^2)]$	$3.195(94)$	$3.047(14)$
n_s	Spectral index	$0.9608(80)$	$0.9665(38)$
τ	Reionization optical depth	$0.081(12)$	$0.0561(71)$
Derived Parameters			
Ω_m	Matter density	$0.2865(96)$	$0.3111(56)$
H_0	Hubble constant	$68.76(84)$	$67.66(42)$
σ_8	Amplitude of matter density fluctuations	$0.820(14)$	$0.8102(60)$

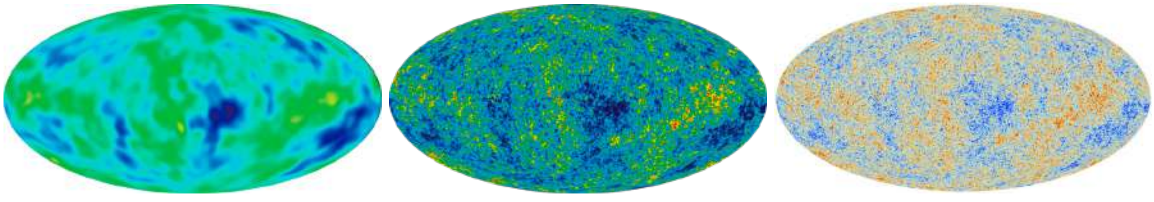
[4] Wilkinson Microwave Anisotropy Probe.

[5] Sloan Digital Sky Survey.

1.2 Cosmic Background Radiation

G. Gamow (1946) realized that after the recombination period, where electrons and protons became bound for the first time, the universe should be filled with a black-body radiation field. Two decades later, this radiation was first measured by Penzias & Wilson (1965) in the microwave regime wavelength of $7.34\,\mathrm{cm}$, from which they inferred a black-body temperature of $3.5(10)\,\mathrm{K}$ in accordance to Gamow's theory. In 1992, the Cosmic Background Explorer (COBE) outer space probe took precise measurements of the same spectrum detecting anisotropies. It was able to determine a mean temperature of $2.7\,\mathrm{K}$ with a precision of 0.005% (NASA, 2016) corresponding to an almost perfect fit to the black-body spectrum theorized in the *inflationary* model. It also provided an estimate to the magnitude of anisotropies, around 10^5 times smaller than the average temperature of the radiation field.

The Wilkinson Microwave Anisotropy Probe (WMAP) operated from 2001 to 2010 providing high resolution measurements of the ΛCDM parameters (see Table 1.1). These results estimate the age of the universe at $13.76(11)\,\mathrm{Gyr}$ (thousand million years), and the data is very well fit for a universe dominated by *dark energy*. In year 2009 the European Space Agency (ESA) launched the Planck probe (Planck Collaboration, 2015) with even higher resolution, being the first to distinguish details in the structure of the CMB that were not distinguishable before. Measurements have found that in fact CMB is isotropic and has a thermal black-body temperature of $2.725\,48(57)\,\mathrm{K}$ (Planck took this measurement from Fixsen (2009) with negligible impact on their results). Figure 1.2 shows how measurement resolution from outer space probes of temperature density contrast $\Delta T/T$ has improved over the years. Although it might seem as if the spectrum is not isotropic, anisotropies are very small with variations smaller than a few µK (Wright, 2004).



(a) COBE

(b) WMAP

(c) Planck

Figure 1.2: Comparison between CMB density contrast sky surveys improving over 20 years[6]. The scale is of order $\Delta T/T \sim 10^{-5}$, where blue is *cooler* and yellow/red is *hotter* than the average temperature T. Images retrieved from http://www.nasa.gov and https://www.cosmos.esa.int.

As mentioned before, CMB is a field of photons that went out of thermal equilibrium with matter in the early universe after it became transparent, around $z_{\rm reion}$ $\approx$1,100. This is often called the *last scattering*, referring to when photons were scattered by hot plasma electrons for the "last time", constituting the *surface of the last scatter*, i.e. CMB. For the most part, this field is homogeneous and isotropic, meaning that early universe distribution of matter and radiation is also homogeneous and isotropic. However, matter under- and over-densities present at recombination caused fluctuations in the radiation field through their gravitational perturbations, thermodynamic fluctuations in the density of radiation coupled with matter, and through Doppler shifts due to motions of the surface of the last scatter (Birkinshaw, 1999). The angular power spectrum $\frac{\Delta T(\hat{n})}{T_0} = a_{\ell m} Y_{\ell m}(\hat{n})$ [7] is an important tool in CMB statistical analysis. It describes the cosmological information contained within the millions of pixels of a CMB map in terms of a much more compact data representation.

For example, the monopole and dipole terms are related to the radiation's average temperature and earth's relative motion with respect to the CMB rest frame, respectively. Higher order multipoles are more complex contributions of several effects that deform the spectrum such as: gravitational lensing produced by mass distribution inhomogeneities; and gravitational redshift known as Sachs-Wolfe effect, where radiation's frequency shifts due to gravity wells. Other sources of radiation also contribute, like the well known bremsstrahlung and synchrotron radiation produced by charged particles; radiation emitted from stars and absorbed by dust clouds (just to mention some); as well as scattering effects like regular and inverse Compton. All these (and more) give the anisotropies observed (see Fig. 1.2) and contain information on the mass distribution around the universe and LSS. In this sense, anisotropies contain information about how the universe is and came to be.

The most likely sources of perturbations to CMB are galaxy clusters with masses that often exceed 10^{14} $M_\odot$ and radii around $\rm Mpc$. The total gas fraction is around 16% with $\sim$13% in the hot intra-cluster medium (ICM) and the remaining 3% in stars in the cluster galaxies. The remaining $\sim$84% of the mass is a dark matter halo. Gas densities in cluster centres range from as much as 10^{-1} to 10^{-3} particles per $\rm cm^3$ in peaked clusters to the non-peaked ones. This is in stark contrast to the mean cosmic density of baryons of about 10^{-8} particles per $\rm cm^3$ (Peterson & Fabian, 2006).

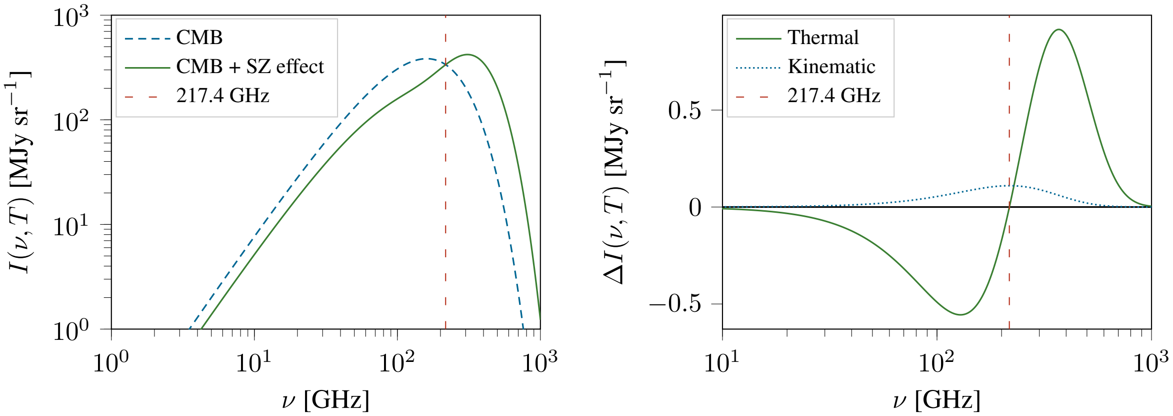
The ICM is plasma that is nearly fully ionized due to the high temperatures created by the deep dark matter gravitational potential. It consists mainly of ionized hydrogen and helium

[7] Assume Einstein summation convention for repeated indices.

and strongly emits X-Ray radiation, primarily due to bremsstrahlung process. Free electrons in the medium can also scatter low energy CMB photons and cause distortions to its spectrum. This is known as the Sunyaev-Zel'dovich effect (SZ).

Few years after Penzias and Wilson's measurements, Zeldovich & Sunyaev (1969) were able to find an analytical expression for the distortion of the CMB spectrum produced by the scattering of thermal electrons in the non-relativistic approximation studied by Kompaneets (1956). This is formally known as the thermal Sunyaev-Zel'dovich (tSZ) effect and it's the result of inverse Compton scattering of low CMB photons by thermally distributed hot electrons in massive clusters of galaxies (LaRoque et al., 2002). Figure 1.3a shows the CMB black-body spectrum and how it is distorted by the SZ effect. TSZ manifests as an increment in the high frequency part of the CMB spectrum and as a decrement in the low frequency region, with a crossover frequency of $217.4\,\mathrm{GHz}$ (see Fig. 1.3b).

Additionally, a kinematic Sunyaev-Zel'dovich (kSZ) contribution in the spectrum, is caused by the ICM bulk radial motion with respect to the CMB rest frame (Doppler shift). This effect shifts the CMB black-body spectrum to a slightly lower (higher) temperature for receding (approaching) velocities from the observer with its minimum (maximum) drop (raise) at the crossover frequency (Birkinshaw, 1999). This makes the measurement of kSZ effect to be around the $217\,\mathrm{GHz}$ mark. Because tSZ signal is much more intense than kSZ (approx. 10^2, see Fig. 1.3b) this makes it harder to detect kSZ alone.



(a) CMB black-body spectrum and distorted spectrum by SZ effect.

(b) Spectral form of both thermal and kinetic SZ effects if $T = 2.725\,\mathrm{K}$ (Rephaeli, 1995). kSZ was amplified to ease visualization.

Figure 1.3: Sunyaev-Zel'dovich effect distortion to CMB black-body spectrum

The most interesting aspect of kSZ is that it provides a method for measuring the line-of-sight peculiar velocity of an object at large distance, provided that the velocity and thermal effects can be separated, as they can using their different spectral properties. Further discussion is found in section 1.4.5 when the proper solutions are showed.

1.3 Peculiar Velocities

Peculiar motion refers to the movement of an object relative to a reference frame at rest. As galaxies are typically found in groups or clusters, they have significant gravitational effects on each other. The peculiar velocity field is therefore sensitive to mass fluctuations on large scales, thus a powerful tool for constraining cosmological parameters.

However, the measurement precision of the velocity field is limited by the error in radial distance measurement, which tend to have a non-Gaussian error distribution that may bias the results. More so, peculiar velocity errors tend to be proportional to the radial distance which in turn have fractional observational errors typically around 20%. For these reasons, a single peculiar velocity measurement is not a good approximation to a cluster's velocity. Statistical ensembles (especially of low-order moment statistics) may be a good estimator of the cosmic velocity field and therefore a good tracer of the mass distribution in the Universe.

Many recent studies have focused on the bulk flow, which is the lowest order statistic of the velocity field and is generally thought of as the average of peculiar velocities in a volume. Its calculation is typically done by two approaches: a maximum likelihood estimate (MLE) and minimum variance (MV). A drawback from MLE is that, even though it basically reduces the entire data to the bulk flow vector components, since the particular data and error distribution in the surveys analysed are unique to each catalogue it is difficult to compare between independent surveys. Regarding MV, which minimizes the differences between the actual observational data and an "ideal" survey that may be designed to probe a volume in a particular way, it easily lends itself to direct comparisons between independent surveys.

Another approach to study LSS velocity field is the pairwise velocity statistic $v_{12}(r)$ introduced by Ferreira et al. (1999). It takes the mean peculiar velocity difference between galaxy pairs at a distance r. Since only the line-of-sight component of the velocity is observed $v_i \equiv \hat{\mathbf{r}}_i \cdot \mathbf{v}_i$, rather than the full three-dimensional velocity $\mathbf{v}_i$, it is not possible to compute v_{12} directly. The pairwise velocity estimator weights each pair by the factor $c_{ij} \equiv \hat{\mathbf{r}}_{ij} \cdot (\mathbf{r}_j + \mathbf{r}_i)$ and sums over all pairs at fixed separation $r = |\mathbf{r}_j - \mathbf{r}_i|$.

$$v_{12}(r) = \frac{2\sum (v_j - v_i)\ c_{ij}}{\sum c_{ij}^2} \tag{1.2}$$

On very large scales, $v_{12}(r)$ is proportional to $\Omega^{0.6}\sigma_8^2$, and on intermediate scales the degeneracy is removed and Ω and σ_8 can be measured separately (Juszkiewicz et al., 1999). This work limits to the estimation of $v_{12}(r)$ given in Eq. (1.2) as a test for peculiar velocity estimation methods in ensemble statistical analysis (see section 3.8).

1.4 Sunyaev-Zel'dovich Effect

CMB is the dominant electromagnetic radiation field in the Universe, and Compton scattering is one of the main physical processes that couples radiation and matter. Scattering of CMB photons by free electrons can have important observable effects, for instance, the SZ effect. This section contains a revision of Compton scattering as a means to justify the Thomson scattering approximation (Rybicki & Lightman, 1985; Peebles, 1993) at the electron's rest frame; succeeded by a derivation of the Kompaneets equation for the non-relativistic limit of thermal scattering, and the obtention of the SZ equations (Kompaneets, 1956; Zeldovich & Sunyaev, 1969, 1972; Sunyaev & Zeldovich, 1980; Rephaeli, 1995; Birkinshaw, 1999).

1.4.1 Compton Scattering

Consider a single scatter process $e_0 + \gamma_0 \longrightarrow e_1 + \gamma_1$ as seen from the laboratory frame of reference (LF) S at earth's surface. The subscript 0 refers to the initial state before scattering; and 1, to a single scatter process. The photon will have four-momenta given by $P_0 = c^{-1}\epsilon_0\,(1, \hat{\mathbf{n}}_0)$ and $P_1 = c^{-1}\epsilon_1(1, \hat{\mathbf{n}}_1)$, where $\epsilon = h\nu$ and $\hat{\mathbf{n}}$ is a unitary vector; and the electron, $Q_0 = \gamma m\,(c, \mathbf{v})$ and $Q_1 = \gamma_1 m(c, \mathbf{v}_1)$, where $\hat{\mathbf{v}}$ is the velocity measured from S. Note that the subscript 0 will not be necessary as shown below. Four-momentum conservation states

$$P_0 + Q_0 = P_1 + Q_1 \tag{1.3}$$

Using the Minkowski norm invariance it's possible to express the final state in terms of the initial state by taking the norm squared on both sides of Eq. (1.3), getting $P_0 \cdot Q_0 = P_1 \cdot Q_1$. Multiplying Eq. (1.3) by P_1 from the left on both sides and considering the previous result, evaluates to the equivalent explicit expression

$$c^{-2}\epsilon_1\epsilon_0\,(\hat{\mathbf{n}}_1 \cdot \hat{\mathbf{n}}_0 - 1) + c^{-1}\epsilon_1\gamma m\,(\hat{\mathbf{n}}_1 \cdot \mathbf{v} - c) = c^{-1}\epsilon_0\gamma m\,(\hat{\mathbf{n}}_0 \cdot \mathbf{v} - c) \tag{1.4}$$

This expression is independent of the electron's final state, i.e. it does not affect the scattered photon, which is valid only for the single scatter. In reality the electron most likely won't be at rest after the scatter but fixing the reference frame at the initial position suffices for describing the photon's energy change. Introducing the scattering angle $\cos\varphi_{01} = \hat{\mathbf{n}}_1 \cdot \hat{\mathbf{n}}_0$ between the incoming and the outgoing photon directions; and the angles with respect to the axis defined by the electron velocity $v\cos\theta_0 = \hat{\mathbf{n}}_0 \cdot \hat{\mathbf{v}}$ and $v\cos\theta_1 = \hat{\mathbf{n}}_1 \cdot \hat{\mathbf{v}}$; and solving Eq. (1.4) for the scattered energy ϵ_1 gives

$$\epsilon_1 = \epsilon_0 \, \frac{1 - \beta\cos\theta_0}{1 - \beta\cos\theta_1 + \dfrac{\epsilon_0}{\gamma mc^2}\left(1 - \cos\varphi_{01}\right)} \tag{1.5}$$

The change in the photon's frequency is due to regular Compton scattering by an electron with energy $E = \gamma mc^2$ and velocity $\beta = \frac{v}{c}$, from an angle θ_0 to an angle θ_1 measured from LF. Changing to the electron rest frame (RF) S' means taking $\beta = 0$, with $E' = mc^2$

$$\epsilon_1'\left(\epsilon_0', \Omega\right) = \epsilon_0' \, \frac{1}{1 + \dfrac{\epsilon_0'}{E'}\left(1 - \cos\varphi_{01}'\right)} \tag{1.6}$$

where φ_{01}' is the angle of deflection of the photon measured from the direction of incidence; Ω is the solid angle defined by φ_{01}'. The corresponding transformations between reference frames are simply $\epsilon' = \epsilon\gamma\left(1 - \beta\cos\theta\right)$ and $\epsilon = \epsilon'\gamma\left(1 + \beta\cos\theta'\right)$; the angle transformation law relative to the electron's velocity is $\cos\theta' = \frac{\cos\theta - \beta}{1 - \beta\cos\theta}$; and the deflection angle is given by $\cos\varphi_{01} = \cos\theta_1\cos\theta_0 + \sin\theta_1\sin\theta_0\cos(\phi_0 - \phi_1)$, where ϕ_1 and ϕ_0 are the azimuthal angles of the scattered photon and incident photon in S.

1.4.2 Inverse Compton Scattering

Since photons can come from anywhere, $\varphi_{01}' \in [-\pi, \pi]$, energy is lost from the recoil except for purely forward scattering, as Fig. 1.4. This means that at RF a photon will always cede energy by regular Compton scattering with an outgoing energy $\epsilon_1' \leqslant \epsilon_0'$. Energy loss is big for very energetic photons with $\epsilon_0' \gg mc^2$, as the outgoing frequency is very small at big angles. On the other hand, for low energy photons and mildly relativistic or non-relativistic electrons $\epsilon_0' \ll mc^2 \ll E$ the scattering is almost elastic $\epsilon_1' \to \epsilon_0'$. This limit recovers Thomson scattering, which is also appropriate for the SZ effect and causes considerable simplifications in the physics and mathematics.

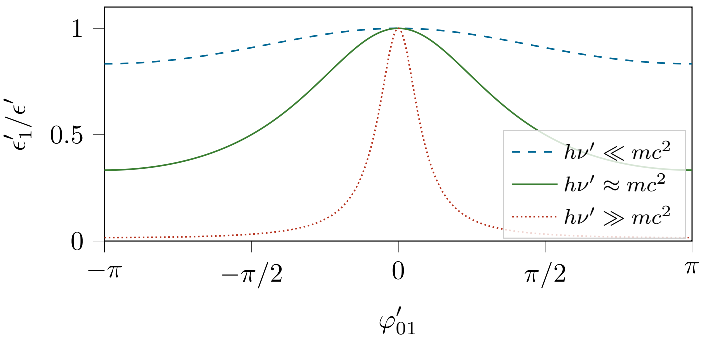


Figure 1.4: Fractional frequency change of the scattered photon measured from RF.

Whenever the moving electron has sufficient kinetic energy compared to the photon, net energy may be transferred from the electron to the photon. The outgoing photon's energy (frequency) is higher and it's said to be up-scattered. This process is called inverse Compton scattering. The working conditions will be taking $\beta \to 1$ and $\gamma \gg 1$ for relativistic electrons, but maintaining low energy photons $\epsilon_0' \ll \gamma mc^2$ so that the scattering is Thomson in RF but inverse Compton in LF.

Photons can come from any random direction but the outgoing photon is strongly beamed in the direction of the velocity so $\cos\theta_1' \approx 1$. From Eq. (1.6), typical scattering angles of $\pi/2$, taking into account the considerations previously made for inverse Compton scatter, and working back to LF quantities it's possible to approximate the photon's energy by

$$\epsilon_1 \approx \epsilon_0\gamma\left(1+\beta\cos\theta_1'\right)\left(1+\beta\cos\theta_0\right) \sim \gamma^2\epsilon_0$$

Although a coarse estimation, it shows that even if the incoming photon's energy in RF is low, as long as the electron has a large kinetic energy, the outgoing photon can greatly increase its energy in LF. Because $\gamma \gg 1$, the up-scatter can give very large energies. The maximum energy boost is around $\gamma^2\epsilon_0 < \gamma mc^2 = \gamma(511\,\mathrm{keV})$ so the energy ϵ_1 can be enormous and the scattering is still Thomson in RF.

1.4.3 Inverse Compton Power

Let $n(p)$ be the photon phase space distribution function (Lorentz invariant), and $v d\epsilon$ be the density of photons having energy in range $d\epsilon$ [8]. Then v and n are related by $v d\epsilon = n d^3p$. Because $d^3p = \gamma d^3p'$, i.e. transforms as energy under Lorentz transformations the quantity

[8] In the following the subscript 0 is omitted for incoming or before scattering quantities.

$\frac{v d\epsilon}{\epsilon} = \frac{v' d\epsilon'}{\epsilon'}$ is Lorentz invariant. The total power emitted in RF due to scattering is

$$P' = c\sigma_{\mathrm{T}} \int \epsilon' v' d\epsilon' \tag{1.7}$$

The integral's kernel is the energy density of incident photons. Assuming that the change in energy at RF is negligible compared to the change at LF is valid because the energy difference in both reference frames is of order γ^2. Also, the total emitted power invariance $P = P'$ allows to express

$$P = \gamma^2 c\sigma_{\mathrm{T}} \int (1 - \beta \cos\theta_0)^2 \, \epsilon v d\epsilon$$

which refers solely to quantities in S (LF). For an isotropic distribution of photons the average value over all directions of the term in parenthesis is $1 + \frac{1}{3}\beta^2$, thus

$$P = \gamma^2 c\sigma_{\mathrm{T}} \left(1 + \frac{1}{3}\beta^2\right) U_{\mathrm{ph}} \quad \text{with} \quad U_{\mathrm{ph}} = \int \epsilon v d\epsilon \tag{1.8}$$

where U_{ph} is the initial photon energy density, whose decrease rate is simply $-c\sigma_{\mathrm{T}} U_{\mathrm{ph}}$. The net power lost by the electron and thereby converted into increased radiation is then $\frac{4}{3} c\sigma_{\mathrm{T}} \gamma^2 \beta^2 U_{\mathrm{ph}}$. Introducing the electron's equilibrium temperature $\Theta = \frac{kT}{mc^2}$, the thermal energy is related to β by $\langle \beta^2 \rangle = 3\Theta$ for an electron cloud at absolute temperature T. So the total power for a thermal distribution of non-relativistic electrons with number density n_e at equilibrium temperature T is

$$P = 4\Theta c\sigma_{\mathrm{T}} n_e U_{\mathrm{ph}} \tag{1.9}$$

The average energy of a photon due to ICM scatterings with thermal electrons at the non-relativistic limit, and averaging over all angles:

$$\frac{\Delta\epsilon'}{\epsilon'} \approx -\frac{\epsilon'}{mc^2}$$

On the other hand, the energy change from the power emitted per scattering divided by the collision rate gives a net energy gain $\frac{\Delta\epsilon}{\epsilon} \sim \frac{4}{3}\beta^2 = 4\Theta$. Considering energy conservation in the scattering process, the energy loss of electrons is equal to the energy gain of photons. Thus the energy fractional change of the radiation field per scattering due to non-relativistic electrons in thermal equilibrium must be:

$$\frac{\Delta\epsilon}{\epsilon} = 4\Theta - \frac{\epsilon}{mc^2} \tag{1.10}$$

If $\epsilon < 4\Theta$, the photon gain energy (inverse Compton); if $\epsilon = 4\Theta$, there is no energy exchange; and if $\epsilon > 4\Theta$, the photon cedes energy (Compton).

1.4.4 Non-Relativistic Limit: Kompaneets Equation

Examine the evolution of photon phase space density $n\left(\nu,t\right)$ with frequency $\nu = h^{-1}\epsilon$ at time t due to scattering from electrons. Assuming n to be isotropic, and if $f\left(\mathbf{p}\right)$ is the phase space density of electrons with momentum $\mathbf{p}$, then the Boltzmann equation states that the change in photon occupation number of a radiation field $n\left(\nu,t\right)$ with respect to time due to Compton scatterings is:

$$\frac{\partial n}{\partial t} = -\,c\int d^3p\, d\Omega\, \frac{d\sigma}{d\Omega}\left[n(\nu)f(\mathbf{p})\left(1+n(\nu_1)\right) - n(\nu_1)f(\mathbf{p}_1)\left(1+n(\nu)\right)\right] \tag{1.11}$$

considering the scattering events $\mathbf{p}+\nu \rightleftharpoons \mathbf{p}_1+\nu_1$ (Peebles, 1993). The first term inside the brackets in Eq. (1.11) represents scattering out of frequency ν into frequencies ν_1, and the second term represents the opposite process. The factors $1+n(\nu)$ and $1+n(\nu_1)$ take into account stimulated scattering effects, that is: the probability of scattering from ν to ν_1 is increased by the factor $1+n(\nu_1)$. Considering photon number conservation from the radiation field in scattering process, photons must obey Bose-Einstein statistics and tend toward mutual occupation of the same quantum state.

The probability that photons are scattered into a solid angle Ω is the differential cross section given by Klein-Nishina formula for relativistic electrons. For low photon energies $\epsilon \ll mc^2$ and non-relativistic electrons, elastic scattering dominates so Klein-Nishina formula reduces to the differential Thomson cross section for unpolarized incident radiation:

$$\frac{d\sigma_{\mathrm{T}}}{d\Omega} = \frac{1}{2}r_0^2\left(1+\cos^2\theta\right) \tag{1.12}$$

where $r_0 = 2.818\times10^{-15}\,\mathrm{m}$ is the classical electron radius and $\sigma_{\mathrm{T}} \approx 6.6524\times10^{-29}\,\mathrm{m}^2$ is Thomson's cross section. A solution to Eq. (1.11) in the non-relativistic second order approximation for thermal electron population (or Fokker-Planck approximation) was first derived by Kompaneets (1956) considering that CMB photons are in thermal equilibrium once they enter the ICM at an absolute temperature Θ. The electron phase space density $f\left(\mathbf{p}\right)$, alternatively $f(E)$ where $E = \mathbf{p}^2/2m$ is given by

$$f(\mathbf{p}) = \left(2\pi m^2c^2\Theta\right)^{-3/2}\, n_e\, \exp\left[-\frac{\mathbf{p}^2}{2m^2c^2\Theta}\right] \tag{1.13}$$

where n_e is the electron space density and $f\left(\mathbf{p}\right)$ integrates to n_e. The frequency shift due to Compton scattering $\Delta\nu$ is very small, such that both occupation number and electron distribution after the scattering can be expanded in Taylor series of the small variable $\Delta\nu$. Introducing the dimensionless frequency x and dimensionless energy transfer Δ variables:

$$x = \frac{h\nu}{kT} \qquad \text{and} \qquad \Delta = \frac{h\Delta\nu}{kT} \tag{1.14}$$

where T is the radiation field temperature (CMB)[9]. An increase in frequency is expected due to inverse Compton scattering and a corresponding decrease in electron energy[10], so:

$$n\left(\nu_1\right) \quad = n(\nu) + \Delta\frac{\partial n}{\partial x} + \frac{1}{2}\Delta^2\frac{\partial^2 n}{\partial x^2} + \cdots \tag{1.15}$$

$$f(E_1) \quad = f(E)\left[1 + \Delta + \frac{1}{2}\Delta^2 + \cdots\right] \tag{1.16}$$

Using Eqs. (1.15) and (1.16) in Eq. (1.11):

$$\begin{aligned}\frac{\partial n}{\partial t} = &\left(\frac{\partial n}{\partial x} + n(n+1)\right)\boxed{c\int\frac{d\sigma}{d\Omega}\,f(\mathbf{p})\,\Delta\,d^3\mathbf{p}\,d\Omega}\nearrow\mathcal{I}_1\\ &+\frac{1}{2}\left(\frac{\partial^2 n}{\partial x^2} + 2(n+1)\frac{\partial n}{\partial x} + n(n+1)\right)\boxed{c\int\frac{d\sigma}{d\Omega}\,f(\mathbf{p})\,(\Delta)^2\;d^3\mathbf{p}\,d\Omega}\nearrow\mathcal{I}_2\end{aligned}$$

It's possible to anticipate (as expected) that the Bose-Einstein distribution $n(x) = \left(e^{x+\alpha} - 1\right)^{-1}$ is a steady-state solution since both terms involving derivatives vanish for this distribution, and photon number conservation is assumed. Using the elastic limit differential cross section in Eq. (1.12), the first integral $\mathcal{I}_1$ (in blue) can be calculated noticing that by definition it is the energy transfer rate, which is the average transfer times the number of collisions $\sigma_{\mathrm{T}} n_e c\Delta\epsilon/kT$. The average energy transfer per collision of photons with energy ϵ is given by Eq. (1.10), which can be rewritten in terms of x as $\frac{\Delta\epsilon}{kT} = \Theta\; x\,(4-x)$, thereby $\mathcal{I}_1 = c\sigma_{\mathrm{T}} n_e \Theta\; x\,(4-x)$. The second integral $\mathcal{I}_2$ (in red) can be obtained by direct calculation which gives $2c\sigma_{\mathrm{T}} n_e \Theta\; x^2$. From these results, the equation transforms to

$$\frac{\partial n}{\partial t} = \frac{\Theta\sigma_{\mathrm{T}} n_e c}{x^2}\frac{\partial}{\partial x}\left[x^4\left(\frac{\partial n}{\partial x} + n(n+1)\right)\right] \tag{1.17}$$

[9] Note the use of Θ for referring to the electron and ICM temperature, whereas the variable T is reserved for radiation temperature only.

[10] It has been expressed in terms of the electron's energy due to calculus simplification.

Equation (1.17) is the Kompaneets equation (Kompaneets, 1956; Weymann, 1965). Using random walk arguments it's possible to analyse what happens in the case of multiple scatterings by a dispersive medium. When photons co-exist in a region of size ℓ, the repeated scattering distort the original spectrum of photons, i.e. comptonization. The mean free path of the photon due to Thomson scattering is $\lambda = \frac{1}{n_e \sigma_\mathrm{T} \ell}$. If the size of the region ℓ is such that $\lambda^{-1} \gg 1$, then the photon will undergo many collisions in this region, but if $\lambda^{-1} \ll 1$ there will be few collisions.

The optical depth is defined as $\tau \equiv \lambda^{-1} = n_e \sigma_\mathrm{T} \ell$. If $\tau \gg 1$ (optically thick, strong scattering) then the photon undergoes N ($\gg 1$) collisions in travelling a distance ℓ. From standard random-walk arguments, $N \simeq \tau^2$. On the other hand, if $\tau \leqslant 1$ (optically thin), then the number of collisions is of order $1-e^{-\tau} \approx \tau$, i.e. $N \simeq \tau$. Therefore an estimate for the number of scatterings is $N \simeq \max(\tau, \tau^2)$. The average fractional change in the photon energy per collision is 4Θ, hence the condition for significant change of energy is $4\Theta \max(\tau, \tau^2) \simeq 1$. The optical depth requires knowing the electron density in the dispersive region. Typical optical depths values for rich clusters are around $0.02 - 0.03$.

A measure for the distortion induced by a scattering region is the Compton$-y$ parameter (or simply comptonization) defined as

$$y = \int \Theta \, d\tau = \int \Theta n_e \sigma_\mathrm{T} \, d\ell \tag{1.18}$$

where Θ and τ are, in general, functions of the path length measured by $\ell = ct$. When $y \gtrsim 1$, the total spectrum and total photon energy is significantly altered (unsaturated comptonization); whereas for $y \ll 1$, the total energy is not much changed (modified blackbody). This parameter can be interpreted as being proportional to the electron gas pressure $n_e kT$.

The occupation number change is sometimes expressed as a function of y, so after integrating both sides of Eq. (1.17) along the path length throughout the scattering region, considering Eq. (1.18), one gets a complete solution to the Boltzmann equation:

$$\Delta n(x, y) = \frac{y}{x^2} \frac{\partial}{\partial x} \left[x^4 \left(\frac{\partial n}{\partial x} + n(n+1) \right) \right] \tag{1.19}$$

For black-body radiation, as it is the case of CMB, with temperature T the occupation number of photons is actually $n\left(x\right) = \frac{1}{e^x - 1}$, which corresponds to the previous assumption that it is Planckian.

1.4.5 Sunyaev-Zel'dovich Effect

Because $n(x)$ constitutes a steady-state solution, the parenthesis in Eq. (1.19) vanish resulting in $\Delta n = 0$. However, CMB radiation field has its peak intensity at the low-frequency regime making it possible to approximate $\partial_x n \gg n(n+1)$. By doing so, Eq. (1.19) simplifies to

$$\Delta n(x,y) = \frac{y}{x^2}\frac{\partial}{\partial x}\left[x^4\frac{\partial n}{\partial x}\right] \tag{1.20}$$

and the photon occupancy change is readily solved.

$$\Delta n(x,y) = y\frac{xe^x}{(e^x-1)^2}\left[x\coth\left(\frac{x}{2}\right)-4\right] \tag{1.21}$$

Nonetheless occupation number is not an observable quantity but the spectral intensity. Given that Eq. (1.19) has been integrated along the path length throughout the scattering region, spectral intensity is given along the line-of-sight. Recalling the relationship between spectral intensity and occupation number $I = i_0 x^3 n(x)$, the net change in intensity is

$$\Delta I_t = i_0\, y\, \frac{x^4e^x}{(e^x-1)^2}\left[x\coth\left(\frac{x}{2}\right)-4\right] \qquad \text{with} \qquad i_0 = 2\frac{(kT)^3}{(hc)^2} \tag{1.22}$$

Taking the fractional change in intensity of CMB photons due to free electrons in the ICM, thermal Sunyaev-Zel'dovich (tSZ) is

$$\frac{\Delta I_t}{I} = y\frac{xe^x}{e^x-1}\left[x\coth\left(\frac{x}{2}\right)-4\right] \qquad \text{or} \qquad \boxed{\frac{\Delta T_t}{T} = 2y\left[\frac{x}{2}\coth\left(\frac{x}{2}\right)-2\right]} \tag{1.23}$$

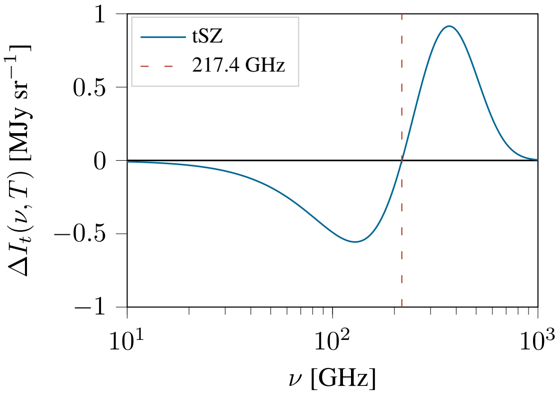


Figure 1.5: TSZ spectrum corresponding to $y = 0.0005$ with cut frequency at 217.4 GHz.

From x's definition it turns out that the scale in frequency is around GHz such that the effect is noticeable.

$$x = 1.762 \left(\frac{\nu}{100\,\mathrm{GHz}} \right) \tag{1.24}$$

Equation (1.22) has a zero at $x_0 = 3.83$ or $\nu_0 = 217.4\,\mathrm{GHz}$. Furthermore, ΔI_t is negative for values below x_0 and positive for values above it. This means that intensity decreases in the low frequency region of the spectrum, and increases in the high frequency region (see Fig.1.5). In the Raleigh-Jeans limit tSZ temperature contrast approximates simply to $-2y$.

On top of the spectral distortion caused by scattering the contribution of the bulk motion of the ICM relative to LF must be taken into account. Starting from the radiative transfer equation (Birkinshaw, 1999):

$$c^{-1}\frac{dI}{dt} = j_\nu - \left(\alpha_{\nu,\mathrm{abs}} + \alpha_{\nu,\mathrm{sca}}\right) I(\hat{\mathbf{k}}_1) + \alpha_{\nu,\mathrm{sca}} \int \phi(\hat{\mathbf{k}}, \hat{\mathbf{k}}_1) I(\hat{\mathbf{k}}) d\Omega \tag{1.25}$$

where j_ν is the emission coefficient; $\alpha_{\nu,\mathrm{abs}}$ and $\alpha_{\nu,\mathrm{sca}}$ are the absorption coefficients due to simple absorption and scattering, respectively; and ϕ is the probability of scattering from an angle $\mu = \cos\theta$ to an angle $\mu_1 = \cos\theta_1$ specified by the wave vectors $\hat{\mathbf{k}}$ and $\hat{\mathbf{k}}_1$ (given by Chandrasekhar, 1950).

$$\phi(\mu, \mu_1) d\mu = \frac{3}{8}\left[1 + \mu^2\mu_1^2 + \frac{1}{2}\left(1-\mu^2\right)\left(1-\mu_1^2\right)\right] d\mu \tag{1.26}$$

Considering the change in specific intensity solely by scattering, i.e. no emission nor absorption, and changing to a dependence on the optical depth along the path length in line-of-sight direction, the transfer equation reduces to

$$\frac{dI}{d\tau} = \int_{-1}^{1} \phi(\mu, \mu_1) \left(I - I_1\right) d\mu \tag{1.27}$$

By integrating the left side of Eq. (1.27) it's possible to rewrite it as a relativistic invariant $\frac{\Delta I}{I}$ where I corresponds to the CMB radiation field spectrum just as before. Since the observer at LF detects scattered photons along the line-of-sight, $\mu_1 = 1$.

$$I(\tau, \mu_1) - I(0, \mu_1) = \tau\, I(0, \mu_1) \int_{-1}^{1} \phi(\mu, \mu_1) \left(\frac{I(0,\mu)}{I(0,\mu_1)} - 1 \right) d\mu$$
$$\frac{\Delta I}{I} = \frac{3}{8}\,\tau \int_{-1}^{1} \left(1+\mu^2\right) \left(\frac{e^x - 1}{e^{x''} - 1} - 1 \right) d\mu$$

where $x'' = x\gamma^2(1+\beta)(1-\beta\mu)$. For small velocities ($\beta \ll 1$) the integrand can be expanded in powers of β and the symmetry of the integrand ensures that only even powers of μ in the expansion will appear in the result. Integration over all angles is easily done, so that the kinematic Sunyaev-Zel'dovich (kSZ) equation is

$$\frac{\Delta I_k}{I} = -\tau\beta\frac{xe^x}{e^x-1} \qquad \text{or} \qquad \boxed{\frac{\Delta T_k}{T} = -\tau\frac{v}{c}} \tag{1.28}$$

Because Eq. (1.28) describes bulk motion, the velocity parameter $\beta = \frac{v}{c}$ and optical depth τ represent the full ICM or galaxy cluster for that matter. Moreover, v is the line-of-sight peculiar velocity because it depends on the observer's relative motion to the ICM. In the low-frequency regime both temperature contrasts for tSZ and kSZ depend on the comptonization parameter y and optical depth τ, respectively. For this reason parameter estimates or measurements are needed in order to study SZ effects (see chapter 3).

The peak intensity of kSZ is found at the cross-over frequency x_0. However the relative intensity of kSZ to tSZ is very small. In fact, the ratio of the brightness temperature changes caused by the effects is

$$\frac{\Delta T_k}{\Delta T_t} = \frac{1}{2\Theta}\frac{v}{c} = 0.085\left(\frac{v}{1000\,\mathrm{km\,s^{-1}}}\right)\left(\frac{kT}{10\,\mathrm{keV}}\right)^{-1} \tag{1.29}$$

which is small for the expected velocities of a few hundred $\mathrm{km\,s^{-1}}$ or less, and typical cluster temperatures of a few keV. This makes it very difficult to locate kSZ in the presence of tSZ at low frequency. However, both effects may be separated using their different spectra. Indeed, kSZ peak intensity change (Fig. 1.6) is at the same frequency where tSZ is zero (Fig. 1.5).

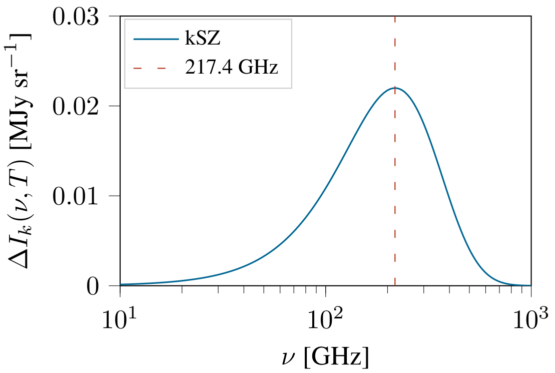


Figure 1.6: KSZ spectrum corresponding to $\beta = -0.0015$ ($v \approx -450\,\mathrm{km\,s^{-1}}$) approaching to the observer with maximum at the crossover frequency $217.4\,\mathrm{GHz}$.

1.4.6 Relativistic Limit

It has been found that non-relativistic electrons produce tSZ and kSZ temperature contrasts $\frac{\Delta T}{T}$ equal to $-2y$ and $-\tau\beta$, respectively, for low-energy photons. However, a proper treatment of the most general physical situation is far more challenging as there's no closed form solution. Nevertheless it's always essential to check for relativistic corrections to the classical theory in order to decide which approximation is more convenient for the study at hand.

Using the exact probability distribution and relativistically correct form of the electron velocity distribution (Maxwellian), Rephaeli (1995) calculated the resulting intensity change in the limit of small optical depth to Thomson scattering, τ, keeping terms linear in τ. The resulting frequency shift is

$$s = \ln\left(\frac{\nu_1}{\nu}\right) = \ln\left(\frac{1+\beta\mu_1}{1+\beta\mu}\right) \tag{1.30}$$

where β is the electron velocity in the CMB frame. The probability that a scattering results in a frequency shift s is (Wright, 1979)

$$\mathcal{P}(s,\beta) = \frac{1}{2\gamma^4\beta}\int \frac{e^s\,\phi(\mu,\mu_1)}{(1+\beta\mu)^2}d\mu \tag{1.31}$$

where $\phi(\mu,\mu_1)$ is given by Eq. (1.26). Averaging over a Maxwellian distribution for the electrons

$$\mathcal{P}_1(s) = \frac{\int \beta^2\gamma^5 \exp\left\{-\frac{\gamma-1}{\Theta}\right\}\mathcal{P}(s,\beta)d\beta}{\int \beta^2\gamma^5 \exp\left\{-\frac{\gamma-1}{\Theta}\right\}d\beta} \tag{1.32}$$

Finally, the total change in photon occupation number along a line of sight to the cluster can now be written as

$$\Delta n_t(x) = \tau\int_{-\infty}^{\infty}\left[n(xe^s) - n(x)\right]\mathcal{P}_1(s)ds \tag{1.33}$$

At a glance, this procedure already shows mayor differences to Eq. (1.21) integrated from the Kompaneets equation. The most evident is the explicit dependence on the optical depth instead of the comptonization parameter. The other major difference is that the problem now is to solve an integral equation rather than a second order differential equation.

Regardless of how $\Delta n_t(x)$ is calculated, the intensity change is $\Delta I_t(x) = i_0 x^3 \Delta n_t(x)$. Introducing the spectral functions $h(x)$ and $g(x)$, the non-relativistic intensity spectra for both effects can be easily rewritten as $\Delta I_t = i_0\, yg(x)$ for tSZ and $\Delta I_k = -i_0\,\tau\beta\, h(x)$ for kSZ.

The dependence on the cluster gas density n_e is encoded on both y and τ parameters.

$$h(x) = \frac{x^4 e^x}{(e^x - 1)^2} \qquad \text{and} \quad g(x) = h(x)\left[x \coth\left(\frac{x}{2}\right) - 4\right] \tag{1.34}$$

The exact relativistic calculation does not lead to a simple analytic solution so, in order to obtain an approximate expression, the formal expression for $\Delta I = \Delta I_t + \Delta I_k$ needs to be expanded in powers of (the small quantities) τ, Θ, and β. For the resulting expansion to be accurate to within ∼2% for $kT < 50\,\mathrm{keV}$, Shimon & Rephaeli (2004) included terms up to $\tau\Theta^{12}$, $\tau^2\Theta^5$, and $\beta^2\Theta^4$. However, since cluster velocities are expected to be generally below $1000\,\mathrm{km\,s^{-1}}$, terms quadratic in β can be ignored. Doing so gives the total intensity change as the sum

$$\frac{\Delta I}{i_0} = \tau \sum_{i=1}^{8} f_i(x)\Theta^i \;+\; \tau^2 \sum_{i=1}^{4} f_{i+8}(x)\Theta^{i+1} \;-\; \tau\beta\left[h(x) + \sum_{i=1}^{4} f_{i+12}(x)\Theta^i\right] \tag{1.35}$$

where $f_i(x) = x^3 F_i(x)$, with $F_i(x)$ defined in Shimon & Rephaeli (2004). The crossover frequency x_0 varies accordingly with Θ and τ:

$$x_0 = 3.83\left(1 + 1.12\,\Theta + 2.08\,\Theta^2 - 80.74\,\Theta^3 + 1548.25\,\Theta^4 + 0.8\,\tau\Theta + 1.18\,\tau\Theta\right) \tag{1.36}$$

Consider, for example, a typical cluster temperature of $10\,\mathrm{keV}$. Because kSZ is the signal of interest for this work, the first order correction is in Θ regulated by the spectral function $f_{13}(x)$ (retrived from Shimon & Rephaeli, 2004, and reworked for ease of presentation)

$$f_{13}(x) = x^3\left[-\frac{9}{4}\frac{x}{\sinh^2\frac{x}{2}} + \frac{47}{20}\frac{x^2\cosh\frac{x}{2}}{\sinh^3\frac{x}{2}} - \frac{7}{40}\frac{x^3\left(2\cosh^2\frac{x}{2} + 1\right)}{\sinh^4\frac{x}{2}}\right]$$

It goes without saying that the functional relationship between spectral intensity I and frequency x is far more complicated, however the peak only rose by 10% and the cross-over frequency is merely shifted to 222.1 GHz (see Fig. 1.7).

Be as it may, the theoretical description accuracy of the SZ effect surpasses the precision with which the effect can be measured nowadays. In addition to observational errors, systematic modelling uncertainties are relatively large. Thus, the impact of the relativistic treatment for the purpose of measuring peculiar velocities introduces negligible differences. For this reason, the first order approximation on β for the kSZ temperature contrast in Eq. (1.28) suffices as a model for cluster peculiar velocities.

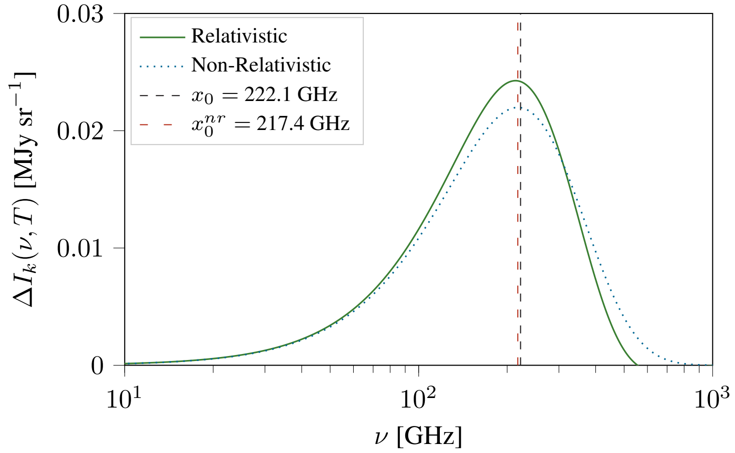


Figure 1.7: Relativistic kSZ spectrum with first order correction in Θ.

Chapter 2 – Deep Learning

Machine Learning (ML) is the field of study that gives computers the ability to *learn* without being explicitly programmed (Samuel, 1959). As the name suggests, ML is concerned with the problem of how to construct computer programs that automatically improve with experience. Mitchell (1997) identifies three elements for the learning process: experience, tasks, and performance measure. A computer program is said to learn from **experience** E with respect to some class of **tasks** T and **performance measure** P, if its performance at tasks in T, as measured by P, improves with experience E. Mitchell's goal is not to analyse the meaning of the English word "learning" as it is used in everyday language, but to define precisely a class of problems that encompasses interesting forms of learning, to explore algorithms that solve such problems, and to understand the fundamental structure of learning problems and processes.

The present work aims to fit numerical values to images, meaning that the task to perform is regression. And, because the data available is both image and desired numerical value (see chapter 3), supervised learning is the most appropriate type of experience. This chapter will introduce the pertinent general definitions and concepts required to work seemingly with ML. The program's performance can be evaluated with respect to the desired value, however this is properly addressed in the next chapter.

2.1 Supervised Learning

The design of a learning system requires the following steps. Firstly, fix a task to perform, which refers to the exact type of knowledge to be learned (e.g. play checkers, classify images, learn how to drive, etc.). Next, choose a training experience (e.g. supervised learning). The type of training experience can have a significant impact on the success or failure of the learner, and it basically depends on the training examples available. Are they a direct representation of the desired data? Do they provide a broad distribution of cases or situations? In general, learning is most reliable when the training examples follow a distribution similar to that of future test examples. Lastly, set a performance measure (e.g. percent of correct classifications) to know how well the program is doing, whether it provides direct or indirect feedback regarding the choices made by the performance system.

Although there are other types of learning experiences such as unsupervised and reinforced learning, supervised learning is used for classification tasks that involve the prediction of class labels (categorical data) and regression tasks that involve the prediction of a numerical label. In supervised learning the target for every training example is known *a priori* and is called hypothesis as it's supposed to explain the corresponding one. This type of learning can be implemented to automatize complex tasks of classification or regression, and thrives with a large volumes of examples.

Consider the set of n real values x_{ij} ($\mathbb{R}$ are the real numbers), called features, representing an instance $\mathbf{X}_i$ in Eq. (2.1). An instance is a training example or particular case of the data to be fitted or classified. It can be anything from a set of points, images, words, measurements, etc. Then, the instances dataset given by Eq. (2.2) is conformed by *all* available instances[1].

$$\mathbf{X}_i = \{x_{ij} \in \mathbb{R} \,|\, j = \overline{1, n}\} \tag{2.1}$$

$$\mathbb{X} = \{\mathbf{X}_i \mid i = \overline{1, N}\} \tag{2.2}$$

Features in $\mathbf{X}_i$ can be arranged into different structures depending on how many dimensions the instance has. For example, a full colour image ($\mathbf{X}_i$) can be represented by a 3-D array with every colour pixel being a different feature, and colour channels stacked in the third dimension of the array. The concept of tensors are a natural option to represent arbitrary dimensional arrays (TensorFlow, 2015). In fact, ML techniques exploit this concept and represent all datasets, variables and procedures involved in a computation as tensors.

To every instance corresponds a known value, label or target $\hat{y}$ representing the hypothesis. A total of N pairs of the form instance-target complete the training set. These pairs are expected to be *good* representations of the observations and follow a distribution similar to future examples. Moreover, N is expected to be very large, of at least a few thousand. It is common to find that complex examples require large datasets so if the example number is small it may become difficult to train a reliable learning algorithm. In supervised learning the targets dataset can be defined containing the *a priori* target label corresponding to every instance in $\mathbb{X}$:

$$\hat{\mathbb{Y}} = \{\hat{y}_i \in \mathbb{R} \mid i = \overline{1, N}\} \tag{2.3}$$

[1] The index i is explicitly written for enumerating elements of a set. When omitted, the variable refers to any arbitrary instance, feature or target.

When classifying amongst several categories, say n_{cat}, each label can be encoded to an integer representing it, where the last category's is n_{cat}. Some algorithms have faster convergence when the labels are in vectorized with the canonical base for $\mathbb{R}^{n_{\text{cat}}}$, with each category corresponding to a different base vector. For regression tasks the most obvious choice for the type of information to be learned is a program or function that can process every instance in order to produce a single real scalar. Call this objective function $\hat{f}$ and use the notation $\hat{f} : \mathbb{X} \mapsto \hat{\mathbb{Y}}$ to indicate that this function accepts as input any instance from the set of examples $\mathbb{X}$ and outputs some value in the targets dataset $\hat{\mathbb{Y}}$. Additionally, assume there's a direct relation between instance and target, and the objective function ideally represents it.

$$\begin{aligned} \hat{f} : \mathbb{X} \mapsto \hat{\mathbb{Y}} \\ \hat{f}(\mathbf{X}) = \hat{y} \end{aligned} \tag{2.4}$$

Because the explicit definition of $\hat{f}$ is not known, therefore is not computable by a program, it is said to have a non-operational definition. The goal of learning in this sense is to discover an operational description of $\hat{f}$; that is, it can be used by the program in order to evaluate states and select the best fit. Thus, the learning task has been converted to the problem of discovering an operational description of the objective function. However, it's expected that learning algorithms acquire only some approximation to the ideal function $\hat{f}$.

The function that is actually learned by the program is called prediction function, denoted f. Representation of this function is a key design choice and, in general, involves a crucial trade-off. On one hand, an expressive representation allows for a better approximation to the objective function; on the other, more expressiveness implicates that the program will need larger training datasets in order to choose between the different hypotheses it can represent. Ultimately, a program achieves *true knowledge* when the prediction function is consistent with the training dataset, i.e $f(\mathbf{X}) = \hat{f}(\mathbf{X})$ for all $\mathbf{X} \in \mathbb{X}$, but this is practically not achievable for very complex tasks.

The action of performing the learning task via f will output a single prediction y for every instance. Defining the predictions dataset $\mathbb{Y}$ as the set of all predicted values:

$$\mathbb{Y} = \{y_i \in \mathbb{R} \mid i = \overline{1, N}\} \tag{2.5}$$

In order to compute f a parameter set $\boldsymbol{\theta}$ is required . This set defines the choice of representation and the expressiveness of f is embedded in the relationships between features and parameters, which can be very complex as it'll be shown later in sections 2.2 to 2.4. The prediction function is a mapping from the instances dataset to the predictions dataset governed by the parameter set:

$$\begin{aligned} f : \mathbb{X} &\mapsto \mathbb{Y} \\ f(\mathbf{X}; \boldsymbol{\theta}) &= y \end{aligned} \tag{2.6}$$

Note that y depends directly on the choice of $\boldsymbol{\theta}$, which represents the current *knowledge* of the task. From now on "learn" and/or "learning" will also refer to the determination of the parameter set $\boldsymbol{\theta}$. A change in the parameter values modifies the prediction y and, in turn, the set. This means that f can be evaluated with respect to every choice of $\boldsymbol{\theta}$ and compared for consistency with the training examples. In this sense, the predictions dataset is not fixed until the end of the learning process when it becomes the model's output. Hopefully, as the program trains, predictions improve accordingly.

Equation (2.6) is written explicitly dependent of the parameter set to highlight the fact that its output corresponds to that particular choice of values. There are many possible choices of f to represent the objective function. For example, a large table specifying a value for each input, or a set of rules to match with each state, or a parametric function with embedded data features, or an artificial neural network. In practice, this can never be assured so it's necessary to monitor the training and stop when it has achieved a certain level of accuracy.

The quantification of *goodness* or *badness* for a particular set of parameters $\boldsymbol{\theta}$ is done by a performance measure, called loss function $\mathcal{L}$. It maps each pair of prediction and target $(y, \hat{y})$ corresponding to the same instance to a single real value representing a prediction score for the current knowledge. It's also commonly referred to as prediction error however the term loss gives a more general description.

$$\begin{aligned} \mathcal{L} : (\mathbb{Y}, \hat{\mathbb{Y}}) &\mapsto \mathbb{R} \\ \mathcal{L}\left(y, \hat{y}\right) &= \mathcal{L}\left(\mathbf{X}, \hat{y}; \boldsymbol{\theta}\right) \end{aligned} \tag{2.7}$$

The common approach is to select one which minimizes the error between training and predicted values, e.g. least mean squares. For categorical data, softmax cross entropy loss gives the probability of an instance belonging to a category and checks if it's equal to the target category. The loss function is computed for every instance but a more general value can quantify the total cost of the operation as the expected value of individual losses. To prevent from fitting just training data, a regularization term $\mathcal{R}(\boldsymbol{\theta})$ can be added to the cost as a penalty

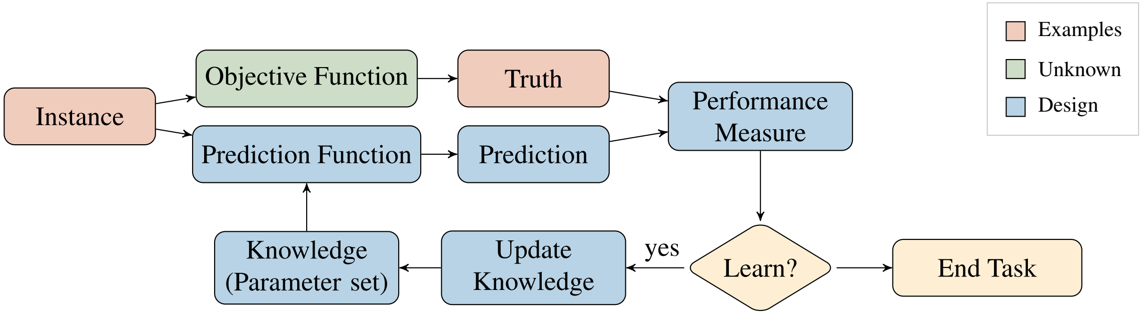


Figure 2.1: Supervised learning scheme for a single instance. Red nodes constitute the examples datasets; green nodes, never known values; and blue, design choices. The instance is processed by the objective and prediction functions. Although the former is unknown its output isn't (given by $\hat{y}$). The performance measure scores the current task and the algorithm uses it for knowledge updating.

that encourages the program to pick a simpler $\boldsymbol{\theta}$. There are other schemes like dropout methods (inverse jackknife) that allow to reduce the number of parameters and chance to over-fit. The final step in learning, is to take this performance measure and *update* the program's knowledge of the task.

Figure 2.1 shows a chart for supervised learning process. For every instance both objective and prediction functions output a target and a prediction value that are later compared by the performance measure. Then, the learning algorithm takes place by updating the knowledge of the task according to the performance measure. After that, a new prediction can be made and scored once again by the performance measure. This process is repeated for all examples in the training set until consistency is achieved, or (most likely) either some level of accuracy is reached by the program or a certain amount of time has been invested.

2.2 Artificial Neural Networks

The study of artificial neural networks (ANNs) has been inspired in part by the observation that biological learning systems are built of very complex webs of interconnected neurons. To develop a feel of this analogy consider a few facts from neurobiology. The biological neuron is comprised by a soma or cellular body and axons which connect to other neurons soma. The membrane has a potential threshold (determined by the neuron) that an excitation from external signals must exceed to produce a synapse. Inspired by the biological neuron, McCulloch & Pitts (1943) proposed a simple model for an artificial neuron (AN) which consists of one or several binary inputs computed as a linear combination, and one binary output.

While ANNs are loosely motivated by biological systems in reality neurons are far more complex and form networks beyond computational capabilities (to this date). Input signals are not necessarily processed in a linear manner which adds another complication. Moreover, neurons do not fire a single output but a spike train with encoded information. This model has been known for a long time but has gained popularity in the last decade or two. With improvements in techniques and technology, DL implementation for problem solving has escalated exponentially with the AN at its core.

The AN model uses one of the simplest representations: a linear combination of inputs or features. Conveniently extending the number of features in a single instance by fixing a new input x_0 to some real constant (usually 1), it becomes easy to write:

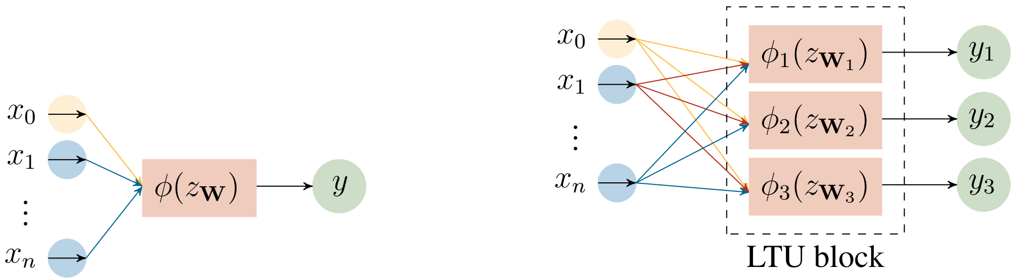
$$z\left(\mathbf{X};\boldsymbol{\theta}\right) = \sum_{j=0}^{n} x_j w_j \tag{2.8}$$

where $x_j \in \mathbf{X}$ for $j = \overline{0,n}$, and the parameter set is simply $\boldsymbol{\theta} = \{w_j \in \mathbb{R} \mid j = \overline{0,n}\}$. The weights will determine the relative influence of the various features when computing the prediction function value, whereas the product $x_0 w_0$ provides a bias.

Equation (2.8) looks awfully similar to linear regression because the output is a function of the weighted sum $z_{\mathbf{W}} = z\left(\mathbf{X};\boldsymbol{\theta}\right)$. The main difference is an extra layer of complexity allowing a neuron to modify its response behaviour. In analogy with the biological neuron membrane potential threshold, an activation function $\phi(z_{\mathbf{W}})$ receives the weighted sum as input then outputs a single value representing that neuron's response and, in doing so, simulates how a neuron activates or fires. The output value is a prediction, i.e. the prediction function itself $f(\mathbf{X};\boldsymbol{\theta}) = \phi(z_{\mathbf{W}})$. Figure 2.2a is a sketch of an AN.

That's all there is to an AN because the next step falls into the learning process. The loss function $\mathcal{L}$ will take a single neuron output value, compare it to the target and give a *score* that can be used to update the parameter set $\boldsymbol{\theta}$. Updating the parameter set is not trivial nor a simple task and it will be covered in Section 2.3.

When ANs activity response is all-or-none it's called linear threshold unit (LTU), which uses the Heaviside or step function for binary output. Non-linear functions can be used depending on the working model and the expressiveness desired. The most commonly used are sigmoid-like functions: logistic function, $\tanh(x)$, $\arctan(x)$, $\mathrm{erf}(x)$, etc.; softmax, softplus, rectified linear unit, and more. All these change the behaviour from a LTU to general AN.



(a) Artificial neuron (AN) schematic. A vector with n entries (plus a bias) is input, then the weighted sum $z_{\mathbf{W}}$ is passed to the activation function ϕ, which returns a value y.

(b) Simple ANN schematic with three stacked neurons. Single instance $\mathbf{X}$ fed to every LTU which process ϕ and outputs a value f independent from the other all others. The networks output is the ordered tuple $\mathbf{y}_i = (y_{i1}, y_{i2}, y_{i3})$, and the parameter set in this case is $\boldsymbol{\theta} = \bigcup \mathbf{W}_i$.

Figure 2.2: Artificial neuron and neuron layer schematics.

The AN model is able to solve AND, NAND, OR and NOR logic gates classification tasks, but fails with XOR (exclusive OR) and XNOR. It has been known since 1969 and illustrates the necessity of using several neurons to solve more complex problems. One option would be to make a sequence of neurons where the firsts output is sent to the next input and so on. Or stack parallel neurons into a layer, all receiving the same input, and passing all outputs as inputs for a next neuron, and so on. This is the base of DL. One thing to keep in mind is that the concatenation of linear models is also a linear model, i.e. the network collapses into a single neuron. To avoid this problem, activation functions must be used to remove linearity between layers and distort the outputs.

Take the perceptron for example (see Fig. 2.2b). It's a set of stacked LTUs where each one is connected to all inputs (the simplest perceptron has only one LTU). The idea of learning in a neuron is the same of that of biological neurons: when a neuron frequently activates another, the connection between them is reinforced. Perceptrons are trained to reinforce only those connections that give the correct result. The perceptron is capable of solving the XOR problem with only two neurons because each added neuron represents an additional straight line (a hyperplane in the most general case).

The perceptron is a linear classifier and so it does not give any information about the probability of an instance belonging to the output class due to the activation function. Also, it is not capable of classifying data that is not linearly separable. To achieve this, one can couple several perceptrons in concatenated layers, the resulting ANN is called multilayered perceptron (MLP) and is deepened by each additional perceptron between input and output. Every layer between input and output is called hidden layer and is a LTU fully-connected to the next one.

The most general case is reached when LTUs are replaced by arbitrary activation functions and linearity is lost. These are called deep neural network (DNN), characterized by an arbitrary number of densely populated hidden layers, where each one can have a different activation function and the relationships between layers is not necessarily sequential (e.g. recurrent neural network [RNN]).

2.3 Gradient Descent and Back-Propagation

Most deep learning algorithms involve optimization of some sort. This means minimizing or maximizing some cost function $J(\boldsymbol{\theta})$ that measures the error when using the model. The total cost of the operation can be taken as the expected value of per-instance losses.

$$J(\boldsymbol{\theta}) = \mathbf{E}_{\mathbb{X},\mathbb{Y}}\left[\mathcal{L}\left(\mathbf{X}, \hat{y}; \boldsymbol{\theta}\right)\right] = \frac{1}{\#(\mathbb{X})} \sum_{\substack{\mathbf{X} \in \mathbb{X} \\ \hat{y} \in \hat{\mathbb{Y}}}} \mathcal{L}\left(\mathbf{X}, \hat{y}; \boldsymbol{\theta}\right) \tag{2.9}$$

where $\#(\mathbb{X})$ is the cardinality of the set. The linear regression algorithm, for example, uses mean squared error (MSE) as cost, i.e. the loss function is $\mathcal{L} = \left(\mathbf{X} \cdot \mathbf{W} + w_0 - \hat{y}\right)^2$. The parameter set represents a model in itself so, in order to learn, the cost must be optimized via differentiation with respect to every parameter and, because the cost is an error measure, must be in fact minimized.

Why? The derivative gives the slope of a function at a certain point. The derivative is therefore useful for minimizing a function because it specifies how to scale a small change to the input in order to obtain the corresponding change in the output. For linear regression, the cost is somewhat easily differentiated because of its simple form. In general, however, most cost functions cannot be optimized in closed form, requiring an alternative numerical procedure.

The derivative gives the slope, but the directional derivative tells the direction of greatest change in a multidimensional space. Assume a function $F(\boldsymbol{\theta} + \alpha\mathbf{u})$, then the directional derivative in direction of the unit vector $\mathbf{u}$ is $\partial_\alpha F(\boldsymbol{\theta} + \alpha\mathbf{u})|_{\alpha=0}$, which simply evaluates to $\mathbf{u}^{\mathrm{T}} \cdot \nabla_{\boldsymbol{\theta}} F(\boldsymbol{\theta})$. To minimize the function F one must find the direction in which F decreases the fastest.

$$\min\left(\mathbf{u}^{\mathrm{T}} \cdot \nabla_{\boldsymbol{\theta}} F(\boldsymbol{\theta})\right) = \min\left(||\nabla_{\boldsymbol{\theta}} F(\boldsymbol{\theta})|| \cos\vartheta\right)$$

where ϑ is the angle between $\mathbf{u}$ and the gradient. This expression simplifies to $\min\left(\cos\vartheta\right)$, which is minimized when $\mathbf{u}$ points in the opposite direction of the gradient. In other words, the gradient points directly *uphill* and the negative directly *downhill*. This is known as the method of steepest descent or gradient descent (GD).

The idea is to find the local gradient in the parameter space, *move* in the opposite direction of the gradient to a new point assumed to be closer to a local minimum. The process is repeated until convergence, when every element in the gradient is zero. In the context of DL, functions may have many local minima that are not optimal, and many saddle points surrounded by very flat regions making optimization and convergence very difficult to perform and achieve. Therefore, one settles for finding a value of F that is very low, but not necessarily minimal in any formal sense. GD proposes a new point $\boldsymbol{\theta}'$ given by:

$$\boldsymbol{\theta}' = \boldsymbol{\theta} - \epsilon \nabla_{\boldsymbol{\theta}} F(\boldsymbol{\theta})$$

where ϵ is the learning rate, a positive scalar determining the size of the step. It's possible to choose ϵ in several different ways. One is to set ϵ to a small constant, while another is to solve $F(\boldsymbol{\theta} - \epsilon \nabla_{\boldsymbol{\theta}} F(\boldsymbol{\theta}))$ for several values of ϵ and choose that which results in the smallest objective function value. The learning rate is a hyperparameter[2] of a NN because it controls how the algorithm will update parameter values throughout the learning process.

IF ϵ is large, the algorithm will quickly move around the parameter (multidimensional) space but can overpass local minima or move away from one and never converge. If ϵ is very small, the algorithm will be able to sweep thoroughly at the expense of higher iterations and rendering inefficient the search for a local minimum. It is possible to improve ϵ dynamically changing its value, making it bigger when the gradient is big and smaller when the gradient is small.

As stated before, the function to be optimized in the context of DL is the cost $J(\boldsymbol{\theta})$. Meaning that for every step of GD the per-instance loss is calculated for every instance in order to know the *goodness* (or *badness* depending on the readers half-full half-empty view of things) of the current model, and averaging over all losses. For the cost function in Eq. (2.9), GD requires computing:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbf{E}_{\mathbb{X},\mathbb{Y}} \left[\nabla_{\boldsymbol{\theta}} \mathcal{L} \left(\boldsymbol{\theta}; \mathbf{X}, \hat{y} \right) \right] = \frac{1}{\#(\mathbb{X})} \sum_{\substack{\mathbf{X} \in \mathbb{X} \\ \hat{y} \in \hat{\mathbb{Y}}}} \nabla_{\boldsymbol{\theta}} \mathcal{L} \left(\boldsymbol{\theta}; \mathbf{X}, \hat{y} \right) \tag{2.10}$$

[2] The values of hyper-parameters are (in principle) not adapted by the learning algorithm itself.

The computational expense is of order $N \cdot \#(\boldsymbol{\theta})$ so, as the training sample size grows, the time to take a single gradient step becomes unavoidably long. The insight in GD is that the gradient is an expectation value, which can be approximately estimated using a small set of samples. Imagine that the complete dataset $\mathbb{X}$ is sampled in non-intersecting minibatches of instances $\mathbb{B}$ drawn uniformly.

$$\mathbb{B} = \{\mathbf{X}_{i'} \in \mathbb{X} \mid i' = \overline{1, M},\ M < N\} \subset \mathbb{X} \tag{2.11}$$

$$\hat{\mathbb{Y}}_{\mathbb{B}} = \{\hat{y}_{i'} \in \hat{\mathbb{Y}} \mid i' = \overline{1, M},\ M < N\} \subset \hat{\mathbb{Y}} \tag{2.12}$$

The minibatch size M is usually very small compared to the total number of instances N, ranging from one to a few hundred. Also, M is held fixed as the training set size N grows in order to let the computational effort stay fixed for a single cost estimation, increasing only the number of minibatches. The estimate of the gradient is formed as:

$$\mathbf{g} = \frac{1}{\#(\mathbb{B})} \sum_{\substack{\mathbf{X} \in \mathbb{B} \\ \hat{y} \in \hat{\mathbb{Y}}_{\mathbb{B}}}} \nabla_{\boldsymbol{\theta}} \mathcal{L}\left(\mathbf{X}, \hat{y}, \boldsymbol{\theta}\right) \tag{2.13}$$

using samples from the minibatch $\mathbb{B}$. The stocastic gradient descent (SGD) algorithm then follows the estimated gradient downhill:

$$\boldsymbol{\theta} \ \leftarrow \ \boldsymbol{\theta} - \epsilon \mathbf{g} \tag{2.14}$$

Optimization methods that use the entire training set are called deterministic or batch gradient descent (BGD) methods because they process all of the training samples simultaneously in one large batch. Methods that use minibatches or a single instance at a time are called minibatch stochastic gradient descent or simply SGD methods. The main advantage of BGD is that the cost is calculated for all training instances, representing *full knowledge of all known possibilities*[3], but has serious memory issues for large datasets because the complete set must be loaded in cache. SGD takes less memory but can take longer to converge because the cost represents only the current minibatch. To minimize this problem, a large dataset is required with lots of minibatches. Additionally, the most common approach is SGD with minibatches and usually performs very similar to BGD. Memory issues are of great concern most of all when working with high-resolution image data which can take a lot of memory per instance.

[3] This is a manner of speak because the idea is to use trained models to process both unknown and new information.

The basis of a feed-forward NN have been laid out up to this point: inputs provide initial information that flows forward through the network, then propagates up to the hidden units at each layer and finally produces an output. This is called forward propagation and can continue onward until it produces a scalar $J(\boldsymbol{\theta})$. The cost is needed to update all parameter values, so the information has to flow backwards through the network in order to compute the gradient. This backward pass is called back-propagation.

In a nutshell, back-propagation is an algorithm for determining how a single training instance would like to *nudge* the parameters in terms of what relative proportions to those changes cause the most rapid decreases for the cost function. Back-propagation enables to compute the gradient of any function, not specific to the cost. Because the prediction is the composition of functions $\left(\mathcal{L} \circ f\left(\mathbf{X}, \hat{y}\right)\right)[\boldsymbol{\theta}]$ applied to the input, there's a chain of responsibility from the output of a single neuron back to the input parameters.

Consider a DNN with L layers where the ℓ-th layer has m_ℓ neurons. In the following, assume that the activation function of the k-th neuron in the ℓ-th layer is given by $f_{\ell k}\left(\boldsymbol{\theta}; \mathbf{X}, \hat{y}\right) = \phi_{\ell k}(z_{\ell k}(\boldsymbol{\theta}))$, i.e. is a function of the parameter set $\boldsymbol{\theta}$ with a fixed instance. Note that the parameter set $\boldsymbol{\theta}$ contains *all* parameters from *all* neurons in the model, not only those corresponding to the (ℓ, k) neuron. Now, compute the loss for the ℓ layer.

First: call $\mathbf{A}_L$ the output values for layer $\ell = L$, which is clearly $\mathbf{A}_L = y$, with $\#(\mathbf{A}_L) = m_L$ the number of values needed to make a prediction for instance $\mathbf{X}$. Layer L's input values are those outputted by the previous layer, i.e. $\mathbf{A}_{L-1}$, which in turn has inputs $\mathbf{A}_{L-2}$; and so on and so forth. This is done recursively until $\ell = 1$ where its inputs are given by $\mathbf{A}_0 = \mathbf{X} \cup \{x_0\}$ with $m_0 = n + 1$ (the bias x_0 adds one parameter to the layer). Every layer has a bias

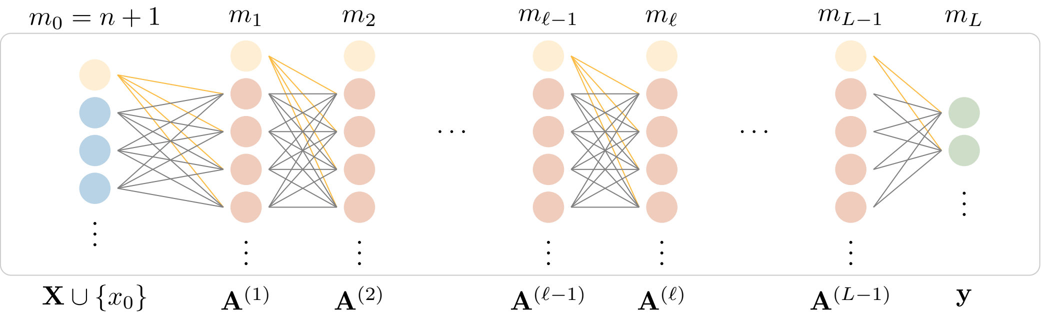


Figure 2.3: Sample DNN to explain back-propagation. Blue and green nodes are input and output (known) values; red nodes are hidden neuron layers; and yellow are the biases for every input. Note how each bias is independent of the previous layer and how the output layer doesn't have a bias value.

parameter independent from all other parameters in the network, except for the output layer. Including $\mathbf{A}_0$, there are a total of L different sets defined as follows:

$$\mathbf{A}_\ell = \{a_{\ell k'} \in \mathbb{R} \mid k' = \overline{0, m_\ell},\ m_\ell \in \mathbb{N}\} \tag{2.15}$$

Second: connect the k'-th neuron in layer $\ell - 1$ to the k-th neuron in layer ℓ. Then, all parameters in $\boldsymbol{\theta}$ are identified by the (ℓ, k, k') tuple. In matrix form, the weighted sum operation (like in Eq. (2.8)) between layers can be written as the matrix product $\boldsymbol{\Omega}_\ell \mathbf{A}_{\ell-1}$, where $\boldsymbol{\Omega}_\ell$ is the $m_\ell \times m_{\ell-1}$ connection matrix[4] between layers $\ell - 1$ and ℓ; and $\mathbf{A}_{\ell-1}$ is defined in Eq. (2.15). A forward pass through the network satisfies the following relation of recurrence:

$$\mathbf{A}_\ell = \{\phi_\ell\left(\mathbf{Z}_\ell\right) \in \mathbb{R} \mid \mathbf{Z}_\ell = \left(\boldsymbol{\Omega}_\ell \mathbf{A}_{\ell-1}\right)_k \in \mathbb{R},\ k = \overline{1, m_\ell}\} \cup \{a_{\ell 0}\} \tag{2.16}$$

The parameter set for the network can be explicitly defined:

$$\begin{aligned} \boldsymbol{\theta} &= \{\theta_{\ell k k'} \in \mathbb{R} \mid \ell = \overline{1, L},\ k = \overline{0, m_\ell},\ k' = \overline{0, m_{\ell-1}}\} \\ &= \{\boldsymbol{\Omega}_\ell \equiv (\theta)_{\ell k k'} \in \mathbb{M}_{m_\ell \times m_{\ell-1}} \mid \ell = \overline{1, L}\} \\ \#(\boldsymbol{\theta}) &= \sum_{\ell=1}^{L} m_\ell \left(m_{\ell-1} + 1\right) \end{aligned} \tag{2.17}$$

And third: the loss function definition is fixed and according to Eq. (2.10), the derivative with respect to a single parameter is the average over all derivatives of the per-instance loss for each training example $\mathcal{L}$. In this sense $\mathcal{L}$ is a random variable. Additionally, for very large N, both expectation value and arithmetic mean converge to the same value.

Allowing the activation function to be defined for every neuron (most general although not common case) so that $\phi_{\ell k} = a_{\ell k}$, the (ℓ, k, k')-th component of the cost's gradient is:

$$\begin{aligned} \frac{\partial J}{\partial \theta_{\ell k k'}} &= \mathbf{E}_{\mathbb{X},\mathbb{Y}}\left[\frac{\partial \mathcal{L}}{\partial \theta_{\ell k k'}}\right] \\ &= \mathbf{E}_{\mathbb{X},\mathbb{Y}}\left[\frac{\partial \mathcal{L}}{\partial \phi_{\ell k}} \frac{\partial \phi_{\ell k}}{\partial z_{\ell k}} \frac{\partial z_{\ell k}}{\partial \theta_{\ell k k'}}\right] \qquad \text{with} \qquad z_{\ell k} = \sum_{k'=0}^{m_{\ell-1}} \theta_{\ell k k'}\, a_{(\ell-1)k'} \end{aligned} \tag{2.18}$$

Assuming all parameters in the same layer are independent, the last derivative in the expectation value is simply $a_{(\ell-1)k'}$. The other two derivatives tell how by much $\mathcal{L}$ is modified by

[4] $\mathbb{M}_{m_\ell \times m_{\ell-1}}$ is the vector space of all matrices of shape $m_\ell \times m_{\ell-1}$.

a small change in the weighted sum. Call this layer loss or layer error:

$$\delta_{\ell k} = \frac{\partial \mathcal{L}}{\partial \phi_{\ell k}} \frac{\partial \phi_{\ell k}}{\partial z_{\ell k}} \tag{2.19}$$

It is a measure of the **responsibility** that the (ℓ, k)-th neuron has in the final result (the loss), and is common to all k' parameters. By doing this, the gradient's component is the expectation of the input value regulated (multiplied) by the layer error.

$$\frac{\partial J}{\partial \theta_{\ell k k'}} = \mathbf{E}_{\mathbb{X},\mathbb{Y}} \left[\delta^{(\ell k)} \; a_{(\ell-1)k'} \right] \tag{2.20}$$

Going back one more layer, the chain of responsibility for the k''-th parameter belonging to the $(\ell - 1, k')$-th neuron dictates:

$$\begin{aligned} \frac{\partial J}{\partial \theta_{(\ell-1)k'k''}} &= \mathbf{E}_{\mathbb{X},\mathbb{Y}} \left[\frac{\partial \mathcal{L}}{\partial \phi_{\ell k}} \frac{\partial \phi_{\ell k}}{\partial z_{\ell k}} \frac{\partial z_{\ell k}}{\partial \phi_{(\ell-1)k'}} \frac{\partial \phi_{(\ell-1)k'}}{\partial z_{(\ell-1)k'}} \frac{\partial z_{(\ell-1)k'}}{\partial \theta_{(\ell-1)k'k''}} \right] \\ &= \mathbf{E}_{\mathbb{X},\mathbb{Y}} \left[\delta_{\ell k} \; \mathbf{\Omega}_\ell \; \frac{\partial \phi_{(\ell-1)k'}}{\partial z_{(\ell-1)k'}} a_{(\ell-2)k''} \right] \end{aligned}$$

where an alternative definition to the connection matrix is $\mathbf{\Omega}_\ell = \frac{\partial z_{\ell k}}{\partial \phi_{(\ell-1)k'}}$. The error of that layer is then

$$\delta_{(\ell-1)k'} = \delta_{\ell k} \; \mathbf{\Omega}_\ell \; \frac{\partial \phi_{(\ell-1)k'}}{\partial z_{(\ell-1)k'}} \tag{2.21}$$

With the set of equations: Eq. (2.19) for the layer loss, Eq. (2.20) for the derivative with respect to a single parameter, and Eq. (2.21) as recurrence relation, the back-propagation algorithm is complete and can compute the full gradient of the cost function either Eq. (2.10) or Eq. (2.13).

A true GD step would take all the training instances and average the desired changes, but that's computationally slow. Instead most algorithms use minibatch gradients and compute a training step with respect to each minibatch. It's not going to be the complete gradient of the function but it gives a good approximation and a significantly computational speed-up.

Together, back-propagation and GD complete the scope for parameter optimization in basic DL algorithms. There are a lot of new algorithms that improve the efficiency in computing the gradient and optimize the learning rate, yet these two sit at the base of most (and this work's) NNs.

2.4 Convolutional Neural Networks

Amongst the plethora of NNs, deep neural networks (DNNs) and convolutional neural networks (CNNs) are the most common and illustrative models. Both can solve a lot of very complex problems and are simple enough to show the power of ML. At its most basic, both models are implementations of the previous concepts: neurons, layers, optimization methods. As it was previously discussed, a DNN is similar to a MLP, which is a classical type of NN used for classification and regression tasks, with the difference in that MLP is feed-forward while DNN can have loops and feedback from other layers.

One key advantage CNN has over DNN is in pattern recognition, which is done prior to classification. It is typical to decompose a CNN into two sub-networks: the feature extraction subnet, consisting of multiple convolution layers; and the decision making subnet, composed of a couple of fully connected layers. By letting the program convolve a single input with multiple filters, it can recognize various patterns which in turn are fed to different regions of the decision making subnet. The classic example of decision making subnets are DNNs because they are able to specialize regions of the net so that when the appropriate value is fed to a particular region, all the subsequent neurons fire and give the correct answer. In this context, the CNNs feature extraction subnet is an extension to the DNNs decision making net enabling further enhancements.

The idea behind CNNs is to automatize a computer algorithm that can apply *filters* on grid-like topologies and retrieve relevant information without the bias of human interaction. The word "convolutional" indicates that the network employs the mathematical operation of convolution in at least one of its layers. The convolution is defined for any two functions for which the following integral is defined over an interval of the continuous variable t (or sum for a discrete variable):

$$F(t) = (I * K)(t) = \int I(t)K(t-a)dt \tag{2.22}$$

In CNN terminology, the first argument $I(t)$ is often referred as the input; the second, a kernel or filter $K(t)$; and the output $F(t)$, a feature map. In practice, the multidimensional input is usually zero everywhere but the finite set of points for which values are stored. This means that the summation can be done over a finite number of array elements. Convolutions are often performed on more than one axis at a time which makes it a powerful tool easily extensible to higher dimensions. Technically speaking, in-practice implementation of convolution is really a cross-correlation because the kernel is applied directly without flipping it, as it's the (rather confusing) case of convolution. Applying cross-correlation to a two-dimensional discrete input, each element of the feature map is given by

$$F_{\alpha,\beta}(t) = \sum_m \sum_n I_{\alpha+m,\beta+n}(t) K_{mn}(t) \tag{2.23}$$

Cross-correlation is a more natural operation when dealing with matrix products because they can be quickly performed with a graphics processing unit (GPU). Compared to the central processing unit (CPU) which is optimized on serialized computations, the GPU is optimized to work with 3-D arrays which are common in image and video processing as well as 3-D modelling. For this reason most ML libraries implement cross-correlation but call it convolution.

When a convolution is performed on an image, the filter K is swept across the full array and, in doing so, generates a new feature map F that contains a pattern *decoded* from the input I. One can have as many feature maps as filters are defined, and each filter will retrieve a different pattern. This process usually constitutes a single convolutional layer. The following step is to draw another set of filters to convolve with the previously extracted feature maps. If this is repeated several times, then the total number of filters at the last convolutional layer will be very large and, depending on the filter size, the amount of information to handle for a single instance grows very quickly. Here becomes clear why CNNs incur in memory issues.

To reduce the information volume generated per convolutional layer, every layer can be extended by adding a "pooling" kernel P. This part of the layer is applied over the feature map F obtained after the convolution operation. It sweeps each map with an identity kernel P that takes the maximum or average value. By doing this, the next layer (either convolutional or fully connected) will have less input values from a single feature map.

In addition to lessening the amount of data, pooling operations also help to decrease the number of parameters needed in the network. Another way of achieving parameter count reduction is via a "dropout" operation. During training, some number of layer outputs are randomly ignored or "dropped out." This has the effect of making the layer look-like and be treated-like a layer with a different number of nodes and connectivity to the prior layer. Dropout has the effect of making the training process noisy, forcing nodes within a layer to probabilistically take on more or less responsibility for the inputs. At each training stage, individual nodes are either dropped out of the net with probability $1 - p$ or kept with probability p, so that a reduced network is left; incoming and outgoing edges to a dropped-out node are also removed. Although dropout helps prevent over-fitting, this can also be achieved in simple networks with low parameter count and adequate training time.

The feature extraction subnet is conformed by several layers made up with convolutional, pooling and activation operations. At the end of the subnet, the program will have a set of feature maps ready to be passed to a DNN which takes a list of features. Unless the final pooling operation reduced all feature maps to 1×1 arrays (for monochromatic inputs), all arrays must be "flattened" into a single 1-D array or vector. This *flattening* step is the nexus between feature extraction and decision making subnets in every CNN. It's also expected that the feature extraction part is deep enough to decode the most information balanced with large enough filters that allow to *see* better every region of the input.

It must be clear now that parameter counting is crucial when it comes to training NNs. Starting with fully connected layers where it is usually very large because every neuron has the same number of parameters as the others in the same layer. Increasing one neuron in a hidden layer could mean adding hundreds, thousands, or even hundreds of thousands of parameters. For example, if the input instance has 256^2 attributes (a stretched 256^2 pixel image) and the receiving layer has 1,024 neurons, the amount of parameters between those two is over 67 million. Not only it means 67 million products will be computed, but also the same amount parameters must be optimized! Remember: the goal of supervised learning is to find the parameter set $\boldsymbol{\theta}$ that gives the best representation of the objective function $\hat{f}$ with the simplest expressiveness. Conversely, too few parameters produce under-fit, which in turn means ill-defined architectures with slow convergence or non at all. CNNs reduce the number of parameters as well as the size of images via pooling operations whilst still learning patterns. A single filter will have something like 9 or 25 parameters which are used for a convolution. Only these parameters are optimized every training step. Consider a CNN having forty 3×3 kernels spread into five layers, and pooling by a factor of two after each one. This CNN introduce only 400 parameters and can reduce every image up to 32 times!

The main task for the programmer is to play around with the number of feature maps to generate per layer and pooling operations, the kernel's sizes for every convolution and pooling, depth of the decision making subnet, and the types of activation functions to use in order to construct an *adequate* NN. All of the before mentioned are hyper-parameters for a CNN not optimized by the algorithm. This much variability and non-standardized way of programming NNs make ML methods interpretability a challenge and open field of active study.

Chapter 3 – Network Design & Velocity Estimation

In the past century, a new way to understand and predict natural phenomena via simulations came into play. Simulations help science design experiments and scenarios whenever a physical test is impractical or not attainable for current technology. It also means it's possible to control and fine-tune every aspect of the physical situation. When it comes to cosmology, it is the typical way of testing for cosmological parameters obtained from probe measurements such as WMAP and Planck (for latest results see Hinshaw et al., 2013; Planck Collaboration, 2018a, and Table 1.1). One of the most used simulation codes is GADGET (**GA**laxies with **D**ark matter and **G**as int**E**rac**T**), a gravitational force simulator using N-Body and smoothed-particle hydrodynamics (SPH) techniques that was first used for the Millennium Simulation (Springel et al., 2000, 2005). Additionally, the Magneticum Simulation (Dolag et al., 2015) aims to include the baryonic component to galaxy formation and distribution that dark matter alone simulations cannot resolve, by modelling baryonic matter as a perfect fluid and solving the hydrodynamic equations.

This chapter describes the simulation box from where the catalogue kSZ signal maps were extracted. Then, the base NN architecture to train for peculiar velocities is defined, followed by the exploration of some ideas to preprocess these maps for different and simpler NNs. Finally, data analysis and model validation methods are presented and discussed along with the obtained results.

3.1 Magneticum Simulation

The Magneticum Simulations are a set of state-of-the-art, cosmological hydrodynamical simulations of different cosmological volumes with different resolutions performed with an improved version of GADGET 3 (Springel, 2005; Soergel et al., 2018). These simulations use WMAP 7 cosmology parameters (Table 3.1 contains values extracted from Larson et al., 2011) and rely on SPH to simulate the formation of cosmological structure. The simulation covers the physical processes controlling galaxy formation, intra-galactic and intra-cluster medium, as well as the detailed properties of galaxies including morphological classification and internal properties. This also includes the distribution of different metal species within galaxies and galaxy clusters.

Table 3.1: Cosmological and simulation parameters of *Box 0* from Magneticum Simulation (available at http://www.magneticum.org).

Ω_m	0.272	Box size [h^{-1} Mpc]		2688
Ω_Λ	0.728	Number of particles		2×4536^3
Ω_b	0.046	Dark matter particles mass [$10^9\ h^{-1}\ \mathrm{M}_\odot$]	m_{DM}	13
h	0.704	Gas particles mass [$10^9\ h^{-1}\ \mathrm{M}_\odot$]	m_{gas}	2.6
σ_8	0.809	Particle softening [h^{-1} kpc]	f_p	10
n_s	0.963	Star softening [h^{-1} kpc]	f_s	5

The present work makes use of *Box 0* with over 100 thousand million particles (see Table 3.1, and also Bocquet et al., 2016), making it the largest cosmological hydrodynamical simulation performed to date[1]. Such a large box size is fundamental for kSZ signal analysis because it only vanishes at scales $r \gtrsim 300\,\mathrm{Mpc}$. Figure 3.1 shows both SZ maps from the Magneticum simulation (Dolag et al., 2015; Soergel et al., 2018), $\approx 1600\,\mathrm{deg}^2$, determined by the size of the simulation and the highest desired redshift. This is sufficiently large to match the sky coverage of current high resolution CMB and LSS data[2].

To create SZ maps, ideally, one would solve the lightcone equation for every particle, interpolating their positions between snapshots for an observer at one corner of the simulation box. Only gas particles within the lightcone would then contribute to the SZ map. This approach is, however, not feasible for hydrodynamical simulations as the gas properties cannot be interpolated safely between snapshots relatively far apart, as it's the case for large simulations

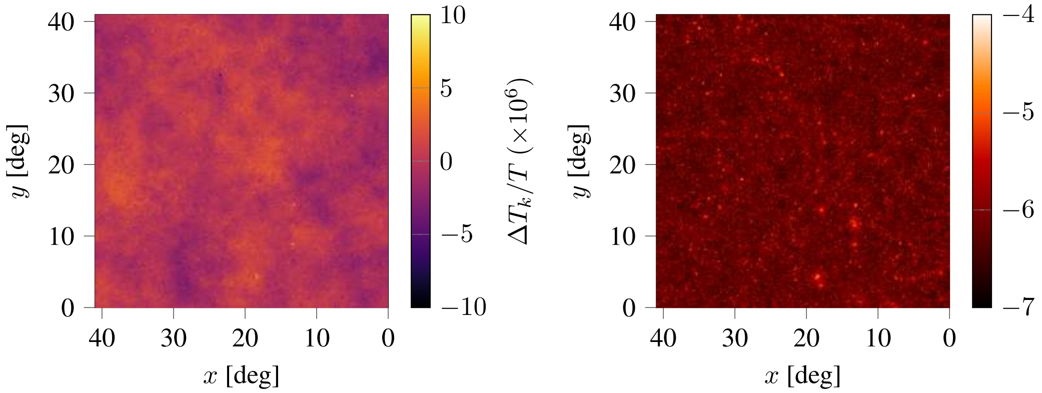


Figure 3.1: SZ maps from the Magneticum simulations. KSZ signal on the left. For tSZ in the right panel the logarithm of the Compton-y parameter is shown to increase the dynamic colour range (available at http://magneticum.org/data.html#SZ).

[1] At the elaboration of this thesis and knowledge of the author.

[2] The effective overlapping sky area between DES Year 1 and SPT is around $1200\,\mathrm{deg}^2$.

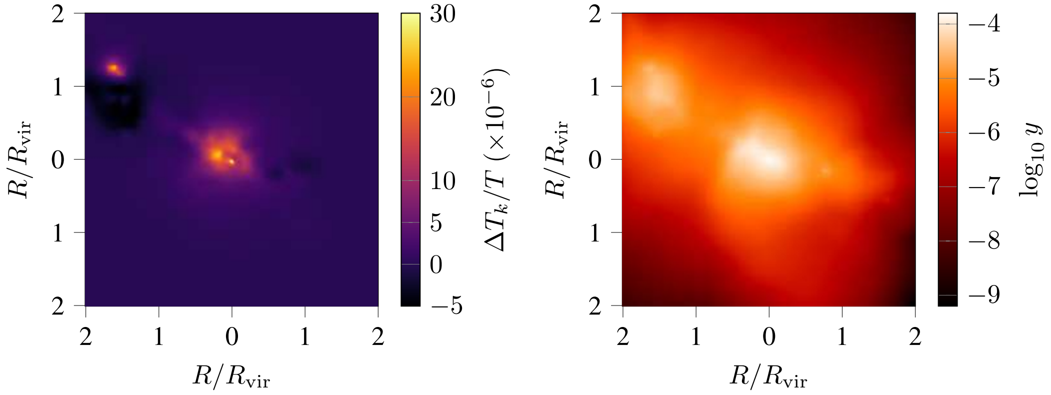


Figure 3.2: Sample kSZ (left) and tSZ (right) images.

like this one. The SMAC code (Dolag et al., 2005)[3] is a map making utility for idealized observations. With this tool, it is possible to extract kSZ, tSZ, and electron density maps for individual clusters (Figs. 3.2 and 3.9).

For the main analysis a total of 10,000 clusters from the redshift slice $z \in [1.04,\ 1.32]$ were selected. The size of a cluster image is set to be two times of its virial radius calculated using the slice's redshift instead of each individual cluster's which means the size of the cluster images is not perfectly normalized to the virial radius. This reduces computation expense while keeping negligible effect on the results. Velocity and mass selections are shown in Fig. 3.3 exhibiting typical cluster values. A broader redshift selection up to $z = 2.15$ can be found in Wang et al. (2020) however testing methods are similar.

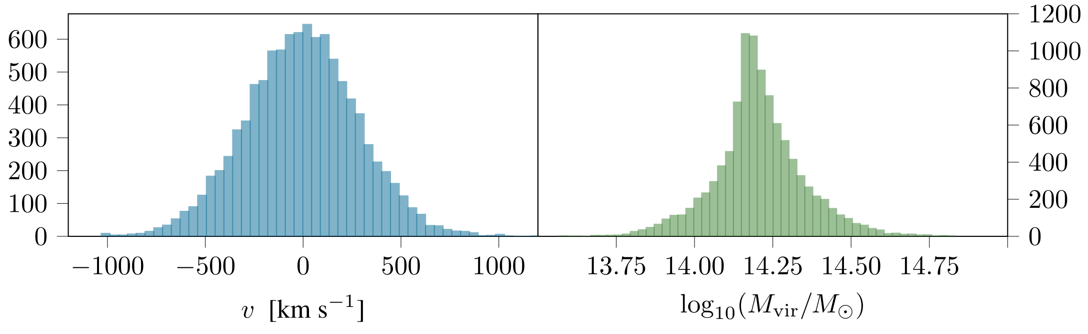


Figure 3.3: Peculiar velocity and mass selection frequency histograms for the selected redshift slice.

[3] https://wwwmpa.mpa-garching.mpg.de/~kdolag/Smac/

3.2 Network Design

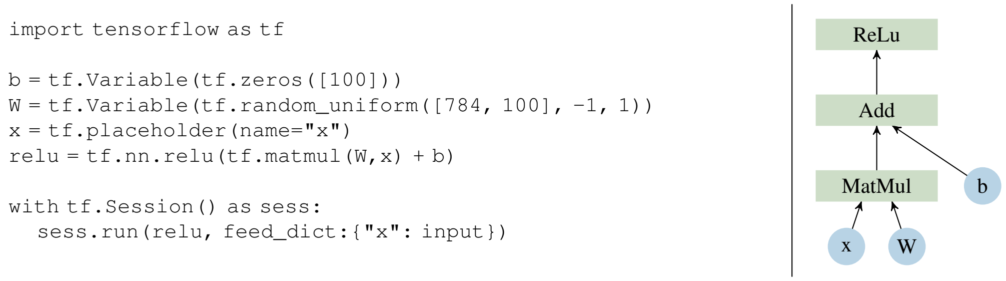
Before diving into very complex network architectures with a sophisticated design of feature extraction convolutions and deep decision making subnets, consider the following argument for how simple architectures can outperform deeper and more complex ones:

> [The] depth of a CNN plays an important role in the discriminability power the network offers. The deeper the better... While this approach has been useful, there are some inevitable issues that arise when the network gets more complex. Computation and memory usage cost and overhead is one of the critical issues that is caused by the excessive effort put on making networks deeper and more complex in order to make them perform better. [It] would be highly desirable to propose efficient architectures with smaller number of layers and parameters that are as good as their deeper versions... a simple architecture, with minimum reliance on new features that outperforms almost all deeper architectures with 2 to 25 times fewer parameters. (Hasanpour et al., 2016)

They tested a simple CNN architecture with *CIFAR-10*, *CIFAR-100* (Krizhevsky, 2009), *MNIST* (LeCun et al., 2010), and *SVHN* (Netzer et al., 2011) datasets, achieving an accuracy of 95.32%, 73.42%, 99.72% and 98.21% respectively. Their architecture has good performance and is simple enough to understand and implement with Machine Learning (ML) libraries.

TensorFlow is an Application Programming Interface (API) developed and maintained by Google™, distributed in the form of software libraries designed as a platform for ML intended to simplify its implementation. A TensorFlow computation is described by a *directed graph* composed of a set of *nodes* and represents the data-flow computation. These graphs are typically constructed by clients (users of the API) in one of the supported frontend languages, such as Python (TensorFlow, 2015). Take for example Fig. 3.4, where a code fragment to construct and then execute a TensorFlow graph using Python is shown, including the resulting computation graph. The NN chosen to train the regression task for kSZ images and peculiar velocities was fully coded using the latest TensorFlow 2.0 for Python 3.8.

The base CNN is composed by six layers that work with monochromatic images. It employs a simplistic design with only four convolutional layers and one hidden layer. The first two convolutions are 16×16 with max pooling; the next two have 5×5 without pool; and one hidden layer connecting the feature extraction convolutional subnet to the decision making neuron (subnet). Convolution stride is set to 1×1 and kernel stride is always set to the



```
import tensorflow as tf

b = tf.Variable(tf.zeros([100]))
W = tf.Variable(tf.random_uniform([784, 100], -1, 1))
x = tf.placeholder(name="x")
relu = tf.nn.relu(tf.matmul(W,x) + b)

with tf.Session() as sess:
  sess.run(relu, feed_dict:{"x": input})
```

Figure 3.4: Example TensorFlow code fragment and its corresponding computation graph.

kernel shape. Per-instance loss is measured by the mean squared error (MSE) and prediction accuracy by the coefficient of determination (Eq. (3.19) below).

The main reason for using this over-simplified network is because kSZ signal is monochromatic so one could expect for the signal strength to give the correct correlation to the peculiar velocity. Also the feature extraction subnet of the CNN will specialize in the overall signal distribution rather than pattern recognition as it is the case with object classification tasks. A single hidden layer with non-linear activation is also expected to provide an appropriate (and yet simple) level of discriminability for the signal extracted distribution. Other caveats that may arise are left to the reader's consideration.

The NN is trained using samples from instances datasets $\mathbb{X}_0$ with $\#(\mathbb{X}_0) = 10,000$. This means that the dataset contains all kSZ images from the redshift slice z. In the following it will be assumed that i counts from 1 to 10,000 unless specified otherwise.

$$\mathbb{X}_0 = \{\mathbf{X}_i \in \mathbb{M}_{256\times256} \mid \mathbf{X}_i \text{ is the kSZ signal of a cluster belonging to } z\} \tag{3.1}$$

where each pixel in the image is a matrix element so that $\mathbf{X}_i = (x_i)_{jk} \in \mathbb{M}_{256\times256}$. Also, $\mathbb{X}_0$ constitutes the catalogue signal denoted by the subscripted zero. The corresponding targets dataset $\hat{\mathbb{Y}}$ and predictions dataset $\hat{\mathbb{Y}}_0$ constructed as follows:

$$\hat{\mathbb{Y}} = \{\hat{v}_i \in \mathbb{R} \mid \hat{v}_i \text{ is the peculiar velocity of a cluster belonging to } z\} \tag{3.2}$$

$$\mathbb{Y}_0 = \{v_i \in \mathbb{R} \mid v_i \text{ is the predicted peculiar velocity of a cluster belonging to } z\} \tag{3.3}$$

Henceforth, all $\hat{v} \in \hat{\mathbb{Y}}$ are also called catalogue velocities, whilst any $v \in \mathbb{Y}_0$ is a predicted velocity. The next section describes how the instances set is processed by CNNs and DNNs in order to produce a prediction for any inputted SZ signal map. Both NNs have only one output neuron whose value v must match the target $\hat{v}$ (up to an arbitrarily small error).

3.3 Training Tests

For the remaining of this work the term “model” is used to refer indistinctly to the set of choices: input dataset, network type (CNN or DNN), and set of outputs from the NN; as well as the parameter set representing the current knowledge. A total of six different models are tested in order to explore various NN options and signal preprocessing. It’s noteworthy that these are arbitrary choices and the reader is encouraged to find better and practical models.

3.3.1 Catalogue Signal

The first test is with the full signal on an as-is basis, i.e. feed the full dataset $\mathbb{X}_0$ to a (*well* defined) CNN and expect it to correctly train. This will function as the reference model with which all other models are compared.

In general, this model is bulky and slow with high-resolution images because the first convolutional layer takes as much matrix products as 256^2 times the number of filters, where each product has 16^2 elements. This has to be done for every training instance, i.e. 10,000 times every epoch (see section 3.4 on how), just for the first convolution . For this reason pooling layers and low parameter count play very important roles in downscaling and simplifying the network. Also GPU acceleration is crucial (and basically essential) to reduce training time.

3.3.2 Systematic Distortions

The simulated images are high resolution idealized observations with little to none systematic errors. To simulate observational conditions the image can be distorted using different filters. For example, a white noise filter added to $\mathbf{X}$ simulates background noise introduced by measurement instruments. Another example is with software processing (common practice nowadays) and softening techniques that produce anti-aliased images to reduce noise. These distortions are basic assumptions of probable error sources and encompass limited examples. With these ideas in mind, the following datasets are proposed:

$$\mathbb{X}_1 = \{\mathbf{X}_i \ + \ \text{white noise} \mid \mathbf{X}_i \in \mathbb{X}_0\} \tag{3.4}$$

$$\mathbb{X}_2 = \{\mathbf{X}_i \ + \ \text{gaussian blur} \mid \mathbf{X}_i \in \mathbb{X}_0\} \tag{3.5}$$

The white noise maps are composed by drawing samples from a uniform distribution proportional to the background of each cluster image at a ratio of 80%. The resulting image has an increase in sharpness and slight loss in homogeneity. The blur was achieved using multidi-

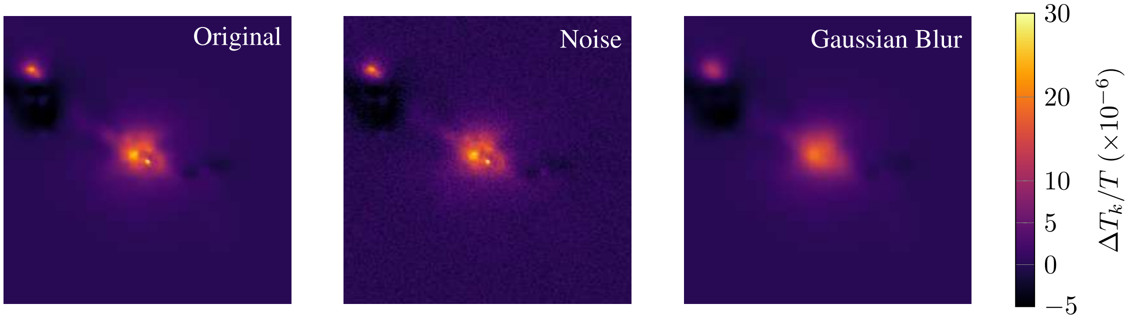


Figure 3.5: Distortions to a kSZ signal example. The original (far left) belongs to $\mathbb{X}_0$; white noise (middle), to $\mathbb{X}_1$; and Gaussian blur (right), to $\mathbb{X}_2$.

mensional Gaussian filter with kernel standard deviation $\sigma_g = 4$ in all dimensions. The last was picked in order to reduce the overall sharpness of the signal whilst maintaining certain resolution, i.e. without incurring in a low-resolution scheme.

The instances datasets $\mathbb{X}_1$ and $\mathbb{X}_2$ are to be trained with the same CNN architecture as $\mathbb{X}_0$. The idea is to compare with the reference and see if the added noise alters the NNs performance. Figure 3.5 shows how the sample instance in Fig. 3.2 is modified by these transformations.

3.3.3 DNN vs CNN

It has been discussed how CNNs can be better than DNNs for image processing. But does it hold true for kSZ signal? In section 2.4 the CNN was presented as the composition of a feature extraction subnet (convolutional) and a decision making subnet (usually a DNN). In other words, an enhancement to DNN. In this context, a comparison between them is not totally unfair but rather natural. However, the data input shape is different for both cases so it will be necessary to think of ways to transform or *vectorize* the kSZ signal in order to feed the deep network.

One idea is to “stretch” each image into a vector. This can be done by stacking each row of the array next to each other in arbitrary, but consistent, order. The resulting vector size is the direct multiplication of the row length by the number of rows. For kSZ maps that would mean a vector with 65,536 entries! One must realise immediately that there are too many input values. To optimize computational expense the image can be downsampled and then stretched. It doesn’t matter if the patterns are now segmented throughout the vector because DNNs specialize different parts of the network via thresholds. Although downsampling looses resolution it enables ease of use for DNNs. The downsample mapping for a squared matrix of size p is defined as:

$$\begin{aligned} \text{down} : \mathbb{M}_{p\times p} &\to \mathbb{M}_{\frac{p}{k}\times\frac{p}{k}} \\ \text{down}\left(\mathbf{X}_i, k\right) &= (f_i)_{\alpha,\beta} \end{aligned} \tag{3.6}$$

where k is rescale factor, $\alpha, \beta = \overline{1, \frac{p}{k}}$, and

$$f_{i,\alpha,\beta} = \sum_{m=1}^{k}\sum_{n=1}^{k} x_{i,\ (\alpha-1)k+m,\ (\beta-1)k+n} \tag{3.7}$$

Definition (3.7) sweeps the input matrix $\mathbf{X}_i$ without overlapping, i.e. taking stride steps of size k. The downsampling can be understood as a pooling layer taking the sum of all elements or convolving with a kernel of ones. After reducing input size, the resulting matrix is vectorized.

$$\begin{aligned} \text{vec} : \mathbb{M}_{p\times p} &\to \mathbb{R}^{p^2} \\ \text{vec}\left(\mathbf{X}_i\right) &= \left(x_{i11}, \ldots, x_{i1p}, x_{i21}, \ldots, x_{ipp}\right)^{\mathrm{T}} \end{aligned} \tag{3.8}$$

For images of size $p = 256$, downsampled by the factor k and then vectorized according to (3.8), defines a new mapping called F_d which in turn gives

$$\mathbb{X}_3 = \left\{ \mathrm{F}_d(\mathbf{X}_i) \;\middle|\; \mathbf{X}_i \in \mathbb{X}_0,\ \begin{array}{c} \mathrm{F}_d : \mathbb{M}_{256\times256} \mapsto \mathbb{R}^{\left(\frac{256}{k}\right)^2},\ k = 2^d,\ d \leqslant 8 \in \mathbb{N} \\ \mathrm{F}_d = \text{vec}(\text{down}\left(\mathbf{X}_i, k\right)) \end{array} \right\} \tag{3.9}$$

This is rather difficult to illustrate with a sample map as it is very large, so consider the following matrix $\mathbf{H}$:

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 2 & 2 & 2 & 1 \\ 1 & 2 & 3 & 3 & 2 & 1 \\ 1 & 2 & 3 & 3 & 2 & 1 \\ 1 & 2 & 2 & 2 & 2 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{Ex. 1}$$

Performing the appropriate change in F_d definition for this example to take a 6×6 matrix, the application of $\mathrm{F}_1(\mathbf{H})$ will output a vector with 9 entries

$$\mathrm{F}_1(\mathbf{H}) = (5,\ 6,\ 5,\ 6,\ 12,\ 6,\ 5,\ 6,\ 5)^{\mathrm{T}} \tag{Ex. 2}$$

Another approach is to assume that each pixel of a cluster image is a random variable realization with a probability distribution of some sort. This means there's a finite non-zero probability that a pixel has the observed value due to some underlying physics. Then, the estimation of the first statistical moments yield some information about the overall distribution on the image. The next dataset is obtained by calculating the first four moments of each image, plus appending the minimum and maximum value to determine signal lower and upper bounds. Only with the first four moments one cannot completely reconstruct the corresponding probability distribution function, but serve as good approximations[4]. The idea is to encode every instance into a smaller vector containing only the most relevant point values that represent the original images.

$$\mathbb{X}_4 = \left\{ \mathrm{M}(\mathbf{X}_i) \;\middle|\; \mathbf{X}_i \in \mathbb{X}_0, \begin{array}{c} \mathrm{M} : \mathbb{M}_{256\times256} \mapsto \mathbb{R}^6 \\ \mathrm{M} = (\min,\ \max,\ E,\ \mathrm{Var},\ \mathrm{Skew},\ \mathrm{Kurt})^{\mathrm{T}} \end{array} \right\} \tag{3.10}$$

The corresponding values for $\mathbf{H}$ given in (Ex. 1) are:

$$\mathrm{M}(\mathbf{H}) = (1.0,\ 3.0,\ 1.55,\ 0.47,\ 0.84,\ -0.50)^{\mathrm{T}} \tag{Ex. 3}$$

A third option is a radial distribution mapping intended to represent the amount of "information" that a single image contains. Let A_w be a squared array of side $2w$, with $w \in \mathbb{N}$, where the only elements different from zero are the $4(2w-1)$ values located along the edges. The mean of non-zero values from A_w represents that particular *slice* (e.g. (Ex. 4) below).

$$\langle A_w \rangle = \frac{1}{4(2w-1)} \sum_{a_{ij} \in A_w} a_{ij} \tag{3.11}$$

The information in each pixel is encoded in both $|a_{ij}|$ and $\mathrm{sign}(a_{ij})$. This way, large values of $|\langle A_w \rangle|$ indicate the presence of strong signal, whilst $\mathrm{sign}(\langle A_w \rangle)$ may be related to signal direction. On the contrary, small values of $|\langle A_w \rangle|$ won't contribute much, indicating the lack of signal and/or small contribution or relevance.

The idea is to map the whole instance $\mathbf{X}$ by taking all possible A_w concentric to $\mathbf{X}$ ranging w. If the instance is of size $p \times p$, then $w = \overline{1, \frac{p}{2}}$. The complete operation of mapping an instance will output a vector that will be used as the full representation of $\mathbf{X}_i$ through the mapping S.

[4] A mathematical justification concerning moment generating functions (MGF) that serve as arguments for the next dataset is found in Appendix A.

This vector will have $\frac{p}{2}$ possible slices, so for kSZ images that means 128 elements.

$$\mathbb{X}_5 = \left\{ \mathrm{S}(\mathbf{X}_i) \mid \mathbf{X}_i \in \mathbb{X}_0,\ \mathrm{S} : \mathbb{M}_{256\times256} \mapsto \mathbb{R}^{128},\ \mathrm{S} = (\langle A_1 \rangle, \ldots, \langle A_{128} \rangle)^{\mathrm{T}} \right\} \tag{3.12}$$

As w increments, so does the size of the slice and further away from the centre of the image is, i.e. it's possible to establish a direct relationship between w and R_{vir}. In this sense, S is a mapping of radial signal. Because each image is $2R_{\mathrm{vir}}$ per side, R_{vir} is at $w = \frac{p}{2}$. In consequence, the distance in $h^{-1}\mathrm{Mpc}$ each slice is from the centre is simply $r = \frac{2w}{p} R_{\mathrm{vir}}$, or equivalently:

$$\frac{r}{R_{\mathrm{vir}}} = \frac{2w}{p} \qquad \text{for} \quad p = 256 \quad \Rightarrow \quad \frac{r}{R_{\mathrm{vir}}} = \frac{w}{128}$$

Plotting $\langle A_w \rangle$ vs w allows to analyse the expected behaviour.

- Over-densities in the image will produce humps for mainly positive values, and valleys for negative ones: signal with different directions will have different signs.
- At the centre of the image (small w) signal is stronger so the mapping will have large values and produce pronounced humps. Away from the centre (large w) signal is fainter or null, so the curve flattens.
- Over-densities farther than R_{vir} are flattened.

Following on the example matrix $\mathbf{H}$ defined in (Ex. 1), of size 6×6, only three concentric slices are possible with $w = 1, 2, 3$.

$$A_1 = \begin{bmatrix} 3 & 3 \\ 3 & 3 \end{bmatrix} \quad A_2 = \begin{bmatrix} 2 & 2 & 2 & 2 \\ 2 & 0 & 0 & 2 \\ 2 & 0 & 0 & 2 \\ 2 & 2 & 2 & 2 \end{bmatrix} \quad A_3 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$\mathrm{S}(\mathbf{H}) = (3,\ 2,\ 1)^{\mathrm{T}} \tag{Ex. 4}$$

In this case the strongest signal is located at the centre and fades linearly. Figure 3.6 shows a complete mapping $\mathrm{S}(\mathbf{X}) \in \mathbb{X}_5$ of the sample kSZ signal in Fig. 3.2 as a function of distance to the centre r. The mapping works as intended: it's mainly positive at the centre and it faints as distance grows. Two negative regions can be identified: one peaking at around 1.38 R_{vir} and the tail. Finally a hump crests over the negative regions which corresponds to the bright spot at the top left corner (see Fig. 3.2). The impact reduction of signal outside the virial

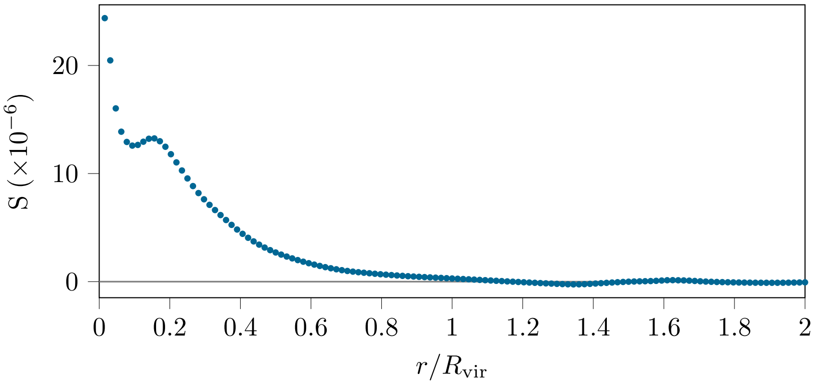


Figure 3.6: Radial mapping over a sample kSZ image in Fig. 3.2.

radius is key because the associated catalogue velocity mainly corresponds to the signal at the centre.

All three $\mathbb{X}_3$, $\mathbb{X}_4$ and $\mathbb{X}_5$ are tested with the same decision making DNN appropriately tuned for input shapes. Due to the nature of the inputs fed to every DNN some tweaks to the activation functions are needed to further separate any linear relations between input features. Nonetheless, the output neuron's activation stayed fixed for both architectures.

3.4 Cross-Validation

In order to train every model the cross-validation method randomly assigns data points $\mathbf{X} \in \mathbb{X}$ to two non-intersecting sets $\mathbb{T}$ and $\mathbb{V}$, called training set and test set, respectively.

$$\mathbb{X} = \mathbb{T} \cup \mathbb{V} \qquad\qquad \mathbb{T} \cap \mathbb{V} \equiv \emptyset \tag{3.13}$$

$$\hat{\mathbb{Y}} = \hat{\mathbb{Y}}_{\mathbb{T}} \cup \hat{\mathbb{Y}}_{\mathbb{V}} \qquad \text{where} \qquad \hat{\mathbb{Y}}_{\mathbb{T}} \cap \hat{\mathbb{Y}}_{\mathbb{V}} \equiv \emptyset \tag{3.14}$$

$$\mathbb{Y} = \mathbb{Y}_{\mathbb{T}} \cup \mathbb{Y}_{\mathbb{V}} \qquad\qquad \mathbb{Y}_{\mathbb{T}} \cap \mathbb{Y}_{\mathbb{V}} \equiv \emptyset \tag{3.15}$$

Every time an instance is fed forward through the network's computational graph (section 2.1 and Fig. 3.4), a comparison between the prediction $v \in \mathbb{Y}_{\mathbb{T}}$ and the target $\hat{v} \in \hat{\mathbb{Y}}_{\mathbb{T}}$ is done internally in order to estimate the loss (supervised learning). At the very last epoch the algorithm performs the final optimization over the parameter set and ends the learning task. This endmost $\boldsymbol{\theta}$ represents the final knowledge of the task and is the model itself, and the output is simply $\mathbb{Y}_{\mathbb{V}}$.

To test the reliability of the trained model a holdout cross-validation method is implemented.

The size of each set is arbitrary with typical train-to-test ratio at about 80/20%. Habitual cross-validation involve multiple runs averaged together (k-fold $\mathbb{T}$ and $\mathbb{V}$), but in holdout a single run is performed. This means less computation time but can result in unstable predictive accuracy since it's not smoothed by the average of multiple iterations. A workaround is to train with the same fixed sets and average over an ensemble of complete trainings in order to better represent the general behaviour. This method is known as Monte Carlo Dropout and is a Bayesian approximation of Neural Networks to Gaussian processes (Wang et al. (2020), see Gal & Ghahramani (2016) for a detailed review). For an ensemble size of N there is one $\mathbb{Y}_{\mathbb{V}}^{j} = \{v_{ij}\}$ for every complete training j. The ensemble's predictions are then the average over all trainings $\mathbb{Y}_{\mathbb{V}} = \langle \mathbb{Y}_{\mathbb{V}}^{j} \rangle$, where each element's mean and dispersion are computed as follows:

$$v_i = \frac{1}{N}\sum_{j=1}^{N} v_{ij} \qquad , \qquad \sigma_i^2 = \frac{1}{N}\sum_{j=1}^{N}(v_{ij} - v_i)^2 \tag{3.16}$$

Each training step is delimited by an update of the parameter set. There can be as much training steps as there are samples in $\mathbb{T}$ (SGD), but remember that a full sweep through the complete training set constitutes a single epoch. Balancing the number of training steps (or batches for that matter) and epochs is key to a reliable training.

An ideal model, consistent with the objective function, will precisely predict all values. Then, a scatter plot of v vs $\hat{v}$ should graph a perfectly straight forty-five degree line, because every prediction hits the target value (truth). In practice, all pairs $(\hat{v}, v)$, are expected to fall around the identity line in this plot due to numerical and unknown error sources. These residuals are the distance from a prediction to its target.

$$\mathbb{E} = \{e_i = v_i - \hat{v}_i \mid \forall\ (v_i, \hat{v}_i) \in (\mathbb{Y}_{\mathbb{V}}, \hat{\mathbb{Y}}_{\mathbb{V}})\} \tag{3.17}$$

The expectation value $\mathbf{E}[e] = \hat{\mu}_e$ is the mean prediction error (MPE) where small values are better, indicate less dispersion and, ideally, tend to zero.

$$\hat{\mu}_e = \frac{1}{\#(\mathbb{E})}\sum_{e \in \mathbb{E}} e \tag{3.18}$$

The algorithms accuracy is measured by the coefficient of determination R^2 (R-squared) that measures how well observations fit the model dividing the squared covariance between the two variables by their variances. A value of $R^2 = 0$ means that the dependent variable (in this case v) cannot be explained by the independent variable (in this case $\hat{v}$), i.e. totally uncorrelated. Conversely, if $R^2 = 1$, the dependent variable is totally explained by the independent one. So the accuracy measure over $\mathbb{V}$ is the value of R-squared at the final validation step.

$$R^2 = \frac{\mathbf{E}\left[(v - \bar{v})\left(\hat{v} - \bar{\hat{v}}\right)\right]^2}{\sigma_v^2\, \sigma_{\hat{v}}^2} \tag{3.19}$$

Formally speaking the independent variable is $\mathbf{X}$ via (2.6), nonetheless it's equivalent to consider $\hat{v}$ as the independent variable given how supervised learning trains. In other words, how much of v is explained "by the fact that $\mathbf{X}$ is the input" and "by the fact that $\hat{v}$ is the target", mean the same thing in this case.

3.5 Deep Learning Predictions

Following the validation method in section 3.4, model training predictions are presented in a $v - \hat{v}$ plot in order to show the capacity of every DL model to reproduce the corresponding target value to every training instance. The expected behaviour of this plot is of course a 45° angle straight line (1:1 scale) for which the slope is computed via uncertainty weighted linear fit. The test set is obtained through the holdout method for cross-validation. Lastly, an ensemble size of 10 is averaged for each model to provide better statistics on its behaviour.

Training with catalogue images only using the holdout method gives a test sample of 2,000 clusters. Figure 3.7 shows the prediction results $\mathbb{Y}_{\mathbb{V}}$ for all datasets and transformations explored in the section 3.3. Error bars obtained from Eq. (3.16) turned to be very small at the scale presented in the figure so they are omitted for ease of visualization. However, it's important to remark that the uncertainty weighted linear fit is very sensitive to small errors but not so with large deviations because every weight is proportional to σ_i^{-1}. An ensemble size of 10 was sufficient to reduce the uncertainties quite a lot.

All models were trained for the same amount of epochs (full dataset passes) to evaluate convergence at the same *time*. CNNs had to train in small batch sizes (SGD) due to memory limitations but DNNs could take the full dataset every epoch (BGD). Moreover, the cardinality of $\boldsymbol{\theta}$ for DNNs is around 10% of that of CNNs. Using GPU acceleration also on DNNs caused a training time reduction down to ∼14% of CNNs time for $\mathbb{X}_3$ and $\mathbb{X}_5$, and to ∼3% for $\mathbb{X}_4$. This huge difference is merely due to convolution layers which are generally slow and increase the number of parameters to optimize, and not because of batch training schemes.

Both slope and R-squared are within good agreement with the ideal line for the first three

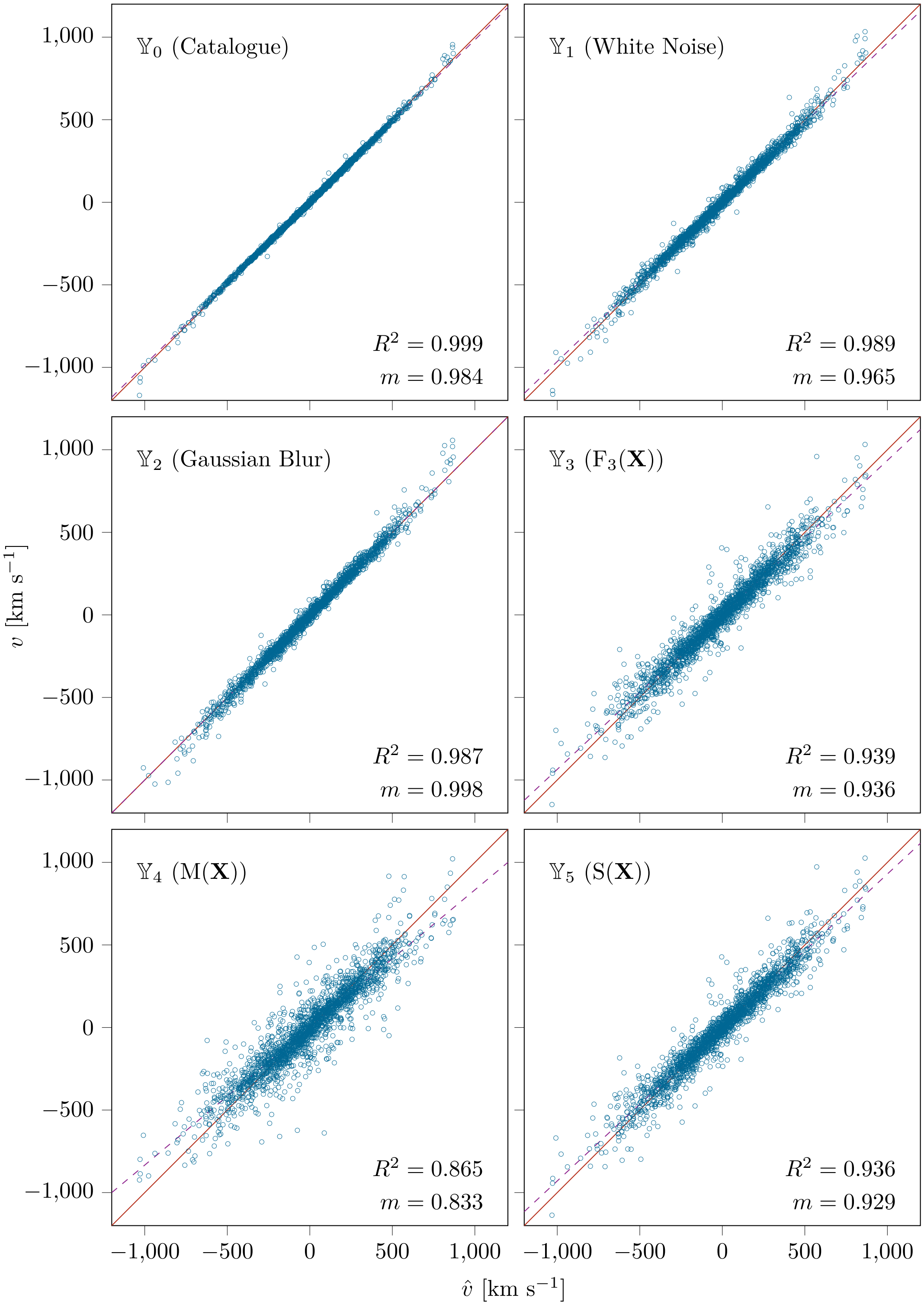


Figure 3.7: Training on all six datasets. Catalogue peculiar velocity is along the horizontal axis whilst the model predicted velocity, along the vertical. The solid line is the ideal relation and the dashed line is the uncertainty weighted linear fit with slope m. No error bars are shown to ease visualization. See section 3.3 for the explicit definition of all datasets.

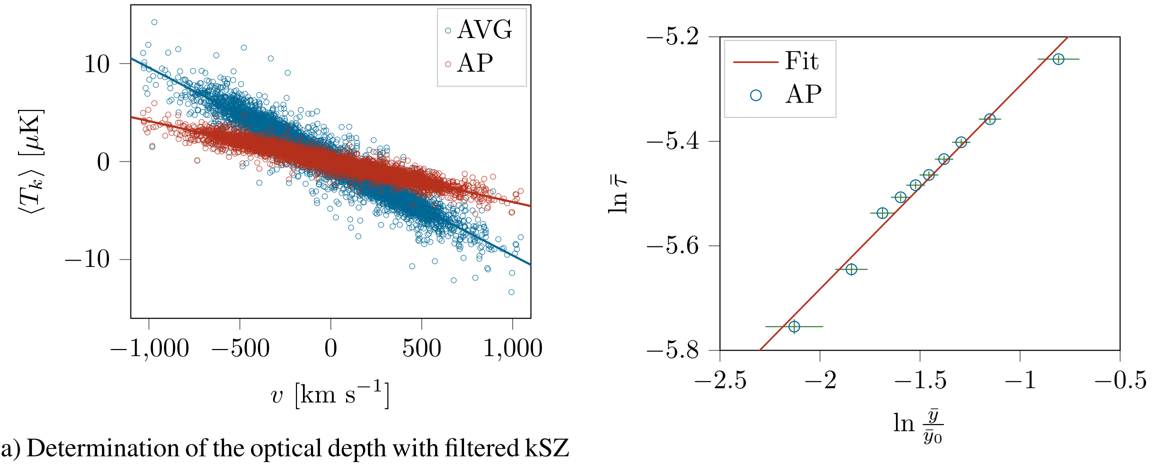
models that were trained as CNNs. At a glance all three exhibit the same precision however, the overall dispersion in the catalogue clean signal $\mathbb{X}_0$ is smaller than the other two cases. Nonetheless, the model proves itself robust enough to *oversee* these noises.

DNN performance on the other hand was surprisingly good. Remember that DNNs are conceptually inferior to CNNs. The datasets $\mathbb{X}_3$ and $\mathbb{X}_5$ encode the signal radial distribution and give acceptable results. This indicates that radial distribution in the perpendicular plane must be useful to the network in order to better find a relation with the peculiar velocity. $\mathbb{X}_4$ is the worst of all, but it doesn't necessarily mean it's conceptually bad. What can be deduced from these three models is that the proposed DNN architecture performs better with radial profiles than few statistical measures of the signal. This can be further improved by optimizing each network to the input particularities.

3.6 Optical Depth Estimation

The method for estimating peculiar velocities directly from Eq. (1.28) requires to know the optical depth τ for each individual cluster. Soergel et al. (2018) estimate this via a linear relation between kSZ and line-of-sight cluster velocity. To provide a point-estimate of average the kSZ signal $\langle T_k \rangle$ they use two types of filters f: average (AVG) and adaptive (AP), in order to estimate $\langle T_k \rangle$ with aperture equal to the virial radius of each cluster. Then fit for $\langle T_k \rangle - v$ where the slope is $\bar{\tau}_f$. This yields the *effective* optical depth of the cluster sample in the respective redshift slice and is readily done in Fig. 3.8a. The obtained optical depths differ considerably with $\bar{\tau}_{\text{avg}} = 0.009\,58(3)$ and for the adaptive filter less than half $\bar{\tau}_{\text{ap}} = 0.004\,13(1)$. However, as Soergel says, "it is not a priori clear that this is also the correct quantity to use for the pairwise kSZ measurement... as internal gas motions or correlations between velocity and optical depth could cause a bias."

Because the optical depth of CMB photons through clusters is not directly observable for an individual cluster, Battaglia (2016) explore inference of an average optical depth within a fixed aperture for a given cluster using a power-law scaling relation. Again, Soergel et al. (2018) use this relation for the observable Compton$-y$ parameter obtained by applying the AVG and AP filters to tSZ maps. For every redshift slice in their selection of maps they take ten mass bins and compute the average $\bar{y}$ over all clusters in each bin. Then fit for the optical depth according to the following scaling relation:



(a) Determination of the optical depth with filtered kSZ T_k and line-of-sight v for the redshift slice. The optical depth is determined from the slope of the relation using Eq. (1.28). AVG filter leads to significantly larger scatter and a slightly steeper slope than AP.

(b) Scaling relation fit. The circular markers and error bars denote the individual mass bin estimates for $\bar{y}$ and $\bar{\tau}$; the solid line is the scaling relation fit to them.

Figure 3.8: Scaling relation method for optical depth estimation.

$$\ln \bar{\tau} = \ln \bar{\tau}_0 + \alpha \ln \frac{\bar{y}}{\bar{y}_0} \tag{3.20}$$

where the normalization constant is set to $\bar{y}_0 = 10^{-6}$, which is a typical value for the aperture-averaged Compton$-y$ parameter for the clusters in the sample; so that $\bar{\tau}_0$ is also representative for the typical aperture-averaged optical depth. Orthogonal regression for α and $\ln \bar{\tau}_0$ is performed using the binned estimates for $\bar{\tau}$ (with AP filter) and corresponding $\bar{y}$ shown in Fig. 3.8b. The fitted values for Eq. (3.20) obtained for the selected redshift are $\alpha = 0.39(2)$ and $\ln \bar{\tau}_0 = -4.90(4)$. These results differ from those reported by Soergel et al. (2018), however this is likely due to the cluster mass selection for the current redshift slice (see Fig. 3.3). Using the scaling relation to calculate individual cluster optical depths $\bar{\tau}_i$, a new set of predictions denoted by $\mathbb{Y}_6$ is defined:

$$\mathbb{Y}_6 = \left\{ v_{\tau,i} \in \mathbb{R} \;\middle|\; v_{\tau,i} = -\frac{c}{\bar{\tau}_i} \left\langle \frac{\Delta T_k}{T} \right\rangle_i \right\} \tag{3.21}$$

According to Eq. (1.18) the optical depth τ is integrated over a path length ℓ at a specific location in space. For a single cluster, τ can calculated by averaging the electron density within the virial radius.

$$\tau = \frac{1}{\pi R_{\text{vir}}^2} \int_0^{R_{\text{vir}}} \int_{-\ell}^{\ell} \sigma_{\text{T}} n_e \, d\ell dr \tag{3.22}$$

This is only possible for simulation data because the electron density maps can be generated and extracted directly, however it is not observable. Figure 3.9 below is the corresponding electron density n_e to the sample map in Fig. 3.2 as retrieved from the simulation via SMAC and normalized to the viral radius.

Similarly to Soergel et al. (2018), kSZ signal is also averaged over the same space using the AP filter so that the peculiar velocity is simply the quotient between average signal and average optical depth. The path length integration limits are set at $|\ell| = 100 \ h^{-1}\text{Mpc}$ giving a total integration distance of $200 \ h^{-1}\text{Mpc}$. The corresponding predictions dataset is

$$\mathbb{Y}_7 = \left\{ v_{n_e,i} \in \mathbb{R} \;\middle|\; v_{n_e,i} = -\frac{c}{\tau_i} \left\langle \frac{\Delta T_k}{T} \right\rangle_i \right\} \tag{3.23}$$

Results for $\mathbb{Y}_6$ and $\mathbb{Y}_7$ are shown in Fig. 3.10. Compared to the catalogue training, it's clear that neither provide the same accuracy and low dispersion as $\mathbb{Y}_0$. Both DL and direct computation methods present strong correlation with the true peculiar velocities, however the direct methods exhibit even larger biases in magnitude. Surprisingly enough the scaling relation velocity predictions $\mathbb{Y}_6$ have a very tight slope value despite being estimated after several approximations which make it prone to large errors. However, the fitting process contains ensemble statistics for the complete redshift slice tSZ spectrum so it might help

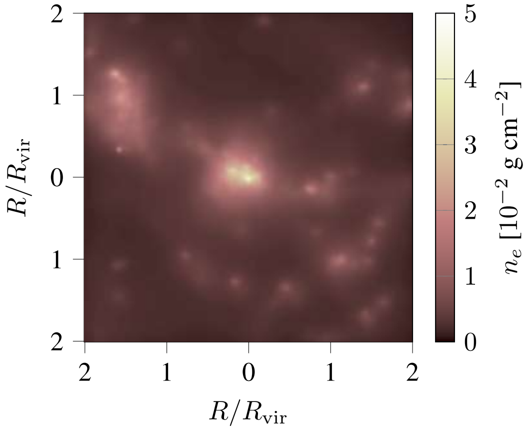


Figure 3.9: Sample electron density image.

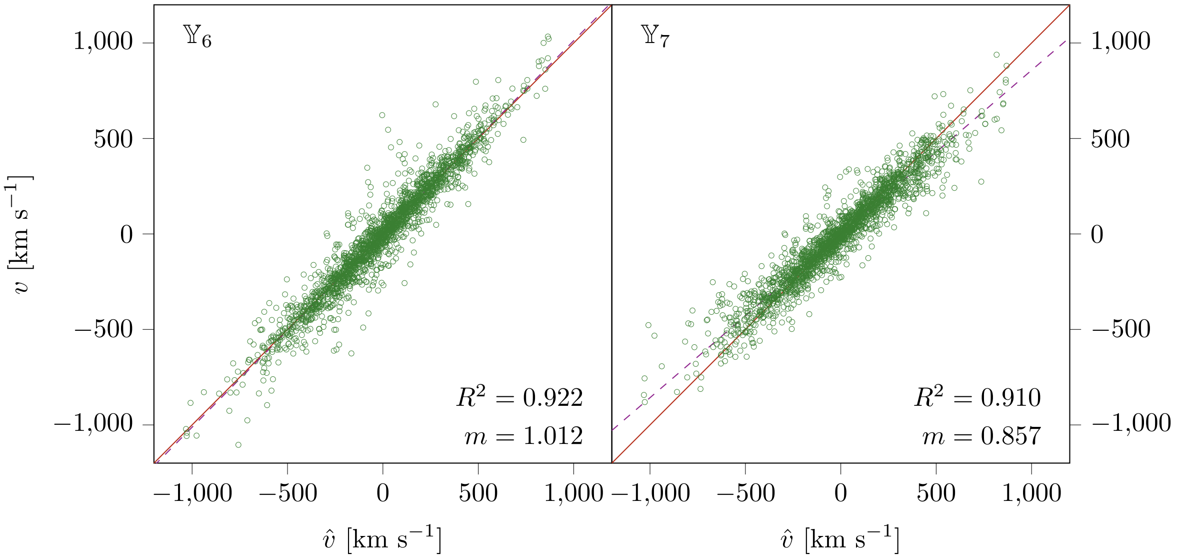


Figure 3.10: Peculiar velocities retrieved by optical depth estimation methods. $\mathbb{Y}_6$ optical depth was computed using the scaling relation after fitting Eq. (3.20); and for $\mathbb{Y}_7$, the direct integration in Eq. (3.22). Catalogue peculiar velocity is along the horizontal axis whilst the model predicted velocity, along the vertical. The solid line is the ideal relation and the dashed line is a simple linear fit with slope m.

with individual cluster estimations. Counterintuitively electron density integration did not provide very good predictions.

3.7 Model Analysis

Table 3.2 collects all results for every model from Figs. 3.7 and 3.10. The mean prediction error $\hat{\mu}_e$ was calculated using Eq. (3.18) and σ_e is simply the standard deviation of residuals (see Eq. (3.17)). Also, μ_e is the mean predictions error computed with bootstrap method and providing the 95% confidence interval. Considering that typical cluster velocities are up to $1000\,\mathrm{km\,s^{-1}}$ and almost normally distributed around zero (see Fig. 3.3), the mean prediction error does not provide a significant bias information. However, dispersions are appreciable high for DNN models and direct methods method. This indicates that the fit is not as good as it would seem from just considering the first two columns. Finally, a quick check on the ratio of wrongly predicted signs give an idea of how well every model can determine the direction of every cluster.

Deviations from the target value are expected due to statistical and systematic errors produced anywhere from the simulation process and extraction of the SZ signal to numerical errors in the network and design decisions. Although R-squared and MPE are useful point values to quantify the *goodness* of a model, they do not provide any information regarding error distribution and systematic biases.

To test for systematics the mean-centred residuals distribution $P(e - \hat{\mu}_e)$ is to be analysed. Particularities are not really important for the discussion but how distributions compare between them is. The overlap index η is a similarity test for two distributions by measuring the area of juxtaposition between them. Let $f_A(x),\ f_B(x) \in \mathbb{R}^n$ be normalized distribution functions defined over real intervals for x denoted by $[a_1,\ a_2]$ and $[b_1,\ b_2]$ respectively:

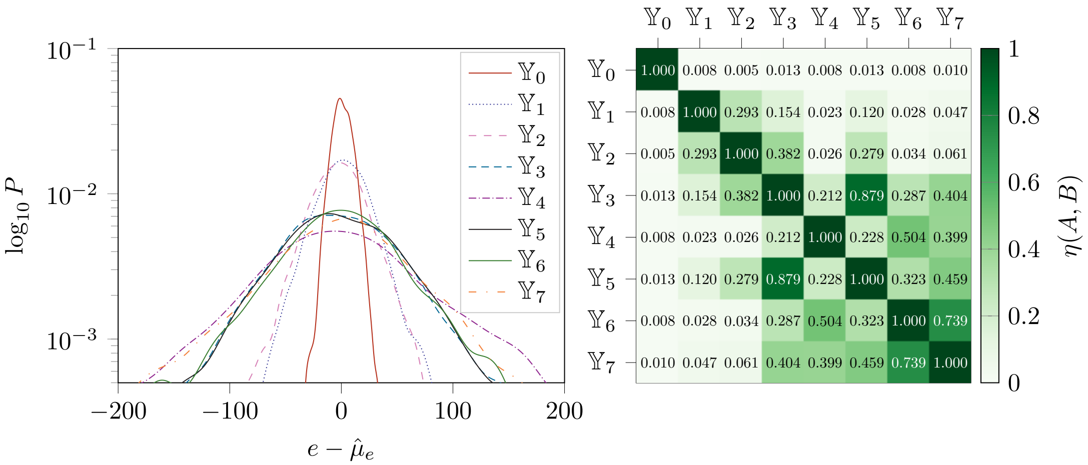
$$\begin{aligned} \eta(A,B) : \mathbb{R}^n \times \mathbb{R}^n \to [0,\ 1] \\ \eta(A,B) = \int\limits_{\mathbb{R}^n} \min\left[f_A(x), f_B(x)\right]\ dx \end{aligned} \tag{3.24}$$

When $\eta = 1$: $a_1 = b_1$, $a_2 = b_2$, and $f_A \equiv f_B$. For $0 < \eta < 1$ both distributions overlap in some interval $[c_1,\ c_2]$, where $c_1 = \max(a_1, b_1)$, $c_2 = \min(a_2, b_2)$, and nothing in particular can be said about the shapes of f_A and f_B. Lastly, when $\eta = 0$, the distributions do not overlap at any point x, and so either $a_2 < b_1$ or $b_2 < a_1$. To ensure all distributions overlap one must take the mean-centred distributions.

The mean centred error distributions were smoothed using kernel density estimation (KDE). With Fig. 3.11 it becomes evident that in fact $\mathbb{Y}_1$ and $\mathbb{Y}_2$ are almost identical even in their error distributions. Another case like this is with $\mathbb{Y}_3$ and $\mathbb{Y}_5$ with a slight bias to the left caus-

Table 3.2: Model training results. The mean prediction error estimation μ_e is computed through bootstrap method to provide 95% confidence interval, with bounds indicated by super- and subscripts. The last column contains the percentage of correct velocity sign predictions.

Model	m	R^2	$\hat{\mu}_e$ [km s^{-1}]	σ_e [km s^{-1}]	μ_e [km s^{-1}]	sign [%]
$\mathbb{Y}_0$	0.984	0.999	0.976	11.019	$0.961\,^{1.902}_{0.011}$	0.80
$\mathbb{Y}_1$	0.965	0.989	2.494	29.937	$2.417\,^{5.405}_{-0.376}$	2.00
$\mathbb{Y}_2$	0.998	0.987	−2.249	33.674	$-2.224\,^{0.694}_{-5.193}$	2.30
$\mathbb{Y}_3$	0.936	0.939	−1.355	71.668	$-1.312\,^{4.879}_{-7.361}$	5.25
$\mathbb{Y}_4$	0.833	0.865	0.002	107.071	$-0.012\,^{9.080}_{-9.704}$	9.10
$\mathbb{Y}_5$	0.929	0.936	0.195	73.791	$0.191\,^{6.889}_{-5.961}$	5.55
$\mathbb{Y}_6$	1.012	0.922	−0.212	81.569	$-0.413\,^{6.832}_{-6.975}$	6.95
$\mathbb{Y}_7$	0.857	0.910	2.123	87.257	$2.103\,^{9.767}_{-5.961}$	6.95



	$\mathbb{Y}_0$	$\mathbb{Y}_1$	$\mathbb{Y}_2$	$\mathbb{Y}_3$	$\mathbb{Y}_4$	$\mathbb{Y}_5$	$\mathbb{Y}_6$	$\mathbb{Y}_7$
$\mathbb{Y}_0$	1.000	0.008	0.005	0.013	0.008	0.013	0.008	0.010
$\mathbb{Y}_1$	0.008	1.000	0.293	0.154	0.023	0.120	0.028	0.047
$\mathbb{Y}_2$	0.005	0.293	1.000	0.382	0.026	0.279	0.034	0.061
$\mathbb{Y}_3$	0.013	0.154	0.382	1.000	0.212	0.879	0.287	0.404
$\mathbb{Y}_4$	0.008	0.023	0.026	0.212	1.000	0.228	0.504	0.399
$\mathbb{Y}_5$	0.013	0.120	0.279	0.879	0.228	1.000	0.323	0.459
$\mathbb{Y}_6$	0.008	0.028	0.034	0.287	0.504	0.323	1.000	0.739
$\mathbb{Y}_7$	0.010	0.047	0.061	0.404	0.399	0.459	0.739	1.000

Figure 3.11: Normalized mean centred error distributions per model (left). Error distribution for $\mathbb{Y}_0$ is very sharp and all others very flat so the $\log_{10} P$ facilitates the visualization. The overlap index (right) is displayed with a colour map.

ing under-shoot. The overlap η, on the other hand, doesn't reflect this fact quite notoriously, but this is rather a numerical problem due to the histograms resolution even after KDE.

Lastly, the absolute relative error expectation value was computed by setting lower bounds for the catalogue velocities in order to remove the influence of small velocities in the average error. For catalogue velocities, the effect is not perceptible as the errors are already very low, around 12% and down to 5% after removing velocities lower than $20\,\mathrm{km\,s^{-1}}$. In general, this lower bound ensures a reduction to the half or more of the mean absolute error. This estimator is not stable at higher bounds due to the decreasingly number of data points (see Fig. 3.3).

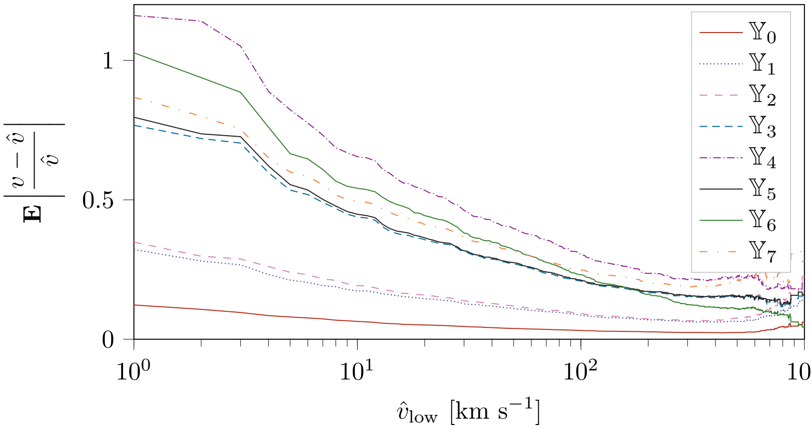


Figure 3.12: Mean absolute relative error for each model. The horizontal axis gives the lower velocity bound in estimating the relative error expectation value.

3.8 Pairwise Velocity Estimator

As it can be seen in sections 3.5, 3.6 and 3.7, individual cluster peculiar velocities have very large uncertainties. For this reason ensemble statistics are more convenient than individual cluster velocity analysis, and also are related directly to cosmological parameters. And so the final test to all models is the pairwise estimator $v_{12}(r)$ defined in Eq. (1.2) (see section 1.3).

Figure 3.13 shows the peculiar velocity statistic calculated for catalogue velocities, DL training prediction, and the conventional methods. A complete overlap between the DL prediction and the target indicates that this technique provides reliable peculiar velocities for cosmological statistical studies. Direct optical depth estimation methods are not far behind but clearly do not provide the same accuracy to the true pairwise velocity. The pairwise statistics for models $\mathbb{Y}_1$ and $\mathbb{Y}_2$ follow $\mathbb{Y}_0$ very closely, so they aren't shown in the figure. Same goes for $\mathbb{Y}_3$ and $\mathbb{Y}_5$ for being very similar to $\mathbb{Y}_7$. This might be happening due to similar dispersion and sign correlation values.

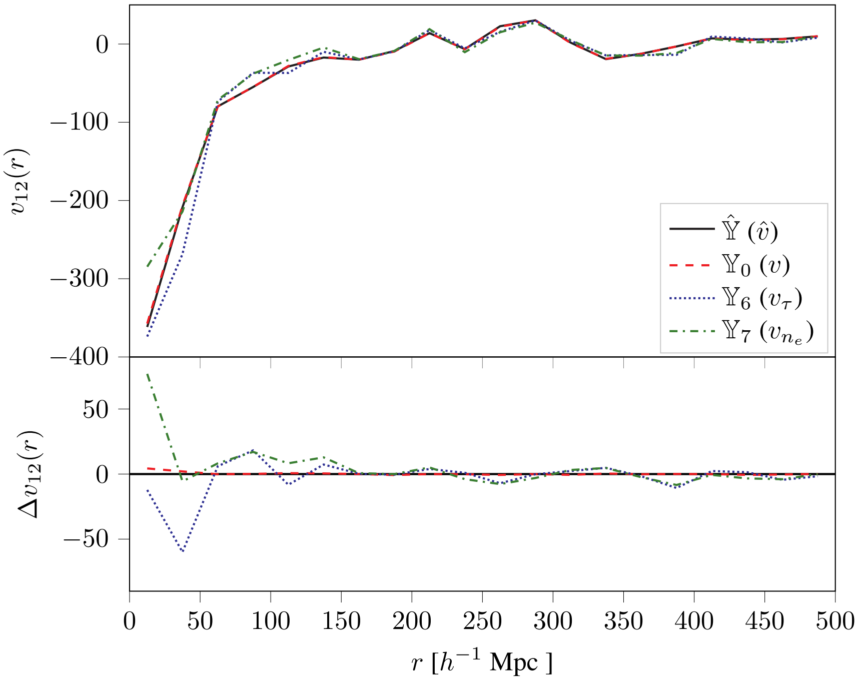


Figure 3.13: Pairwise velocity estimator for peculiar velocities retrieved by three different methods compared to the *true* estimator computed from the catalogue velocities $\hat{v} \in \hat{\mathbb{Y}}$. DL peculiar velocities were selected for the lowest dispersion CNN model trained with the full kSZ catalogue map; v_τ were retrieved from the fitted scaling relation for the optical depth; and v_{n_e}, from the integrated electron density. The difference Δv_{12} is measured with respect to the pairwise estimator computed from catalogue peculiar velocities $\hat{v}$.

Conclusions

The Sunyaev-Zel'dovich effect is instrumental to the study of large-scale structure velocity fields. Conventional methods that extract peculiar velocities form KSZ require several steps, such as signal filtering and mostly optical depth estimation. However, this process is prone to large errors (introduced by filters and approximated models), which in turn make peculiar velocity prediction inaccurate. On the other hand, Deep Learning (DL) neural network algorithms (NN) are state-of-the-art techniques in regression and classification tasks. Therefore, this work tested the feasibility of using NNs to estimate peculiar velocities from the kinematic SZ signal.

The simplistic convolutional NN (CNN) architecture design choice provided excellent results. Training with the catalogue signal directly from the Magneticum Simulation (*Box 0*) yielded the lowest error mean dispersion of $\sigma_e = 11.019\,\mathrm{km\,s^{-1}}$ compared to all other tests. In contrast, the two conventional methods implemented in section 3.6 gave higher values at $81.569\,\mathrm{km\,s^{-1}}$ using the thermal SZ effect (Soergel et al., 2018), and $87.527\,\mathrm{km\,s^{-1}}$ using the direct integration of the electron density (see Eq. (3.22)). In consequence the hypothesis proposed is accepted because the DL algorithm outperformed all other choices and improved the conventional computation of peculiar velocities. One major disadvantage for NN use is model interpretability. Nonetheless, Machine Learning is an active field of research and a lot of methods and techniques are under active development and exploration.

CNNs resulted to be the fittest for image processing although bulky, with $\sim 10^6$ parameters; slow, around several hours to train; and with high memory demand, i.e. low count of samples per batch due to the feature extraction part of the net. In a word: expensive. DNNs are generally lightweight, with $< 10^5$ parameters; fast, taking 3-14% training time compared to CNN's; and inexpensive, even allowing full dataset per batch. DNNs performance is not far behind CNNs despite the image transformations explored in section 3.3 being rudimentary. Further optimizations to DNNs design could in fact match the results for the catalogue training or even outperform it. Considering CNNs have better convergence these would be better implementations to train models intended for library distribution. On the other hand, DNNs dispersion is very close to conventional methods but quicker to implement and train than CNN, which makes them better to get quick results and exploration purposes.

The neural network provided reliable predictions for studying LSS velocity fields using the pairwise velocity estimator v_{12}, in addition to improved correlation compared to conventional methods. Both struggle at lower scales however DL training turned out scale-independent (as Fig. 3.13 shows). This makes it a very desirable tool for cosmological analysis.

A direct application of this methodology for signal regression can be for cluster catalogue simulators who would use the largest cosmological simulations and train models to fit particular parameters for individual clusters. These models can then be distributed as compiled libraries so that small collaborations can test their own small simulation boxes for those parameters. Consistency checks between simulations could also be possible. The ultimate goal would be to train this type of models with simulations and use them with observational data in order to provide estimates for the desired cosmological parameters.

Bibliography

Battaglia N., 2016, Journal of Cosmology and Astroparticle Physics, 2016

Baumann D., 2009, TASI Lectures on Inflation, arXiv e-print (arXiv:0907.5424)

Birkinshaw M., 1999, Phys. Rep., 310, 97

Bocquet S., Saro A., Dolag K., Mohr J. J., 2016, MNRAS

Calafut V., Bean R., Yu B., 2017, Phys. Rev. D, 96, 123529

Chandrasekhar S., 1950, Radiative Transfer, 1 edn. Oxford University Press, Oxford

Diaferio A., et al., 2005, MNRAS, 356, 1477

Dolag K., Hansen F. K., Roncarelli M., Moscardini L., 2005, MNRAS, 363, 29

Dolag K., Remus R.-S., Teklu A. F., 2015, Proceedings of the International Astronomical Union, 11, 292

Eisenstein D. J., Hu W., 1998, ApJ, 496, 605

Feller W., 2008, An Introduction to Probability Theory and its Applications, 2nd edn. No. v. 2 in Wiley publication in mathematical statistics, Wiley India Pvt. Limited

Ferreira P. G., Juszkiewicz R., Feldman H. A., Davis M., Jaffe A. H., 1999, ApJ, 515, L1

Fixsen D. J., 2009, ApJ, 707, 916

Flender S., Bleem L., Finkel H., Habib S., Heitmann K., Holder G., 2016, ApJ, 823, 98

Flender S., Nagai D., McDonald M., 2017, ApJ, 837, 124

Friedman A., 1922, Zeitschrift für Physik, 10, 377

Gal Y., Ghahramani Z., 2016, arXiv preprint (arXiv:1506.02157)

Gamow G., 1946, Phys. Rev., 70, 572

Hand N., et al., 2012, Physical Review Letters, 109, 1

Hasanpour S. H., Rouhani M., Fayyaz M., Sabokrou M., 2016 (arXiv:1608.06037)

Hill J. C., Ferraro S., Battaglia N., Liu J., Spergel D. N., 2016, Phys. Rev. Lett., 117, 051301

Hinshaw G., et al., 2013, ApJS, 208, 19

Hu W., White M., 2004, Scientific American, 209

Juszkiewicz R., Springel V., Durrer R., 1999, The Astrophysical Journal, 518, L25

Kompaneets A. S., 1956, Zhurnal Eksperimentalnoi i Teoreticheskoi Fiziki, 31, 876

Krizhevsky A., 2009, Master's thesis, University of Toronto

LaRoque S. J., Carlstrom J. E., Reese E. D., Holder G. P., Holzapfel W. L., Joy M., Grego L., 2002 (arXiv:astro-ph/0204134)

Larson D., et al., 2011, ApJS, 192

Lavaux G., Afshordi N., Hudson M. J., 2013, MNRAS, 430, 1617

LeCun Y., Cortes C., Burges C., 2010, ATT Labs, 2

Lindner R. R., et al., 2015, ApJ, 803, 79

Makiya R., Hikage C., Komatsu E., 2019, arXiv e-prints (arXiv:1907.07870)

McCulloch W. S., Pitts W., 1943, Bulletin of Mathematical Biophysics, 5, 115

Mitchell T. M. T. M., 1997, Machine Learning, 1 edn. McGraw-Hill Science/Engineering/Math, New York

Mittal A., de Bernardis F., Niemack M. D., 2018, Journal of Cosmology and Astropar. Phys., 2018, 032

NASA 2016, Cosmic Background Explorer

Netzer Y., Wang T., Coates A., Bissacco A., Wu B., 2011, NIPS Workshop on Deep Learning and Unsupervised Feature Learning

Peebles P., 1993, Principles of Physical Cosmology, 1st edn. Press, Princeton University, Princeton, N.J.

Penzias A. A., Wilson R. W., 1965, ApJ, 142, 419

Peterson J., Fabian A., 2006, Physics Reports, 427, 1

Planck Collaboration 2015, A&A, 594, A1

Planck Collaboration 2016, A&A, 586, 1

Planck Collaboration 2018a, A&A

Planck Collaboration 2018b, A&A, 617, 1

Rephaeli Y., 1995, ARA&A, 33, 541

Rybicki G. B., Lightman A. P., 1985, Radiative Processes in Astrophysics. Wiley-VCH Verlag GmbH & Co. KGaA, Weinheim, Germany

Samuel A. L., 1959, IBM Journal of Research and Development, 44, 207

Schaan E., et al., 2016, Phys. Rev. D, 93, 082002

Shimon M., Rephaeli Y., 2004, New Astronomy, 9, 69

Soergel B., et al., 2016, MNRAS, 461, 3172

Soergel B., Saro A., Giannantonio T., Efstathiou G., Dolag K., 2018, MNRAS, 478, 5320

Springel V., 2005, MNRAS

Springel V., Yoshida N., White S. D. M., 2000, New Astronomy, 6, 79

Springel V., et al., 2005, Nature

Sugiyama N. S., Okumura T., Spergel D. N., 2018, MNRAS, 475, 3764

Sunyaev R. A., Zeldovich Y. B., 1980, MNRAS, 190, 413

Switzer E. R., 2016, https://lambda.gsfc.nasa.gov/education/graphic_history/univ_evol.cfm

TensorFlow 2015, Google, http://www.tensorflow.org

Wang Y., Ramachandra N. S., Salazar E. M., 2020, Peculiar Velocity Estimation from Kinematic SZ Effect using Deep Neural Networks, Manuscript submitted for publication.

Weymann R., 1965, Physics of Fluids, 8, 2112

Wright E. L., 1979, ApJ, 232, 348

Wright E. L., 2004, in Freedman W. L., ed., Measuring and Modeling the Universe. p. 291 (`arXiv:astro-ph/0305591`)

Zeldovich Y. B., Sunyaev R. A., 1969, Ap&SS, 4, 301

Zeldovich Y. B., Sunyaev R. A., 1972, Comments on Ap&SS, 4, 173

Appendix A - Moment Generating Function

The following theorems refer to moment generating functions (Feller, 2008).

Theorem. *If a random variable X possesses a moment generating function $M_X(t)$, then*

$$\exists\, \mathbf{E}[X^n] \text{ such that } |\mathbf{E}[X^n]| < \infty \;\forall\, n \in \mathbb{N}$$

where $\mathbf{E}[X^n]$ the n-th moment of X. Furthermore:

$$\mathbf{E}[X^n] = \left.\frac{d^n M_X(t)}{dt^n}\right|_{t=0}$$

Theorem. *If two random variables X_1 and X_2 have the same moment generating functions, i.e., if $M_{X_1}(t) = M_{X_2}(t)$ for all t, then both have the same distribution.*

For any continuous random variable, its probability distribution function (PDF) is also continuous and well behaved.

Lemma. *If X is a continuous random variable for which all moments $\mathbf{E}[X^n]$ are known, then the probability distribution function is determined completely and can be reconstructed.*

Glossary

Notation	**Description**	**Page List**
activation function	Emulates a neuron's response.	31
back-propagation	The flow of information is directed from output to input. Backward pass starts from the cost computation back to the first hidden layer, while calculating all gradients.	36
cost	Performance measure. Scores the *full* error when using a model.	29, 33
feature	Represent some quality or quantity of the instance. Characteristic or attribute of an instance.	27, 31
forward propagation	The flow of information is directed from input to output. A single instance completes a forward pass through the network, where cost and prediction values are calculated.	36
hidden layer	Layer between input and output.	32
hyperparameter	Controls certain aspect of the behaviour of the learning algorithm and are arbitrarily set by the user.	34
hypothesis	Explains an example. Known *a priori*. Target value for the objective function.	27
instance	A particular case, example or single occurrence of the data.	27
instances dataset	Contains all available examples (instances).	27, 46

Notation	**Description**	**Page List**
knowledge	Parametric representation of the task to perform. Parameter set $\boldsymbol{\theta}$.	26
layer	Stack of parallel neurons.	32
learning	Process in which a computer program improves its performance at a given task overtime with experience. Determination of the *best* parameter set $\boldsymbol{\theta}$ that represents the knowledge of the task.	26, 29
learning rate	Step size for the optimization algorithm.	34
loss function	Score function measuring the *goodness* of the current parameter set.	29
minibatch	Subset of the instances dataset.	35
objective function	Holds the *true* relationship bewteen instance and target. It is unknown.	28
parameter set	Represents the knowledge of a task.	28, 33
prediction	Model output value.	28
prediction function	Maps a training example to a value y.	28
predictions dataset	Contains predicted values corresponding to all available examples.	28, 46
supervised learning	Machine learning algorithm in which a program learns a function that maps an input to an output based on example input-output pairs.	27
targets dataset	Contains target values or hypothesis corresponding to all available examples.	27, 46
test set	Contains a subset of the instances and targets datasets not used during training but to compute accuracy parameters.	52

Notation	Description	Page List
training set	Contains a subset of the instances and targets datasets used during training.	52