Universidad de Sonora
Maestría en Ciencias (Física)
Departamento de Investigación en Física
División de Ciencias Exactas y Naturales

SUNYAEV-ZEL'DOVICH EFFECT SIMULATED SIGNAL ANALYSIS USING MACHINE LEARNING

Submitted to the graduate degree program in Physics in partial fulfillment of the requirements for the Degree of Master of Sciences.

By
Edgar Martín Salazar Canizales

Director
Hume A. Feldman

Hermosillo, Sonora, ## July 2020

Printed July 2020

The Thesis Committee for Edgar Martín Salazar Canizales certifies that this is the approved version of the following thesis:

Sunyaev-Zel'dovich Effect Simulated Signal Analysis using Machine Learning

Date approved: _____

_____
Hume A. Feldman, Chair

_____
Carlos A. Calcáneo Roldán, Commitee member

_____
Committee member

_____
Committee member

This page intentionally left blank.

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

This page intentionally left blank.

Acknowledgments

[...] 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, Álvarez, Yeomans, Montoya and Rosas, who invested more time and effort in believing in me. To the University of Kansas Physics Department and the ISS office for being so attentive in providing human and material resources that allowed me to work on this thesis.

To CONACYT for funding my studies.

Edgar M. Salazar

This page intentionally left blank.

Abstract

NOTE: Review and change

Cosmic Microwave background photons are inverse Compton scattered and distorted by energetic electrons in galaxy clusters by a process known as the Sunyaev-Zel'dovich (SZ) effect. The SZ effect is expected to be instrumental in measuring the peculiar velocities of clusters in future surveys. We intend to simplify the traditional computations of peculiar velocities of galaxy clusters using an end-to-end deep learning architecture of parameter estimation and uncertainty quantification. To test our formalism, we use one of the largest Cosmological simulations that includes hydrodynamical effects, the Magneticum simulation, for our analysis. The training images of the distorted photon background are generated from this simulation using a light-ray tracing code for thermal and kinetic SZ effects.

NOTE: NOT FINAL

This page intentionally left blank.

Resumen

NOTE: Traducción al español

This page intentionally left blank.

Contents

List of Figures

List of Tables

List of Abbreviations

ΛCDM Lambda Cold Dark Matter.

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.

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.

NN Neural Network.

RF electron rest frame.

SGD Stocastic Gradient Descent.
SZ Sunyaev-Zel'dovich.

TSZ Thermal Sunyaev-Zel'dovich.

WMAP Wilkinson Microwave Anisotropy Probe.

Notation for Chapter 2

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 $m \times n$

Introduction

What can be said to exist? Throughout history humankind has sought to answer the ontological question and wonder about the genesis of existence. Early cultures, around 11,000−3,000 BP[1] (Diamond & Bellwood, 2003), throughout the world resolved in theism. They developed a *mythos* of superior beings who designed the universe: the container of all things. Western philosophy (2700−400 BP) gave way to a structured logical reasoning divided into natural, moral and metaphysical. The explanation of existence was delegated to metaphysics culminating with the cosmological argument which states that the first cause exists and is unique: a beginning to existence implicates the beginning of the universe and time. However, just as asking "what is north of the North Pole", this raises a serious problem with the question "what was before the first cause?" as the concept of "before" is meaningless in a situation without time.

The conceptualization of the universe continued to evolve and for centuries most of the western world envisaged it to be: eternal (infinite in time); finite in extent (surrounded by infinite void); steady state; and harmonic. Although some cultures at other parts of the world had their own theories to explain the first cause (e.g. Hindu Rigveda oscillating universe that expands and contracts to form a new one every couple billion years; and Sunni Muslim Fakhruddin al-Razi multiverse) they were also mostly based on theological arguments and not on science. There was no scientific evidence.

It wasn't until early 20th century, the *brewery of modern physics*, when two separate findings started a snowball of discoveries and theories that led to the establishment of physical cosmology as a formal field, and set the stage for a scientific theory of an expanding universe cosmogenesis. Vesto Slipher (1910) and Carl Wirtz (1918) observed a systematic redshift of spiral galaxies which is not explained by a universe model where stars are distributed more or less uniformly. Wirtz later published more articles suggesting that the redshifts of distant galaxies were becoming higher than those closer. This was the first evidence of an expanding universe. The Hubble-Lemaître law (with measurements of Edwin Hubble's (1924) and derivation of Georges Lemaître (1927)) is the conclusion to these observations and sates that the recessional velocity of galaxies is proportional to the proper distance to the observer.

[1] Before present. Specifies the number of years before radiocarbon dating began to be used in the 1950's.

General Relativity theory, published by Albert Einstein in 1915, describes the intrinsic relationship between space, time and matter in a set of field equations. These are crucial for describing how electromagnetic radiation is influenced by matter and gravity wells in its way to Earth. Einstein worked a cosmological constant Λ added to the field equations to account for the expansion or contraction of a static space, but later dropped it due to evidence that the universe is in fact expanding provided by Hubble. On other side of the world, Friedman (1922) derived the Friedmann equations for a spatially homogeneous and isotropic, curved and expanding space ($\Lambda > 0$).

If space expands then it must've been contracted all together at some point in time, right? This idea led Lemaître to hypothesize a cosmogenesis that began with the expansion of a "primeval atom". All there is in the universe initiated its existence from an infinitely dense and hot "singularity" that abruptly started its expansion. For this reason, Fred Hoyle unintentionally coined it the *Big Bang* theory later in 1959. Of course, more scientists contributed with observations and theoretical derivations but most findings support these ideas.

But, what evidence is there that suggests the *Big Bang* might've happened? In the midst of the century, George Gamow (1946) realized that after the recombination period, where electrons and protons became bound for the first time, the universe should be filled with black-body radiation: a cosmic background (CMB). Two decades later, this radiation field was first measured by Penzias & Wilson (1965) in the microwave regime wavelength from which they fitted for the temperature predicted by Gamow's theory. This observation can be seen as the culmination of efforts to support the *Big Bang* theory underpinnings.

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 imprint in the CMB anisotropies; distortions produced by under- and overdensities of matter at a period when radiation and matter were coupled. A lot of effort is done to model and explain what causes these distortions to the otherwise *completely homogeneous* signal. The CMB radiation field permeates the universe and is deformed by several known effects such as: Doppler shift (earth's relative motion), gravitational lensing (mass overdensities), 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.

It's clear there's some avail in studying CMB anisotropies directly, in particular the SZ effect which gives information of mass distribution. In fact, thermal SZ 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. Whilst, a kinematic contribution to SZ power spectrum is particularly useful to estimate galaxy cluster peculiar velocities as its directly proportional to the negative of the velocity (Sunyaev & Zeldovich, 1980). Furthermore, SZ is nearly redshift independent. This is a very desirable and uncommon feature that makes this effect a valuable cosmological and cluster evolutionary diagnostic tool (Rephaeli, 1995).

The last fifty years since Penzias and Wilson's first observation of CMB have yielded amazing results. However, high fidelity data from outer space probes (refer to COBE, WMAP and Planck collaborations) is hard to come by, not to mention expensive. Besides, SZ signal observation doesn't provide all parameters needed to extract peculiar velocities from it. For example, the electron density in the intra cluster medium (which must be obtained by other techniques) is required to determine the optical depth and comptonization parameters. With the ever improving computational capabilities on modern day computers, scientists have resorted to simulations based on observed data and theoretical models to be able to study the observable universe and predict its evolution. Some cluster catalog examples are: Millenium (2005, 2009, 2010), Illustris (public data release 2015), and Magneticum (2015). Simulations have the undeniably advantage of giving complete control over cosmological parameters, are free of observational systematics, and far less expensive than outer space probes. Nonetheless, amount of data available becomes a problem because it stacks together with observations and, therefore, magnify the task of data analysis.

Recent work with SZ effect is done to calibrate methods using simulated SZ signal as a probe for cosmology and calculate peculiar velocities, using computational techniques (e.g. signal filtering) to estimate and analyze optical depth and comptonization. For example, Soergel et al. (2018) use adaptive aperture signal map filtering to obtain the average SZ signal for a single cluster; then, bins all clusters in redshift slices, and computes the mean optical depth and comptonization parameter that represents each slice.

Using mean values for redshift slices produce inaccurate estimation of the peculiar velocity for a single cluster, and thus non-reliable pairwise velocity estimator. This estimator [...]

An alternative signal processing algorithm is found in Machine Learning. These algorithms are designed to perform complex analyses in a data-driven manner instead of executing explicit computational of tasks. This allows to develop and calibrate models with unbiased data sets, free from systematic errors and possible unwanted effects. The usage of Machine Learn-

ing algorithms to exempt explicit parameter estimation and signal filtering of SZ effect maps would improve peculiar velocity prediction for single clusters. An automatized computer program will be able to predict the peculiar velocity of a single galaxy cluster from simulated signal maps.

Objectives and tasks. Must edit.

The general objective is to [...] We intend to simplify the traditional computations of peculiar velocities of galaxy clusters using an end-to-end deep learning architecture of parameter estimation and uncertainty quantification.

1. Obtain by explicit derivation the Sunyaev-Zel'dovich equations for both tSZ and kSZ effects.
2. Program and test Neural Networks using TensorFlow to obtain peculiar velocities from simulated signal maps.

In order to achieve this, the following tasks are performed [...] To test our formalism, we use one of the largest Cosmological simulations that includes hydrodynamical effects, the Magneticum simulation, for our analysis. The training images of the distorted photon background are generated from this simulation using a light-ray tracing code for thermal and kinetic SZ effects. The expected tasks to perform include:

- Review the relevant bibliography for the study of the Sunyaev-Zel'dovich effect.
- Develop a model to train with a sample of kSZ signals from a single snapshot of the Magneticum Simulation.
- Build and calibrate different architectures to tune the model.
- Develop a model to train with a sample of both kSZ and tSZ signals from a single snapshot of the Magneticum Simulation.
- Build and calibrate different architectures to tune the model.
- Run the selected models with data from 4 snapshots with 10,000 samples each from the Magneticum Simulation.

Thesis outline; described by chapters

In chapter 1 the reader will find a review of the current cosmological theory and its model known as ΛCDM. Cosmic background radiation is presented as the result of density fluctuations preserved from the recombination and last scatter era in the universe evolution. Some of the most relevant effects known to produce anisotropies are discussed, including the SZ effect. This chapter lays out a background to study the physics related to this work and the 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, based on a review done by Birkinshaw (1999).

Chapter 2 provides the necessary concepts of the Machine Learning field. It covers the core idea of an Artificial Neural Network, a learning algorithm known as back-propagation, and the particular implementation of a convolutional neural network. Finally, TensorFlow API, the tool used for programming a learning algorithm, is briefly described.

In chapter 3, the Magneticum simulations from which KSZ and TSZ signal maps were extracted are first described. Next, the network architecture that is used to train the learning algorithm is defined based on (thoroughly tested) known models (Hasanpour et al., 2016). To show if convolutional neural networks are best suited for the Magneticum simulations signal maps, some image transformations are explored in order to test simple deep neural networks. At the end of this chapter the reader will find a description of the cross validation method for regression and some estimators that are useful to measure the performance of the learning algorithm implemented in this work. This chapter uses the concepts and notation from previous chapters extensively, so it is recommended to read them fist (at least chapter 3).

Finally, chapters 4 and 5 go over the results obtained and conclusions for this work.

Chapter 1 – Cosmology

This first chapter introduce the reader to cosmology by briefly presenting the current standard model and how it describes the observable universe, with a quick review of CMB observations to 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 *Big Bang*. This theory depends on two assumptions: the universality of physical laws[1] and the cosmological principle[2]. It describes how the universe expanded from a state of immeasurable high density and temperature, nearly homogeneous mixture of photons and matter tightly coupled, to a structured expanding universe. 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. Inflation is believed to be the reason why the universe seems to be basically isotropic and flat, although initial conditions of the early plasma are tough 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 acustic waves producing under- and overdensities in the plasma 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

[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.

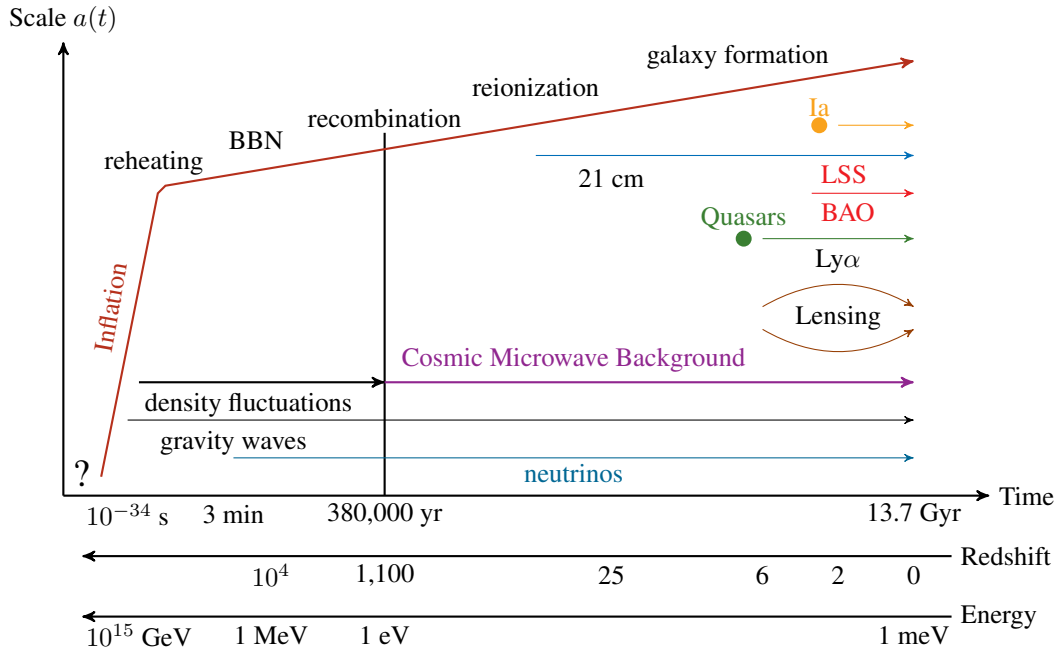


Figure 1.1: History of the universe. This schematic highlights key events in the history of the universe and their associated time and energy scales (adapted Fig. 2 from Baumann, 2009). The question mark at the origin represents all unknown processes occurred at inflation. Acronyms: BBN (Big Bang Nucleosynthesis),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

the photons and baryons (Hu & 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, and 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 Cosmic Microwave Background (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 overdense 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 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, thus the universe expands.

Einstein introduces 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 spacetime; $k = 1$ 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 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. The former is a measure of space expansion whilst the latter, of distance and/or time.

As described by General Relativity and the FLRW metric, the Lambda Cold Dark Matter (ΛCDM) model is a parametrization of the *Big Bang* theory and encompasses the assumption of existence of radiation, ordinary matter (leptons and quarks), cold dark matter, and a negative pressure or soft density of empty space (sometimes called "dark energy") 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 is the scale parameter at time t; and H_0 is the and Hubble parameter. Its common practice to use the reduced Hubble constant h as adimensional 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 $t = t_0$ recedes at $70\,\mathrm{km\,s^{-1}}$.

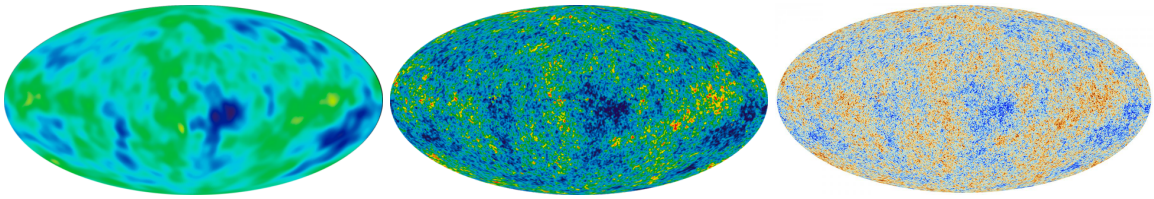
Table 1.1 shows the latest results (Hinshaw et al., 2013; White et al., 2018) for ΛCDM parameters. The first three specify the corresponding density parameters assuming the total density $\Omega = 1$. The curvature fluctuation amplitude measures the deviation of k from a flat spacetime. In this model field fluctuations are the root cause of the observed CMB anisotropies which are characterized by the primordial scalar perturbations as a power law $\propto k^{n_s-1}$, thus n_s is a critical parameter for characterizing the strength of the 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}; its 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 reionization redshift. Lastly, the amplitude of matter density fluctuations parameter defined as the RMS of the $z = 0$ density perturbations on scales of $8\ h^{-1}\ \mathrm{Mpc}$, σ_8 tells how matter is distributed at that scale.

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

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	$0.820(14)$	$0.8102(60)$

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.

[4]Sloan Digital Sky Survey.



(a) COBE (b) WMAP (c) Planck

Figure 1.2: Comparison between CMB density contrast sky surveys improving over 20 years[5]. 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.

1.2 Cosmic Background Radiation

In 1946, 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. 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 spectrum and 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 *Big Bang* theory. It also provided an estimate to the order 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). This 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 launched the Planck probe (Planck Collaboration, 2016) which took high resolution measurements, being the first to distinguish details in the structure of the CMB that were not observable 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 take this result from Fixsen, 2009, with neglible impact on their results). Figure 1.2 shows how temperature density contrast $\Delta T/T$ measurement resolution have improved over the years with outer space probes. Although it might seem as if the spectrum is not isotropic, anisotropies are very small with variations smaller than $10^{-5}\,\mathrm{K}$ (Wright, 2004).

As mentioned before, CMB is a radiation field of photons that went out of thermal equilibrium

with matter in the early universe after it became transparent, around $z_{\text{reion}} \approx 1100$. This is often called the *last scattering*, referring to when photons were scattered by hot plasma electrons in the early universe for the "last time", constituting the *surface of the last scatter*, i.e. CMB. For the most part, this field is homogeneous and isotropic, this means that early universe distribution of matter and radiation is also homogeneous and isotropic. However, matter under- and overdensities 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). Other effects also deform the field in its path to an observer at Earth and are explored through a harmonic expansion of temperature fluctuations $\frac{\Delta T(\hat{n})}{T_0} = a_{\ell m} Y_{\ell m}(\hat{n})$ [6]. The angular power spectrum is an important tool in the statistical analysis of the CMB. It describes the cosmological information contained in 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 average temperature of CMB 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) fluctuations give the anisotropies observed (see Fig. 1.2) and contain information on the mass distribution around the universe as well as 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 $10^{14}\,\mathrm{M}_\odot$ and radii around Mpc. The total gas fraction is about 16% with about 13% in the hot intra-cluster medium (ICM) and the remaining 3% in stars in the cluster galaxies. The remaining 84% of the mass is in dark matter halo. Gas densities in cluster centers range from as much as 10^{-1} to 10^{-3} particles per 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 cm^3 (Peterson & Fabian, 2006).

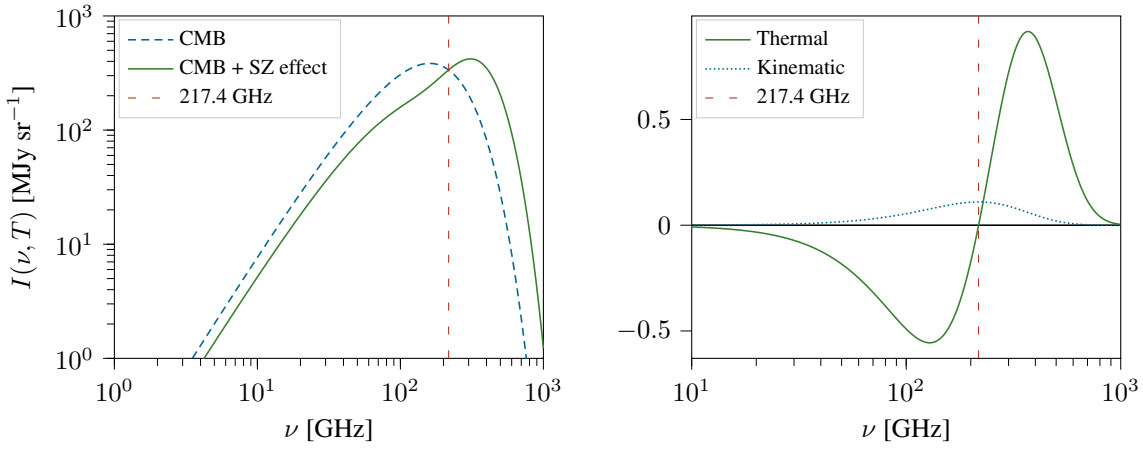
[5] Images retrieved from http://www.nasa.gov and https://www.cosmos.esa.int.

[6] Assume Einstein summation convention for repeated indices.

The ICM is plasma that is nearly fully ionized due to the high temperatures created by the deep dark matter gravitational potential. Consists mainly of ionized hydrogen and helium and strongly emits X-Ray radiation, primarily due to bremsstrahlung process. Free electrons in the medium can scatter low energy CMB photons and cause distortions to its spectrum. This is known as the Sunyaev-Zel'dovich (SZ) effect.

Few years after Penzias and Wilson's measurements, Zeldovich & Sunyaev (1969) were able to find an analytical expression in the non-relativistic approximation studied by Kompaneets (1956) for the distortion of the CMB spectrum produced by the scattering of thermal electrons. This is formally known as the Thermal Sunyaev-Zel'dovich (TSZ) effect and it is a 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. The 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 GHz (see Fig. 1.3b).

Additionally, a kinematic contribution (Doppler shift) in the spectrum caused by the ICM bulk radial motion with respect to the CMB rest frame known as Kinematic Sunyaev-Zel'dovich (KSZ). This effect shifts the CMB-Planck spectrum to a slightly lower (higher) temperature for receding (approaching) velocities from the observer (Birkinshaw, 1999). In fact, the



(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 because its around two orders of magnitude smaller than TSZ.

Figure 1.3: Sunyaev-Zeldovich effect distortion to CMB black-body spectrum.

minimum (maximum) drop (raise) in intensity is at the crossover frequency also. This makes the measurement of SZ effect to be around the 217 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 only.

The most interesting aspect of KSZ is that it provides a method for measuring one component of the 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 motion 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 precision measurement of the velocity field is limited by the error in the measurement of radial distance, which tend to have a non-Gaussian error distribution that may bias the results. Furthermore, the fractional observational errors of the radial distances are typically around 20% and peculiar velocities tend to have errors proportional to the distances, which may be large. For this, a single peculiar velocity measurement is not a good approximation of the velocity of a galaxy. Statistical ensembles (especially of low-order moment statistics) may be a good estimator of the cosmic velocity field and thus 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. It's 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 analyzed are unique to each catalogue it is difficult to compare the bulk flow calculated using this method 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 large-scale velocity field is the pairwise velocity statistic v_{12} intro-

duced 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 $r = |\mathbf{r}_j - \mathbf{r}_i|$.

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

Wang et al. (2018) further discuss a peculiar velocity correlation function but for the sake of simplicity this work sticks to the pairwise velocity estimator. NOTE: What cosmological parameter is extracted or derived form this estimator?

1.4 Sunyaev- Zel'dovich Effect

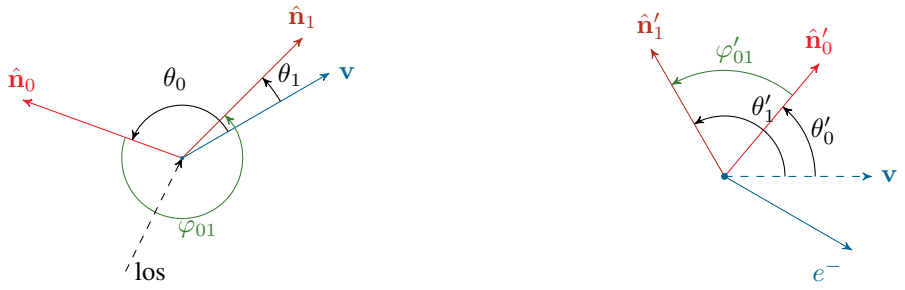
The CMB is the dominant electromagnetic radiation field in the Universe, and Compton scattering is one of the main physical processes that couples radiation to matter. Scattering of CMB photons by free electrons

$$e_0 + \gamma_0 \longrightarrow e_1 + \gamma_1 \tag{1.3}$$

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 obtainment 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 given by Eq. (1.3) 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



(a) Laboratory frame of reference S (LF).

(b) Electron rest frame S' (RF).

Figure 1.4: Compton scatter of a photon by an electron.

S. Note that the subscript 0 will not be necessary as shown below. Fig. 1.4a shows how every vector is measured starting from the electron velocity.

Four-momentum conservation states:

$$P_0 + Q_0 = P_1 + Q_1 \tag{1.4}$$

Using the Minkowski norm invariance its possible to express the final state in terms of the initial state by taking the norm squared on both sides of Eq. (1.4), getting $P_0 \cdot Q_0 = P_1 \cdot Q_1$. Multiplying Eq. (1.4) by P_1 from the left on both sides and considering the previous result, evaluates to the equivalent explicit expression:

$$c^{-2}\epsilon_1\epsilon_0\left(\hat{\mathbf{n}}_1 \cdot \hat{\mathbf{n}}_0 - 1\right) + c^{-1}\epsilon_1\gamma m\left(\hat{\mathbf{n}}_1 \cdot \mathbf{v} - c\right) = c^{-1}\epsilon_0\gamma m\left(\hat{\mathbf{n}}_0 \cdot \mathbf{v} - c\right) \tag{1.5}$$

This expression is independent of the electron's final state, i.e. it does not affect the scattered photon, which is valid 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 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 $v\cos\theta_1 = \hat{\mathbf{n}}_1 \cdot \hat{\mathbf{v}}$ (see Fig. 1.4a). Solving Eq. (1.5) for the scattered energy ϵ_1:

$$\epsilon_1 = \epsilon_0 \frac{1 - \beta\cos\theta_0}{1 - \beta\cos\theta_1 + \dfrac{\epsilon_0}{\gamma mc^2}\left(1 - \cos\varphi_{01}\right)} \tag{1.6}$$

This is the change in photon frequency 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 $E' = mc^2$:

$$\epsilon'_1\left(\epsilon'_0, \Omega\right) = \epsilon'_0 \frac{1}{1 + \frac{\epsilon'_0}{E'}\left(1 - \cos\varphi'_{01}\right)} \tag{1.7}$$

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 transformation to energies in both reference frames are:

$$\begin{aligned} \epsilon' &= \epsilon\gamma\left(1 - \beta\cos\theta\right) \\ \epsilon &= \epsilon'\gamma\left(1 + \beta\cos\theta'\right) \end{aligned} \tag{1.8}$$

The transformation law for the angles relative to the electron's velocity:

$$\cos\theta' = \frac{\cos\theta - \beta}{1 - \beta\cos\theta} \tag{1.9}$$

Finally, the deflection angle:

$$\cos\varphi_{01} = \cos\theta_1\cos\theta_0 + \sin\theta_1\sin\theta_0\cos(\phi_0 - \phi_1) \tag{1.10}$$

where ϕ_1 and ϕ_0 are the azimuthal angles of the scattered photon and incident photon in S (see Fig. 1.4).

1.4.2 Inverse Compton Scattering

Since $\varphi'_{01} \in [-\pi, \pi]$, i.e. photons can come from anywhere, energy is lost from the recoil except for purely forward scattering, as Fig. 1.5 shows comparing the photon incoming energy to the electron's rest mass. 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 math.

Whenever the moving electron has sufficient kinetic energy compared to the photon, net energy may be transferred from the electron to the photon, thus the outgoing photon energy

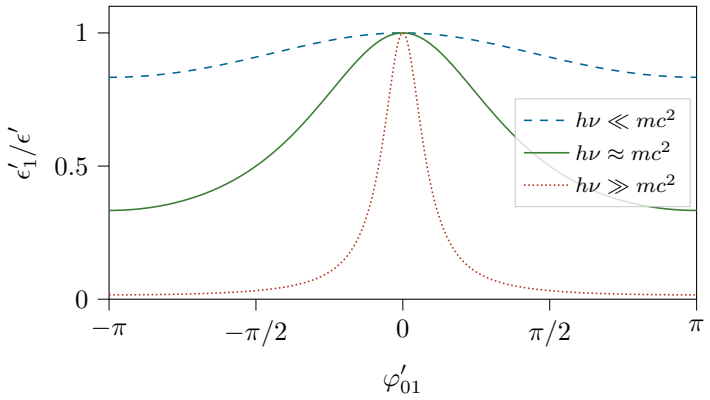


Figure 1.5: Fractional frequency change of the scattered photon measured from RF.

(frequency) is higher and it is said to be upscattered, and 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.

How much larger is the scattered photon energy? 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 Eqs. (1.8) and (1.7), typical scattering angles of $\pi/2$, taking into account the considerations previously made for inverse Compton scatter, and working back to LF quantities its possible to approximate

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

Although a very rough estimation, it shows that even if the incoming photon energy in RF is low, as long as the electron has a large kinetic energy, the energy of the outgoing photon can increase in LF. Because $\gamma \gg 1$, the upscatter can give very big 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 for single scattering

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 $d\epsilon$ [7]. 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

[7] In the following it will be assumed that no subscript refers to the incoming or before scattering value and will be used indistinctly.

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

$$P' = c\sigma_{\rm T} \int \epsilon' v' d\epsilon' \tag{1.11}$$

The 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 a 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:

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

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 $\langle (1 - \beta\cos\theta_0)^2 \rangle = 1 + \frac{1}{3}\beta^2$, thus

$$P = \gamma^2 c\sigma_{\rm T} \left(1 + \frac{1}{3}\beta^2\right) U_{\rm ph} \quad \text{with} \quad U_{\rm ph} = \int \epsilon v d\epsilon \tag{1.12}$$

Here, $U_{\rm ph}$ is the initial photon energy density, whose decrease rate is simply $-c\sigma_{\rm T} U_{\rm ph}$. The net power lost by the electron and thereby converted into increased radiation is:

$$P_{\rm compt} = \frac{4}{3} c\sigma_{\rm T} \gamma^2 \beta^2 U_{\rm ph}$$

Thermal energy is related to β by $\langle \beta^2 \rangle = \frac{3kT}{mc^2}$ for an electron cloud at absolute temperature T. So the total power for a thermal distribution of nonrelativistic electrons with number density n_e at equilibrium temperature T is

$$P_{\rm compt} = 4\Theta c\sigma_{\rm T} n_e U_{\rm ph} \qquad \text{with} \qquad \Theta = \frac{kT}{mc^2} \tag{1.13}$$

where Θ is the electron's dimensionless equilibrium temperature (given by kT). Now, the average energy of a photon due to ICM scatterings with thermal electrons at the nonrelativistic 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.

$$\Delta\epsilon = \frac{P_{\rm compt}}{c\sigma_{\rm T} n_e} = \frac{4}{3}\gamma^2\beta^2\langle\epsilon\rangle$$

This gives an energy gain:

$$\frac{\Delta\epsilon}{\epsilon} \sim \frac{4}{3}\beta^2 = 4\Theta \tag{1.14}$$

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:

$$\frac{\Delta\epsilon}{\epsilon} = 4\Theta - \frac{\epsilon}{mc^2} \tag{1.15}$$

From here we have the following limit cases:

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

Next thing to do is inspect how this affects the photon number over time.

1.4.4 Non-Relativistic Limit: Kompaneets Equation

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

$$\frac{\partial n}{\partial t} = -\, c \int d^3p\, d\Omega\, \frac{d\sigma}{d\Omega}\, \left[n(\nu) f(\mathbf{p})\left(1 + n(\nu_1)\right) - n(\nu_1) f(\mathbf{p}_1)\left(1 + n(\nu)\right)\right] \tag{1.16}$$

considering the scattering events $\mathbf{p} + \nu \rightleftharpoons \mathbf{p}_1 + \nu_1$ (Peebles, 1993). The first term inside brackets in Eq. (1.16) 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 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:

$$\frac{d\sigma_{\mathrm{T}}}{d\Omega} = \frac{1}{2} r_0^2 \left(1 + \cos^2\theta\right) \tag{1.17}$$

where r_0 is the classical electron radius:

$$r_0 = \frac{1}{4\pi\epsilon_0}\frac{e^2}{mc^2} = 2.818 \times 10^{-15}\,\mathrm{m}$$

and σ_{T} is Thomson's cross section:

$$\sigma_{\mathrm{T}} = \frac{8\pi r_0^2}{3} \approx 6.6524 \times 10^{-29}\,\mathrm{m}^2 = 0.66\,\mathrm{b}$$

A solution to Eq. (1.16) in the non-relativistic second order approximation for thermal electron population or Fokker-Planck approximation can be derived as follows. Consider that CMB photons are in thermal equilibrium once they enter the ICM at an absolute temperature of Θ. The electron phase space density $f(\mathbf{p})$, alternatively $f(E)$ where $E = \mathbf{p}^2/2m$ is given by

$$f(\mathbf{p}) = \left(2\pi m^2 c^2 \Theta\right)^{-3/2}\ n_e\ \exp\left[-\frac{\mathbf{p}^2}{2m^2c^2\Theta}\right] \tag{1.18}$$

where n_e is the electron space density and $f(\mathbf{p})$ 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:

$$x = \frac{h\nu}{kT} \qquad \text{and} \qquad \Delta = \frac{h\Delta\nu}{kT} \tag{1.19}$$

where T is the radiation field temperature (CMB)[8]. An increase in frequency is expected due to inverse Compton scattering so for the occupation number of photons:

$$n(\nu_1) = n(\nu) + \Delta\frac{\partial n}{\partial x} + \frac{1}{2}\Delta^2\frac{\partial^2 n}{\partial x^2} + \cdots \tag{1.20}$$

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

And a decrease in electron energy[9]:

$$f(E_1) = f(E)\left[1 + \Delta + \frac{1}{2}\Delta^2 + \cdots\right] \tag{1.21}$$

Using Eqs. (1.20) and (1.21) in Eq. (1.16):

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

It possible to anticipate (as expected) that the Bose-Einstein distribution $n(x) = (e^{x+\alpha} - 1)^{-1}$ 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. (1.17), 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 $\sigma_{\rm T} n_e c \Delta\epsilon/kT$. The average energy transfer per collision of photons with energy ϵ is given by Eq. (1.15), which can be rewritten:

$$\frac{\Delta\epsilon}{kT} = \Theta\; x\,(4 - x)$$

Thereby:

$$\mathcal{I}_1 = c\sigma_{\rm T} n_e \Theta\; x\,(4 - x) \tag{1.22}$$

The second integral $\mathcal{I}_2$ (in red) can be obtained by direct calculation.

$$\mathcal{I}_2 = 2c\sigma_{\rm T} n_e \Theta\; x^2 \tag{1.23}$$

Using these results, the equation transforms to:

$$\frac{\partial n}{\partial t} = \frac{\Theta\sigma_{\rm T} n_e c}{x^2}\frac{\partial}{\partial x}\left[x^4\left(\frac{\partial n}{\partial x} + n(n+1)\right)\right] \tag{1.24}$$

This is the Kompaneets equation (Kompaneets, 1956; Weymann, 1965). Using random walk arguments its possible to analyze 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

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

the original spectrum of photons, i.e. Comptonization. The mean free path of the photon due to Thomson scattering is $\lambda = \frac{1}{n_e \sigma_{\rm T}}$. If the size of the region ℓ is such that $\lambda^{-1} \gg 1$, then the photon will undergo several collisions in this region, but if $\lambda^{-1} \ll 1$ there will be few collisions. The optical depth is defined as $\tau \equiv \lambda^{-1} = n_e \sigma_{\rm T} \ell$, so that $\tau \gg 1$ implies strong scattering.

If $\tau \gg 1$ (optically thick), then the photon undergoes N ($\gg 1$) collisions in traveling a distance ℓ. From standard random-walk arguments, we have $N = \tau^2$. 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. $N = \tau$. Therefore an estimate for the number of scattering is $N = \max(\tau, \tau^2)$. The average fractional change in the photon energy per collision is 4Θ, hence the condition for significant change of energy is $1 \simeq 4\Theta \max(\tau, \tau^2)$.

The optical depth value is not trivial because it requires knowing the electron density in the dispersive region. Some rich clusters have values around $\tau \sim 0.02 - 0.03$. The Comptonization parameter is defined as:

$$y = \int \Theta \, d\tau = \int \Theta n_e \sigma_{\rm T} \, d\ell \tag{1.25}$$

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 in several ways: it is proportional to the electron gas pressure $n_e kT$; it is the product of the scattering optical depth and a weighted average of the electron temperature in units of electron mass Θ (Peebles, 1993).

The occupation number change is sometimes expressed as a function of the comptonization parameter, so after integrating both sides of Eq. (1.24) along the path length throughout the cluster, considering Eq. (1.25), one gets a solution to the Boltzmann equation:

$$\Delta n(x, y) = \frac{y}{x^2} \frac{\partial}{\partial x} \left[x^4 \left(\frac{\partial n}{\partial x} + n(n+1) \right) \right] \tag{1.26}$$

For black-body radiation as it is the case of the CMBR with temperature T we have that the occupation number of photons is actually:

$$n\,(x) = \frac{1}{e^x - 1} \tag{1.27}$$

Which corresponds to the previous assumption that it is Planckian.

1.4.5 Sunyaev-Zel'dovich Effect

Equation (1.27) constitutes a steady-state solution, thus the parenthesis in Eq. (1.26) vanish resulting in $\Delta n = 0$. 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. (1.26) simplifies:

$$\Delta n(x, y) = \frac{y}{x^2}\frac{\partial}{\partial x}\left[x^4 \frac{\partial n}{\partial x}\right] \tag{1.28}$$

And the photon occupancy change is:

$$\Delta n(x, y) = y\frac{xe^x}{(e^x - 1)^2}\left[x \coth\left(\frac{x}{2}\right) - 4\right] \tag{1.29}$$

Albeit this is the solution, occupation number is not an observable quantity but the spectral intensity. Given that Eq. (1.26) has been integrated along the path length throughout the cluster, 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

$$\Delta I_t = i_0\, y\, \frac{x^4 e^x}{(e^x - 1)^2}\left[x \coth\left(\frac{x}{2}\right) - 4\right] \qquad \text{with} \qquad i_0 = 2\frac{(kT)^3}{(hc)^2} \tag{1.30}$$

The CMB radiation field is not distorted before arriving to the cluster, so it's simply given by the photon occupation number $I = i_0 x^3 n$. Taking the fractional change in intensity of CMB photons due to free electrons in the ICM, TSZ is:

$$\frac{\Delta I_t}{I} = y\frac{xe^x}{e^x - 1}\left[x \coth\left(\frac{x}{2}\right) - 4\right] \tag{1.31}$$

Equivalently:

$$\frac{\Delta T_t}{T} = 2y\left[\frac{x}{2}\coth\left(\frac{x}{2}\right) - 2\right] \tag{1.32}$$

From $\frac{h}{kT}$, the scale in frequency is around GHz so that the effect is noticeable.

$$x = 1.762\left(\frac{\nu}{100\,\mathrm{GHz}}\right) \tag{1.33}$$

Equation (1.30) has a zero at $x_0 = 3.83$ or $\nu_0 = 217.4\,\mathrm{GHz}$. Furthermore, ΔI 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.1.6). In the Raleigh-Jeans limit TSZ approximates to $-2y$.

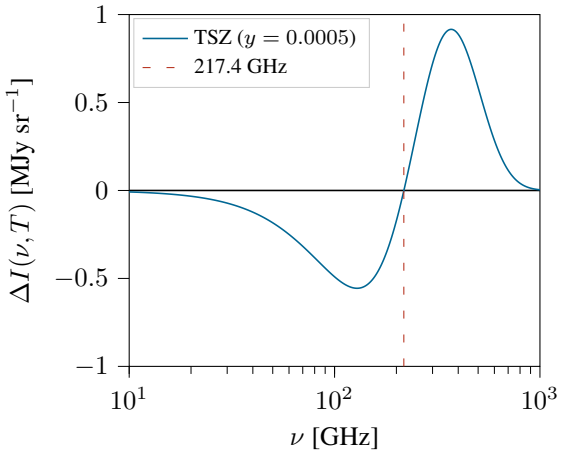


Figure 1.6: TSZ spectrum corresponding to $y = 0.0005$ with cut frequency at 217.4 GHz.

A contribution to the distortion of the spectrum done purely by scattering is due to the relative motion of the ICG to LF. From the radiative transfer equation is possible to obtain this contribution (Birkinshaw, 1999).

$$c^{-1}\frac{dI}{dt} = j_\nu - (\alpha_{\nu,\mathrm{abs}} + \alpha_{\nu,\mathrm{sca}})\, I(\hat{\mathbf{k}}_1) + \alpha_{\nu,\mathrm{sca}} \int \phi(\hat{\mathbf{k}}, \hat{\mathbf{k}}_1) I(\hat{\mathbf{k}}) d\Omega \tag{1.34}$$

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 the probability ϕ of scattering to an angle θ_1 specified by the wave vector $\hat{\mathbf{k}}$ (given by Chandrasekhar, 1950).

$$\phi(\mu, \mu_1) d\mu = \frac{3}{8}\left[1 + \mu^2\mu_1^2 + \frac{1}{2}\left(1 - \mu^2\right)\left(1 - \mu_1^2\right)\right] d\mu \tag{1.35}$$

where μ and μ_1 are the cosine angles of the incoming and scattered photons. Considering the change in specific intensity due solely by scattering, i.e. no emission nor absorption, and changing to a dependence on the optical depth along the path length in los direction:

$$\frac{dI}{d\tau} = \int_{-1}^{1} \phi(\mu, \mu_1)\,(I - I_1)\, d\mu \tag{1.36}$$

where the optical depth is $\tau = \int \alpha_{\nu,\mathrm{sca}}\, d\ell_{\mathrm{los}}$. By integrating the left side of Eq. (1.36) it's possible to rewrite it as a relativistic invariant $\Delta I/I$ where I will correspond to the CMB radiation field spectrum just as before. Since the observer at LF observes scattered photons

along the los, $\mu_1 = 1$. Moreover, the integral even over an even function (cosines) meaning that only even powers of μ will survive.

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

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, resulting:

$$\frac{\Delta I_k}{I} = -\tau\beta \frac{x e^x}{e^x - 1} \tag{1.37}$$

Or simply:

$$\frac{\Delta T_k}{T} = -\tau \frac{v}{c} \tag{1.38}$$

where $v = v_{\rm los}$ is the cluster line-of-sight peculiar velocity.

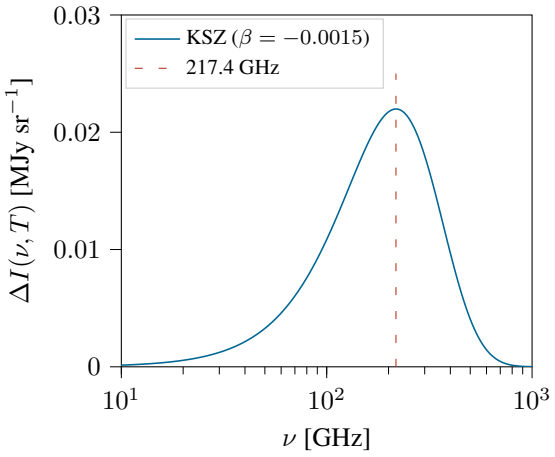


Figure 1.7: KSZ spectrum corresponding to $v \approx -450\,\mathrm{km\,s^{-1}}$ (approaching to the observer) with maximum at the crossover frequency 217.4 GHz.

It would be very difficult to locate the KSZ effect in the presence of TSZ at low frequency. The ratio of the brightness temperature changes caused by the effects is

$$\frac{\Delta T_k}{\Delta T_t} = \frac{1}{2\Theta} \frac{v}{c} = 0.085 \left(\frac{v}{1000\,\mathrm{km\,s^{-1}}} \right) \left(\frac{kT}{10\,\mathrm{keV}} \right)^{-1} \tag{1.39}$$

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. However, TSZ and KSZ effects may be separated using their different spectra: indeed, in the Kompaneets approximation is shown that the kinematic effect produces its maximum intensity change (Fig. 1.7) at the frequency at which the thermal effect is zero (Fig. 1.6).

1.4.6 Relativistic Limit

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

$$s = \ln\left(\frac{\nu_1}{\nu}\right) = \ln\left(\frac{1+\beta\mu_1}{1+\beta\mu}\right) \tag{1.40}$$

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

$$\mathcal{P}(s,\beta) = \frac{1}{2\gamma^4\beta}\int \frac{e^s\,\phi(\mu,\mu_1)}{(1+\beta\mu)^2}d\mu \tag{1.41}$$

where $\phi(\mu,\mu_1)$ is given by Eq. (1.35). Averaging over a Maxwellian distribution for the electrons:

$$\mathcal{P}_1(s) = \frac{\int \beta^2\gamma^5 e^{-\frac{\gamma-1}{\Theta}}\mathcal{P}(s,\beta)d\beta}{\int \beta^2\gamma^5 e^{-\frac{\gamma-1}{\Theta}}d\beta} \tag{1.42}$$

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

$$\Delta n_t(x) = \tau\int_{-\infty}^{\infty}\left[n(xe^s) - n(x)\right]\mathcal{P}_1(s)ds \tag{1.43}$$

The intensity change is calculated as before simply by $\Delta I_t(x) = i_0 x^3 \Delta n_t(x)$. In the non-relativistic limit the Kompaneets equation led to Eq. (1.30) which can be written as $\Delta I_t = i_0 y g(x)$, where the dependence on the cluster gas density n_e and temperature is in the comptonization parameter, and $g(x)$ is the spectral function:

$$g(x) = h(x)\left[x\coth\left(\frac{x}{2}\right) - 4\right] \qquad \text{with} \quad h(x) = \frac{x^4 e^x}{\left(e^x-1\right)^2} \tag{1.44}$$

Whereas the motion of the cluster in the CMB frame induces the kinematic SZ component (Doppler), which was readily calculated for the non-relativistic limit in Eq. (1.38) given as $\Delta I_k = -i_0 \tau \beta h(x)$.

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 expression to be accurate to within $\sim$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

$$\frac{\Delta I}{i_0} = \tau \sum_{i=1}^{8} f_i(x)\Theta^i \; + \; \tau^2 \sum_{i=1}^{4} f_{i+8}(x)\Theta^{i+1} \; - \; \tau\beta \left[h(x) + \sum_{i=1}^{4} f_{i+12}(x)\Theta^i \right] \tag{1.45}$$

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:

$$x_0 = 3.83 \left(1 + 1.12\,\Theta + 2.08\,\Theta^2 - 80.74\,\Theta^3 + 1548.25\,\Theta^4 + 0.8\,\tau\Theta + 1.18\,\tau\Theta\right) \tag{1.46}$$

The analytic expression given here is accurate to within a few percent. Take for example a typical cluster temperature of $10\,\mathrm{keV}$ and first order KSZ correction in Θ, which is given by $f_{13}(x)$:

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

Be as it may, the theoretical description accuracy of the SZ effect surpasses the precision with which the effect can be measured at present. 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 anisotropy of Eq. (1.38) suffices as a model for cluster peculiar velocities.

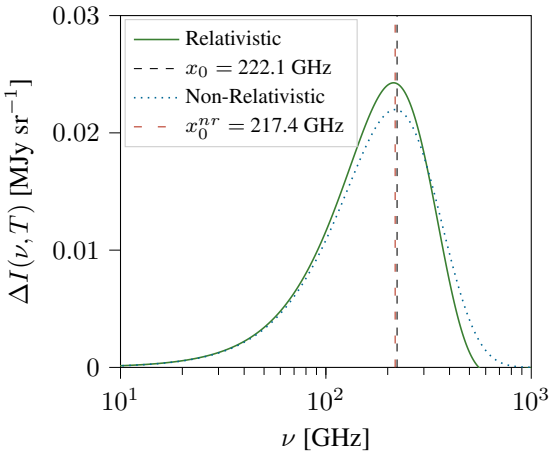


Figure 1.8: Relativistic KSZ spectrum with first order correction in Θ.

Chapter 2 – Machine 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 question 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. His goal is not to analyze 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.

2.1 Supervised Learning

The design of a learning system requires the following steps. First, 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 success or failure of the learner, and it basically depends on the training examples available. Is it a direct representation of the data? Does it have a broad distribution of examples available? In general, learning is most reliable when the training examples follow a distribution similar to that of future test examples. Finally, set a performance measure (e.g. percent of correct classifications) to know how well the program is doing. One key attribute is whether the training experience provides direct or indirect feedback regarding the choices made by the performance system. Is there a way to determine the degree to which a certain way of performing a task is better?

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 because its supposed to explain the example. This type of learning can be

implemented to automatize complex tasks of classification or regression, and thrives with a large amount of examples.

The present work aims to do regression for images. This means that the task to perform is regression. And, because the data available is both image and corresponding target, supervised learning is most appropriate. The program's performance can be evaluated with respect to it. All that's left is to choose a performance measure, but some definitions will be introduced first to adopt ML lexicon.

Consider the set of n real values x_{ij} called features representing an instance $\mathbf{X}$ 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 $\mathbf{X}$ can be arranged into different structures depending on how many dimensions the instance has. For example, a full color image is represented by a 3-D array, where two dimensions are width and height, and the third has color channels. The idea of tensor comes to mind as arbitrary dimensional arrays (Abadi et al., 2015). In fact, ML techniques exploit this concept and represent all datasets, variables and methods involved in a computation as tensors. Section 2.5 engages this idea further.

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. If the example number is very limited, it may be difficult to train a reliable model. It is common to find that complex examples require large datasets. Now, the targets dataset can be defined containing the corresponding truth to every instance.

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

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

When classifying amongst several categories, say n_{cat}, each label can be converted to an integer representing it where the last category is n_{cat}. Some NNs have faster convergence when the labels are in vectorized with the canonical base for $\mathbb{R}^{n_{\text{cat}}}$, where each category corresponds 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 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 nonoperational definition. The goal of learning in this sense is to discover an operational description of $\hat{f}$; that is, a description that can be used by the program in order to evaluate states and select the best fit. Thus, the learning task has been reduced to the problem of discovering an operational description of the objective function. Its 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. The choice of 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 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:

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

A parameter set $\boldsymbol{\theta}$ is required to compute f. 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 refer to the determination of the parameter set $\boldsymbol{\theta}$. A change the parameter values modifies the prediction y and, in turn, the prediction 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. Luckily, when the program trains predictions improve.

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 which embed data features, or an artificial neural network. In practice, this can never be assured so its 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. 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. Its also commonly referred to as prediction error however the term loss gives a more general description.

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

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 its 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 that encourages the program to pick a simpler $\boldsymbol{\theta}$. The final step in learning, is to take this

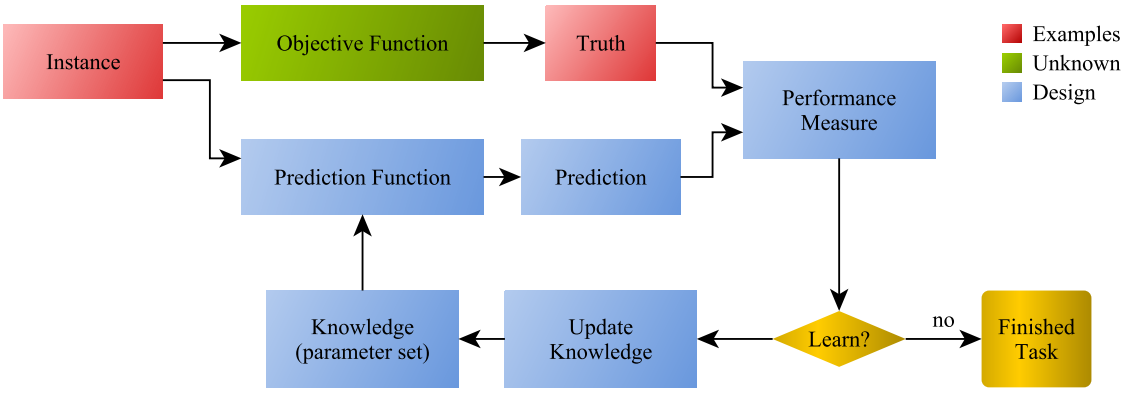


Figure 2.1: Supervised learning scheme for one instance. Red nodes constitute the examples datasets; green nodes, never known values; and blue, design choices. One instance processed by the objective and prediction functions. Although the former is unknown its output isn't (given by $\hat{y}$). The performance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

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 target and 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.

The following sections go over fundamental concepts of a Neural Network (NN) as the implementation of some ideas above discussed; go through two basic NN types; and present a tool used develop ML architectures.

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 only popularized in the last decade or two. With improvements in technique and technology ML 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 $x_{i0} = x_0$ to some real constant (usually 1), it becomes easy to write:

$$y = \phi\left(\sum_{j=0}^{n} x_j w_j\right) \tag{2.8}$$

where $x_j \in \mathbf{X}$ for $j = \overline{1,n}$, and the parameter set is simply $\boldsymbol{\theta} = \{w_j \in \mathbb{R} \mid j = \overline{0,n}\}$. The weights will determine the relative importance 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 $z_{\mathbf{W}} = \sum x_j w_j$. The main difference is an extra layer of complexity allowing a neuron to modify response behavior. In analogy with the biological neuron membrane potential threshold, an activation function $\phi(z_{\mathbf{W}})$ simulates how a neuron activates or fires. The function receives a weighted sum as input and outputs a single value representing that neuron's response. The output value is a prediction, i.e. the prediction function itself $f(\mathbf{X}; \boldsymbol{\theta}) = \phi(z_{\mathbf{W}})$. Figure 2.2 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 will be covered in Section 2.3.

When ANs activity response is all-or-none its called Linear Threshold Unit (LTU), which

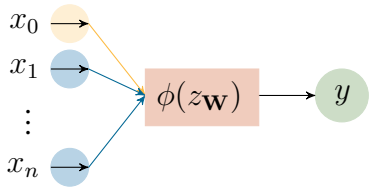


Figure 2.2: Artificial Neuron (AN) schematic. A vector with n entries (plus a bias) is input, then the weighted sum $z_{\mathbf{W}}$ is passed to the activation function ϕ, which returns a value y.

uses the Heaviside or step function for binary output. Of course, more complex functions can be used depending on the working model. The most commonly used are sigmoid-like functions: logistic function, $\tanh(x)$, $\arctan(x)$, $\operatorname{erf}(x)$, etc.; softmax, softplus, rectified linear unit, and more depending on the expressiveness desired for the target function. But all these change the behavior from a LTU to general AN.

The AN model is able to solve AND, NAND, OR and NOR logic gates classification tasks, but fails with XOR (exclusive OR)[2] 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 parallel stack 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 Deep Learning (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. 2.3). Its 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 a two neurons because each added neuron represents an additional straight line (see Fig. A.1).

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 (as any linear classifier) its 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

[2]Refer to Appendix A.

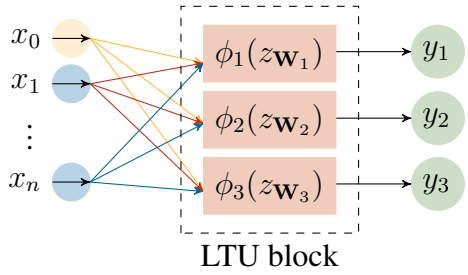


Figure 2.3: Simple ANN schematic with three stacked neurons. The same single instance $\mathbf{X}$ fed to every LTU (or AN). Each LTU process ϕ and outputs a value f independent from the other all others. The networks output is the ordered tuple $\mathbf{y}_i = (y_{i1}, y_{i2}, y_{i3})$, and the parameter set in this case is $\boldsymbol{\theta} = \bigcup_{i=1}^{3} \mathbf{W}_i$.

multilayered perceptron (MLP). The layer between input and output is called hidden layer and is deepened by each additional perceptron between input and output. and it is simply another LTU block fully connected to the next layer.

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 the loss.

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

where $\#(\mathbb{X})$ is the cardinality of the set. The linear regression algorithm, for example, use mean squared error (MSE) as cost, i.e. the loss function is $\mathcal{L} = (\mathbf{X} \cdot \mathbf{W} + w_0 - \hat{y})^2$. The parameter set represent a model in it self so, in order to learn, the cost must be optimized via differentiation. And because the cost is a score of error, must be minimized.

Why? The derivative gives the slope of a function at a certain point. It specifies how to scale

a small change to the input in order to obtain the corresponding change in the output. The derivative is therefore useful for minimizing a function because it tells how to change x in order to make a small improvement in y. For linear regression, the cost is easily differentiated because of its simple form. In general however, most cost functions cannot be optimized in closed form. This requires a change to an iterative 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 $\mathbf{u}$ (a unit vector) is $\partial_\alpha F(\boldsymbol{\theta} + \alpha\mathbf{u})|_{\alpha=0}$, which in turn evaluates to $\mathbf{u}^{\mathrm{T}} \cdot \nabla_{\boldsymbol{\theta}} F(\boldsymbol{\theta})$. To minimize the function F we would like to find the direction in which F decreases 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 $\min(\cos\vartheta)$, 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 method of steepest descent or Gradient Descent (GD).

The idea is to find the local gradient in parameter space, *move* in the opposite direction of the gradient to a new point assumed to be closer to a local minimum, and repeat the process until convergence. GD converges when every element in the gradient is zero. In the context of deep learning, functions that may have many local minima that are not optimal, and many saddle points surrounded by very flat regions makes optimization very difficult. We therefore usually settle 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. Its 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[3] of a NN because it controls how the algorithm will update parameter values throughout the learning process.

Consider ϵ to be large. It will help the algorithm to quickly move around the parameter

[3]The values of hyperparameters are (in principle) not adapted by the learning algorithm itself.

(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 making the search for a local minimum inefficient. 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.

Coming back to deep learning, the function to be optimized 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 Eq. (2.9), GD requires computing:

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

The computational expense is of order $\#(\mathbb{X}) = N$ 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, 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.

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

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

Here, 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 minibatch, 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 SDG methods. The main advantage of BGD is that the cost is calculated for all training instances, representing *full knowledge of all known possibilities*[4], but has serious memory issues for large datasets because the complete set must be loaded in cache. SGD take 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.

Up to now, we have the basis of a feed-forward NN, i.e. inputs provide initial information that flows forward through the network, then propagates up to the hidden units at each layer and then produces an output. This is called forward propagation and can continue onward until it produces a scalar $J(\boldsymbol{\theta})$. The information from the cost is then needed update parameters, so the information flows backwards through the network in order to compute the gradient.

This backward pass to compute the gradient 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 causes the most rapid decreases for the cost function. Allows to compute the gradient of any function, not specific to the cost function or multilayer NNs. A chain of responsibility from the output of a single neuron back to the input parameters is the main idea here. The prediction is the composition of functions $(\mathcal{L} \circ f\left(\mathbf{X}, \hat{y}\right))\left[\boldsymbol{\theta}\right]$ applied to the input. Yes, the chain rule of calculus is at hand.

Think of a DNN with L layers where the ℓ-th layer has m_ℓ neurons. In the following, consider that the 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

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

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 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:

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

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 like matrix product: $\boldsymbol{\Omega}_\ell \mathbf{A}_{\ell-1}$, where $\boldsymbol{\Omega}_\ell$ is the $m_\ell \times m_{\ell-1}$ connection matrix[5] 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:

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

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

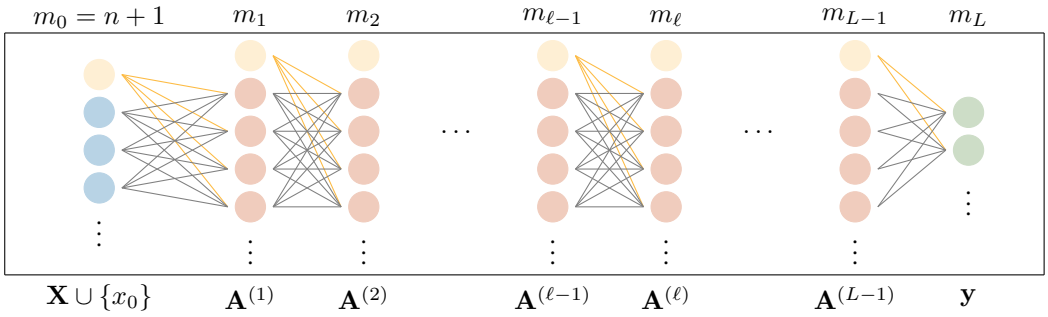


Figure 2.4: Sample DNN to explain back-propagation. Blue and green nodes are input and output (known) values; red nodes are hidden neuron layers; and yellow are the biases for every input. Note how each bias is independent of the previous layer and how the output layer doesn't have a bias value.

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

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

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

Lastly, the set of equations: Eq. (2.19) for the layer loss, Eq. (2.20) for the derivative wrt

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 wrt each minibatch. Its not going to be the complete gradient of the function but it gives a good approximation and a significantly computational speedup.

Together, back-propagation and GD complete the scope for parameter optimization in basic ML algorithms. There are downsides to how this methods can be implemented on modern technology but a lot of effort has been done to improve efficiency and develop new ways to optimize networks (cite) . Yet, this two sit at the base of most (and this work) NNs.

2.4 Convolutional Neural Networks

Amongst the plethora of NNs one can think of, deep neural networks and convolutional neural networks (CNNs) are the most common and illustrative models. Both can solve a lot of very complex problems (cite) 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. A CNN is a concatenation of a convolution process and (usually) a DNN.

The foundation to understand and work DNNs has already been laid so its CNNs turn. The idea 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 for a continuous variable (or sum for a discrete variable):

$$\begin{aligned} s(t) = (x * w)(t) &= \int x(t)w(t-a)dt \\ &= \sum_{a=-\infty}^{\infty} x(t)w(t-a) \end{aligned} \tag{2.22}$$

In CNN terminology, the first argument x is often referred as the input; the second, a kernel; and the output, 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. For example, for a two-dimensional array input I the kernel K will likely be also two-dimensional.

The learning algorithm will search for the appropriate values of the kernel in the appropriate place. Convolution is not naturally equivariant[6] to some other transformations, such as changes in the scale or rotation of an image. Other mechanisms are necessary for handling these kinds of transformations. Moreover, some kinds of data cannot be processed by neural networks defined by matrix multiplication with a fixed-shape matrix. Convolution enables processing of some of these kinds of data. Typical CNNs do make use of further specializations in order to deal with large inputs efficiently, but these are not strictly necessary from a theoretical perspective.

One key advantage CNN has over DNN is in pattern recognition, which is done prior to classification. By letting the program convolve the input with one or multiple kernels, it can recognize different patterns, which in turn are fed to different regions of the DNN. Another one is in parameter reduction via average or max pooling. This part of the layer is implemented after the convolution operation over a feature map. It sweeps each feature map with a simple identity kernel 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 convolution kernel. The main task for the programmer is to play around with the number of feature maps to generate per layer, number of pooling layers, and activation functions for all kernels in order to construct an *adequate* NN. All of the before mentioned are hyperparameters for a CNN not optimized by the algorithm.

Analyse the following exampl of an image has a single gorilla in it. The network should be able to classify it no matter where the gorilla is in the image: top, bottom, left, right, center, a corner, etc. If the input instance is somewhat different, like mirrored, or the color changes, the DNN neurons will fire differently and most likely fail to correctly classify the gorilla. However a CNN won't have this problem as it recognizes patterns. The example image has width, height and three color channels (depth). A first kernel K_1 convolve the image and retrieve a pattern into a feature map, say the edges; a second kernel can retrieve a different

[6]To say this is to say that as the input changes in some way, the output changes in the same manner.

pattern, like shadows; a third one retrieves something like face shapes; and so on and so forth. These feature maps can be pooled and convoluted again, to produce new feature maps that constitute a second convolutional layer. When all desired convolutions are done, all feature maps can be stretched and fed to a DNN in the some order. For example, the top region of the network can specialize in edges, the next region in shadows, the next in faces, etc. Its clear to see that the location of the gorilla inside the picture is not utterly relevant because the model is learning patterns and not direct values. One can think of this type of CNN as an enhancement or upgrade to a simple DNN. The convolution algorithm can be summarized in the following steps:

1. Read input image $\mathbf{X}$.
2. Initialize k kernels with same rank as input.
3. Convolve the kernel K_k with $\mathbf{X}$ and output a feature map F_k. Might reduce output size depending on how the kernel is swept (stride size).
4. Reduce further each feature map via pooling. Also may vary the stride size.
5. Pass the set of feature maps to an activation function.
6. Feed feature maps to a new set of kernels and repeat from (2). When done, stretch feature maps and feed to a DNN.

The number of parameters in NNs is crucial when it comes to training. 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 65,536 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! Therefore, parameter counting is very important for model fitting. Remember: the goal of supervised learning is to find the parameter set $\boldsymbol{\theta}$ that better represent the objective function $\hat{f}$. More expressiveness in representation translates to more parameters. If the model has too many parameters it will be able to achieve better accuracy but also may fall into overfit. Conversely, too few parameters produce underfit, 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 kernels and pooling layers whilst learning patterns. A single kernel will have something like 9 or 25 parameters which are used for a convolution. Only these 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! An image of size 256^2 would reduce to 8^2, and if fed to a 1,024 neuron dense layer that's only 65 thousand parameters more. CNNs reduce the chance to overfit.

2.5 TensorFlow

TensorFlow is an Application Programming Interface (API) developed and maintained by Google™. An API is intended to simplify the implementation of software. The TensorFlow API comes as software libraries designed as a platform for machine learning. In their own words:

> "TensorFlow is an interface for expressing machine learning algorithms and an implementation for executing such algorithms [...] The system is flexible and can be used to express a wide variety of algorithms, including training and inference algorithms for deep neural network models [...] The TensorFlow API and a reference implementation were released as an open-source package under the Apache 2.0 license in November, 2015 [...]"
>
> (Google Research (Abadi et al., 2015))

A TensorFlow computation is described by a *directed graph* which is composed of a set of *nodes* and represents the dataflow computation. This graphs are typically constructed by clients (users of the API) in one of the supported frontend languages, such as Python (Abadi et al., 2015). Take for example Fig. 2.5, where a code fragment to construct and then execute a TensorFlow graph using Python is shown, including the resulting computation graph.

```
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})
```

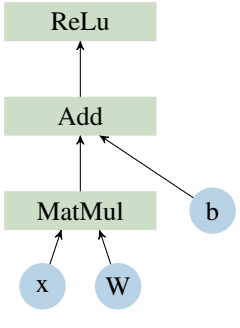


Figure 2.5: Example TensorFlow code fragment and its corresponding computation graph.

A lot of implementations of ML in Python are done using the Keras package because its a high level API based on TensorFlow that compiles the most common computations in simple commands familiar with pythonic code.

Chapter 3 – Simulation Data

In the past century, a new way to understand and predict natural phenomena via simulations came into play. This helps science design experiments and test for different scenarios whenever a physical test is impractical or not possible for current technology. It also means its possible to control and fine-tune every aspect of the physical situation. When it comes to cosmology, it is one way of testing for cosmological parameters obtained from probe measurements such as WMAP and Planck (for latest results see Hinshaw et al., 2013; White et al., 2018, and Table 1.1).

A lot of effort has been made in order to improve how the Universe is simulated and the resolution of this simulations. For example, the GADGET code is a gravitational force simulator using N-Body and Smoothed-Particle Hydrodynamics (SPH) techniques that was 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.

This chapter describes the simulation box from where signal KSZ maps used for this thesis were extracted. Then, the base ANN architecture to train for peculiar velocities is defined, followed by the description of some ideas to transform these maps into simpler formats that allow the use of different ANNs. Finally, data analysis and model validation methods used are properly presented and discussed.

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, allowing to reproduce the properties of large-scale, intra-galactic and intra-cluster medium, as well as the detailed properties of galaxies including morphological classification and internal properties. This also includes

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

Table 3.1: Cosmological and simulation parameters of *Box 0* from Magneticum Simulation.

the distribution of different metal species within galaxies and galaxy clusters.

This work makes use of the largest box *Box 0* with over 100 billion particles (see Table 3.1, and also Bocquet et al., 2016), making it the largest cosmological hydrodynamical simulation performed to date. 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, $\approx 1600\,\mathrm{deg}^2$, 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[1].

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

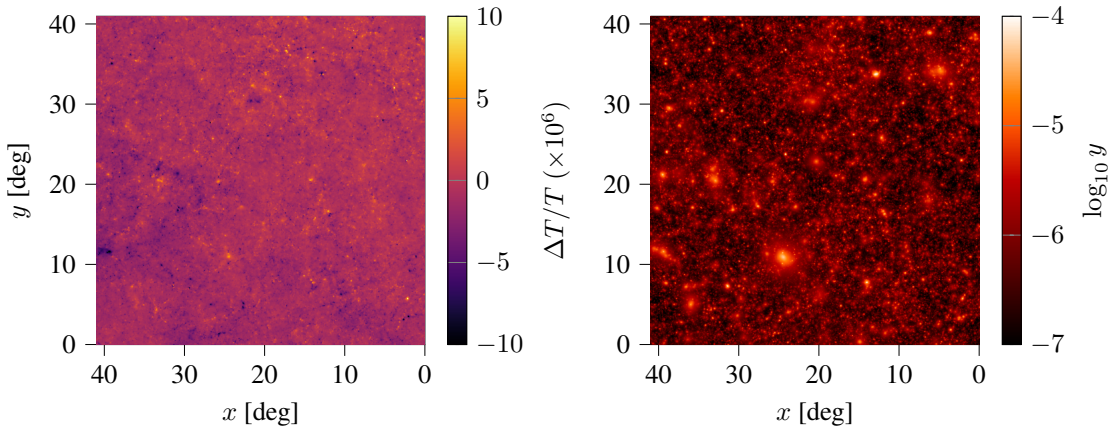


Figure 3.1: SZ maps from the Magneticum simulations. KSZ signal on the left. For the TSZ signal in the right panel the logarithm of the Compton-y parameter is shown to increase the dynamic colour range (see Fig. 1 in Soergel et al., 2018)[2].

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

[2]The simulated SZ maps were made available by Soergel et al. (2018) at http://magneticum.org/data.html#SZ

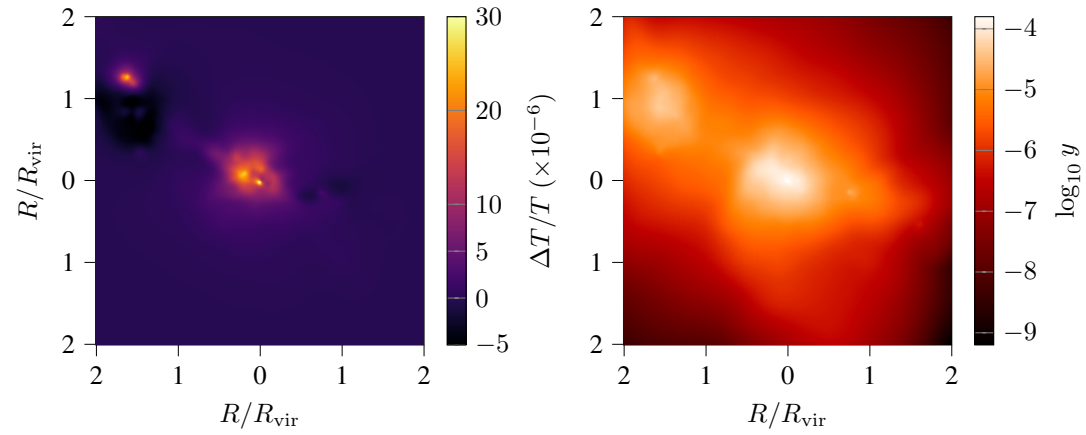


Figure 3.2: Sample KSZ (left) and TSZ (right) images.

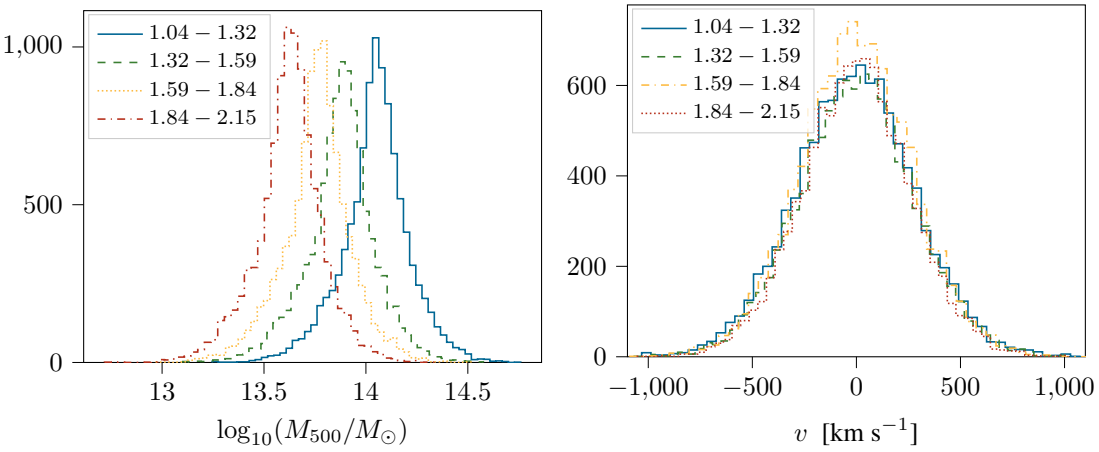


Figure 3.3: Mass selection and peculiar velocity frequency histograms per redshift slice.

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 its the case for large simulations as this one. The SMAC code (Dolag et al., 2005)[3] is a map making utility for idealized observations. With this tool is possible to extract KSZ, TSZ, and electron density maps for individual clusters (Fig. 3.2).

For the main analysis a total of 10,000 clusters from four redshift slices were selected, leaving 40,000 clusters in the sample. Mass selection for every redshift slice is shown in Fig. 3.3 where M_{500} is the mass corresponding to 500 times the critical density of the universe. Peculiar velocity histograms show that the chosen clusters exhibit typical cluster velocities.

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

3.2 Network Design

The base architecture of this work will be a CNN. Although previously discussed why this is a go to option, take the following argument from Hasanpour et al. (2016) where they analyze simple architectures that outperform deeper and more complex ones:

> Since CNNs take into account the locality of the input, they can find different levels of correlation through a hierarchy of consecutive application of convolution filters. This way they are able to find and exploit different levels of abstractions in the input data and using this perform very well on both coarse and fine level details. Therefore the depth of a CNN plays an important role in the discriminability power the network offers. The deeper the better.
>
> (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. This architecture has good performance and is simple enough to understand and implement. For this reason, an adapted version was coded using TensorFlow, both versions 1.14 and 2.0 for Python 3.8, to train the regression task for Magneticum images and peculiar velocities.

The CNN is composed by nine layers that work with monochromatic images. It employs a homogeneous design utilizing 3×3 kernels for convolutional layers and 2×2 kernels for pooling operations, except on the first one with 4×4. Convolution stride is set to 1×1 and kernel stride is always set to the kernel shape. Figure # shows a diagram for the architecture and Table # the specifics. Every convolution block ends with a pooling operation to reduce size before the activation pass.

To better understand how this NN is tested consider the instances datasets $\mathbb{X}_{0,z}$ with cardinality $\#(\mathbb{X}_{0,z}) = 10,000$ each. This means that each dataset contains all KSZ images from a single redshift slice z. In the following it will be assumed that i counts from 1 to 10,000 unless specified otherwise. These datasets can be constructed as follows:

$$\mathbb{X}_{0,z} = \{\mathbf{X}_i \in \mathbb{M}_{256\times256} \mid \mathbf{X}_i \text{ is KSZ signal belonging to } z\} \tag{3.1}$$

Having a total of four datasets that constitute the original data (without transformations) denoted by the subscripted zero. To each instance dataset corresponds one targets dataset

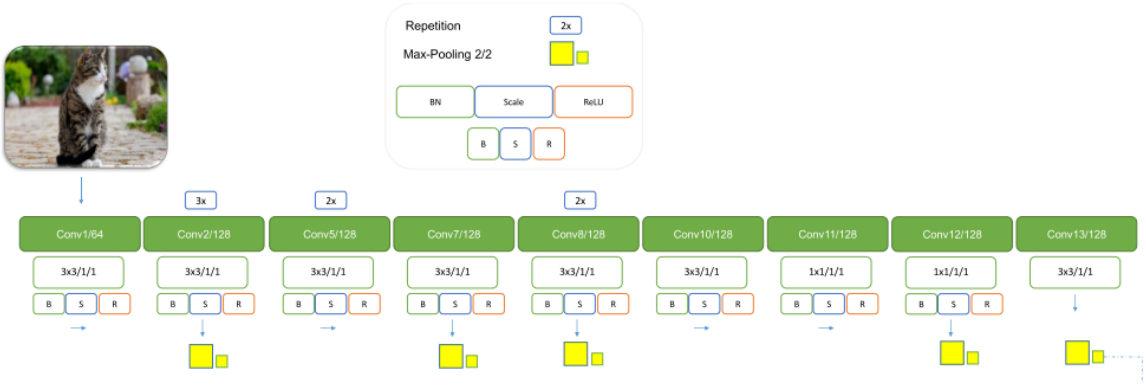


Figure 3.4: CNN architecture.

$\hat{\mathbb{Y}}_{0,z}$ and one predictions dataset $\hat{\mathbb{Y}}_{0,z}$ constructed as follows:

$$\hat{\mathbb{Y}}_{0,z} = \{\hat{v}_i \in \mathbb{R} \mid \hat{v}_i \text{ is the peculiar velocity for a cluster belonging to } z\} \tag{3.2}$$

$$\mathbb{Y}_{0,z} = \{v_i \in \mathbb{R} \mid v_i \text{ is the predicted peculiar velocity for a cluster belonging to } z\} \tag{3.3}$$

The next section describes how each of these instances sets is be processed by CNNs and DNNs in order to produce a prediction for any inputted SZ signal map. Both networks will have only one output neuron whose value $v \in \mathbb{Y}_{0,z}$ has to 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. Also, because transformations are done to all redshift slices, the subscript z is only used to denote individual slices, so that $\mathbb{X}_0$ refers to all pairs $(0, z)$ joined into one dataset containing the $40,000$ samples.

3.3.1 Redshift Dependence

The direct move is to use all instances on an as-is basis, i.e. feed the full dataset to a (*well* defined) CNN and expect it to correctly train (cite) (Yuyu). This is the reference model $\mathbb{X}_0$ to compare all other models.

The reference data is used in the following ways: KSZ per redshift slice (4 models); KSZ all snapshots (1 model more); KSZ+TSZ with all snapshots (1 model more); with a total of six

models to compare. The goal is to see if if the training is redshift independent, and if KSZ alone is enough for the NN to achieve convergence.

3.3.2 Systematic Distortions

Usually, a simulation map is an idealized observation with little to non systematic errors, and very high resolution. But how about real observations? To simulate real measurements the image can be distorted using different filters. For example, a uniformly random filter added to $\mathbf{X}$ simulates background noise introduced by measurement instruments. Software processing is common practice nowadays and some techniques produce anti-aliased images to reduce noise. Finally, simulations allow to have virtually unlimited high resolution images, however the technological capabilities of measurement instruments is a lot more limited, so low resolution results are more common. Still, these distortions are biased assumptions of probable error sources and encompass limited examples. With these ideas in mind, three new datasets are proposed:

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

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

$$\mathbb{X}_3 = \left\{ \mathrm{D}_d(\mathbf{X}_i) \mid \mathbf{X}_i \in \mathbb{X}_0,\ \mathrm{D}_d : \mathbb{M}_{256\times256} \mapsto \mathbb{M}_{\frac{256}{k}\times\frac{256}{k}},\ k = 2^d,\ d \leqslant 8 \in \mathbb{N} \right\} \tag{3.6}$$

These three instances datasets will be trained with the same CNN architecture as the first six specified in 3.3.1 with the exception that for $\mathbb{X}_3$ the input layer will take a different shape. The goal is to compare with the reference and see if systematic distortions or a loss in resolution alters the NN performance. Figure 3.5 shows how the sample instance from 3.2 is modified by these three transformations.

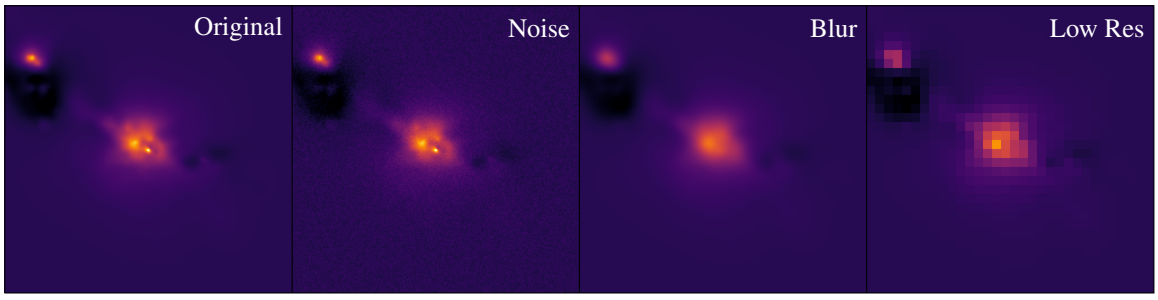


Figure 3.5: Distortions to a KSZ signal example. The original (far left) belongs to $\mathbb{X}_0$; white noise (mid right), to $\mathbb{X}_1$; Gaussian noise or blur (mid left), to $\mathbb{X}_2$; and low resolution (far right) achieved by downsampling $\mathbb{X}_3$ with D_3.

3.3.3 DNN vs CNN

It has been discussed how CNNs can be better than DNNs for image processing. But is it really true for our case of study? To answer this, three new model ideas are proposed. Note that this are arbitrary choices meant for exploration purposes and are briefly justified.

One immediate idea is to stretch each image into a vector. This can be done by stacking each dimension of the array next to each other in arbitrary, but consistent, order. For example take the second row of a matrix and concatenate to the right of the first row, then take the third and concatenate to the right of the previous concatenation, and so on. The resulting vector size is the direct multiplication of the extensions in each dimension. For the KSZ maps that would mean a vector with 65,536 entries! One realizes immediately that there are too many input values. More than 65 thousand parameters for the first layer only. An improved version is a stretched downsampled, e.g. $\mathbb{X}_3$ with D_4, to have at most 4,096 input values. Although this looses resolution and removes the direct comparison to the reference, it gives an idea of how a DNN can perform. Besides, the downsampling mapping D_d is actually doing a simple average pool convolution, this is:

$$\mathbb{X}_4 = \left\{ \mathrm{F}(\mathbf{X}_i) \mid \mathbf{X}_i \in \mathbb{X}_3 \text{ with } \mathrm{D}_2,\ \mathrm{F} : \mathbb{M}_{64\times 64} \mapsto \mathbb{R}^{4,096} \right\} \tag{3.7}$$

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

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

The application of $\mathrm{F}(\mathbf{H})$ will output a vector with 36 entries:

$$\mathrm{F}(\mathbf{H}) = [1\ 1\ 1\ 1\ 1\ 1\ 1\ 2\ 2\ 2\ 2\ 1\ 1\ 2\ 3\ 3\ 2\ 1\ 1\ 2\ 3\ 3\ 2\ 1\ 1\ 2\ 2\ 2\ 2\ 1\ 1\ 1\ 1\ 1\ 1\ 1] \tag{Ex. 2}$$

Another approach would be to suppose that each pixel of the 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 calculation of its mean and variance yields some information of its distribution. A mathematical justification concerning moment generating functions (MGF) that serve as arguments for the

next dataset is found in Appendix B.

The next dataset is obtained by calculating the first four moments of each image, plus appending the minimum and maximum value to determine the limits. Now, with only the first four moments one cannot completely reconstruct the corresponding probability distribution function, but serve as good approximations. Additionally, the idea is to exchange the 65 thousand input values from a single instance for a smaller vector which contains only the most relevant point values that represent the original image.

$$\mathbb{X}_5 = \{\mathrm{M}(\mathbf{X}_i) \mid \mathbf{X}_i \in \mathbb{X}_0,\ \mathrm{M} = (\min,\ \max,\ E,\ \mathrm{Var},\ \mathrm{Skew},\ \mathrm{Kurt})\} \tag{3.8}$$

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

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

A third option arise with the question: is there a mapping that can represent the amount of "information" that a single image contains? Same as before, the objective is to reduce the amount of input values by obtaining a data vector that represents all information contained within a single image. Its important to remark this is a representation of the real image constructed for exploration reasons only, and it's not necessarily the only nor the best one.

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).

$$\langle A_w \rangle = \frac{1}{4(2w-1)} \sum_{a_{ij} \in A_w} a_{ij} \tag{3.9}$$

The amount of information in each pixel is encoded by 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. This basic mapping can be modified to a more elaborate expression.

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 $m \times m$, then $w = \overline{1, \frac{m}{2}}$. 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. This vector will have $\frac{m}{2}$ possible slices, so for KSZ images that means 128 elements.

$$\mathbb{X}_6 = \left\{ \mathrm{S}(\mathbf{X}_i) \mid \mathbf{X}_i \in \mathbb{X}_0,\ \mathrm{S} : \mathbb{M}_{256\times 256} \mapsto \mathbb{R}^{128},\ \mathrm{S} = (\langle A_1 \rangle, \ldots, \langle A_{128} \rangle) \right\} \tag{3.10}$$

As w increments, so does the size of the slice and further away from the center of the image, i.e. its 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 $2R_{\text{vir}}$ per side, R_{vir} is at $w = \frac{m}{2}$. In consequence, the distance in $h^{-1}\mathrm{Mpc}$ each slice is from the center is simply $r = \frac{2w}{m} R_{\text{vir}}$, or equivalently:

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

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

- Overdensities 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.
- Overdensities farther than R_{vir} will produce a humps and valleys but are flattened indicating small contribution or relevance to the overall cluster signal.

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

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

$$S(A_1) = (3,\ 2,\ 1) \tag{Ex. 4}$$

Its clear that the strongest signal is located at the centre $\mathbf{H}$, and it fades linearly. This is the reason why it was constructed in this manner. Of course, this is a simplified example to illustrate how the mapping S works.

Figure 3.6 shows a complete mapping $\mathrm{S}(\mathbf{X}) \in \mathbb{X}_6$ of the sample KSZ signal in Fig. 3.2 as a function of distance to the centre r. It becomes clear the mapping works as intended. Its mainly positive at the centre and it faints as distance from the centre 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). Additionally, this mapping reduces the impact of signal outside the region of interest around the virial radius.

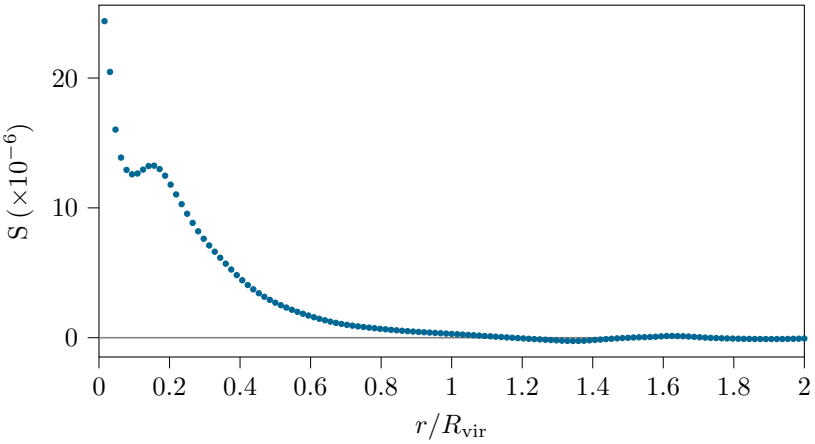


Figure 3.6: Radial mapping over a sample KSZ image in Fig. 3.2.

3.4 Cross-Validation

To test the reliability of the trained model a holdout cross-validation method is implemented. This is done by randomly assigning data points $\mathbf{X} \in \mathbb{X}$ to two non-intersecting sets $\mathbb{T}$ and $\mathbb{V}$, usually called the training set and the test set, respectively.

$$\mathbb{X} = \mathbb{T} \cup \mathbb{V} \qquad \mathbb{T} \cap \mathbb{V} \equiv \emptyset \tag{3.11}$$

$$\hat{\mathbb{Y}} = \hat{\mathbb{Y}}_{\mathbb{T}} \cup \hat{\mathbb{Y}}_{\mathbb{V}} \quad \text{where} \quad \hat{\mathbb{Y}}_{\mathbb{T}} \cap \hat{\mathbb{Y}}_{\mathbb{V}} \equiv \emptyset \tag{3.12}$$

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

The size of each of the sets is arbitrary although usually the train-to-test ratio is 80/20%. Typical cross-validation involve multiple runs averaged together (k-fold), but in holdout a single run is performed. This means less computation time but can result in unstable predictive accuracy since its not smoothed by the average of multiple iterations.

Each training step is delimited by the update of the parameter set. There can be as much

training steps as there are samples in $\mathbb{T}$ (SGD), however 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. Every time an instance is fed forward through the computational graph, a comparison between the prediction $v \in \mathbb{Y}_{\mathbb{T}}$ and the truth $\hat{v} \in \hat{\mathbb{Y}}_{\mathbb{T}}$ is done internally in order to estimate the loss (supervised learning). After the last epoch the algorithm performs one final optimization over the parameter set and ends the learning task. This final $\boldsymbol{\theta}$ represents the final knowledge of the task and is the model itself.

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 truth. In practice, all pairs $(\hat{v}, v)$, are expected to fall around the identity line in this plot due to an error in v. These errors are called residuals and give the distance from a prediction to its target.

$$\mathbb{E} = \{e = v - \hat{v} \mid \text{for all pairs } (v, \hat{v}) \in (\mathbb{Y}_{\mathbb{V}}, \hat{\mathbb{Y}}_{\mathbb{V}})\} \tag{3.14}$$

The expectation value $\mathbf{E}[e] = \hat{\mu}_e$, and for e^2 is the mean squared prediction error (MSPE). Both estimators give an idea of how good is a model, where small expectation values are better, indicate less dispersion and, ideally, tend to zero.

$$\text{MSPE} = \frac{1}{\#(\mathbb{E})} \sum_{e \in \mathbb{E}} e^2 \tag{3.15}$$

The algorithms accuracy is measured by the coefficient of determination R^2 (R-squared) that measures how well observations fit the model by comparing the variances $\mathbf{E}[(e - \hat{\mu}_e)^2] = \sigma_e^2$ and $\sigma_{\hat{v}}^2$. The final validation step in training will produce a predictions dataset with fixed R^2 representing the accuracy measure over $\mathbb{V}$.

$$R^2 = 1 - \frac{\sigma_e^2}{\sigma_{\hat{v}}^2} \tag{3.16}$$

Regardless of representation $\boldsymbol{\theta}$, $R^2 = 0$ means that the dependent variable (in this case v) cannot be predicted using the independent variable (in this case $\mathbf{X}$) as the residuals dispersion is as large $\sigma_{\hat{v}}$. Conversely, if $R^2 = 1$, the dependent variable is always predicted by the independent variable, i.e. the model *precisely* represents the *true* relation. It can also be interpreted as telling how much of the dependent variable can be explained by the independent variable (in percentage).

Deviations from the target value are 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^2 and MSPE are useful point values to quantify how good is a model, when it comes to predicting values they do not provide any information regarding biases.

One way to test for systematic errors is by analysing the residuals distribution $P(e)$. Is it random, uniform, gaussian or something else? In case, particularities of $P(e)$ are not really important for the discussion. What's matters is how a distribution compares to another model's residuals distribution. If the residuals are randomly distributed around zero with distances not much greater than the MSPE, it can be considered a good fit and without systematic errors. Because all residuals are known in $\mathbb{E}$, its possible to provide a confidence interval over the mean $\hat{\mu}_e$ using the bootstrap technique.

Consider the case where two residual distributions belonging to two different models to have the same MSPE and R^2: how can one tell them apart? The overlap index η is a similarity test for two distributions as it measures 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. Then the overlap index η:

$$\begin{aligned} \eta(A,B) &: \mathbb{R}^n \times \mathbb{R}^n \to [0,\ 1] \\ \eta(A,B) &= \int_{\mathbb{R}^n} \min\left[f_A(x), f_B(x)\right]\ dx \end{aligned} \tag{3.17}$$

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 $b_2 < a_1$. But of course, if the overlap between two distributions is zero, the premise that the MSPE is the same for both distributions isn't valid.

3.5 Traditional Method

The traditional method for estimating peculiar velocities requires to know the optical depth for each individual cluster. According to Eq. (1.25) optical depth is integrated over a path length ℓ at a specific location in space. As in (cite) , the optical depth for a single cluster is cal-

culated by averaging the electron density over within the virial radius for $|\ell| = 100h^{-1}\mathrm{Mpc}$. This is possible for simulation data because the electron density maps can be extracted directly.

$$\tau = \frac{1}{\pi R_{\mathrm{vir}}^2} \int_0^{R_{\mathrm{vir}}} \int_{-\ell}^{\ell} \sigma_{\mathrm{T}} n_e \, d\ell dr \tag{3.18}$$

Chapter 4 – Results

The following sections contain the results obtained for sections 3.3, 3.4 and 3.5.

4.1 Model Training

As discussed in the previous chapter, model training predictions are presented in a $v - \hat{v}$ plot in order to show the capacity of a particular machine learning 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 this reason a linear fit is performed. The slope m is reported without considering the y-intercept bias as its assumed the model does not contain one. Slope and coefficient of determination R^2 are reported for every prediction dataset $\mathbb{Y}_{\mathbb{V}}$. This dataset is comprised of randomly selected clusters and obtained through the holdout method for cross-validation (see 3.4). Additionally to reporting the parameter values, two shaded areas are coloured over each plot for visual aid. These are: the linear fit confidence interval for which 95% of the linear fits performed over the same data points will fall inside it; and the linear fit prediction band for which 95% of the predictions will fall. The former is related to the slope, whilst the latter is related to the best fit error.

When testing for redshift dependence, the holdout method gives a test sample of 2000 clusters per redshift. Figure 4.1 shows the prediction results for each individual redshift slice $\mathbb{V}_{0,z}$. As it can be seen, the slope is in good agreement with the ideal line. Also the coefficient of determination is similar in all four cases, proving consistency with Eq. (1.38) for KSZ effect which is redshift independent.

Joining all redshift slices into $\mathbb{V}_0$ the same model is trained but now covering a larger redshift region. Even though this case has larger errors, its also a more general and flexible model. Figure 4.2...

Section 3.3.2 presents a way of simulate systematic distortions that can be added due to observational errors or limitations. Those here considered are: white noise, to simulate background and instrumental noise; gaussian blur, to imitate antialiasing software that reduce noise; and lower resolution image, which better represents the technological capabilities of measurement instruments. Figure 4.3...

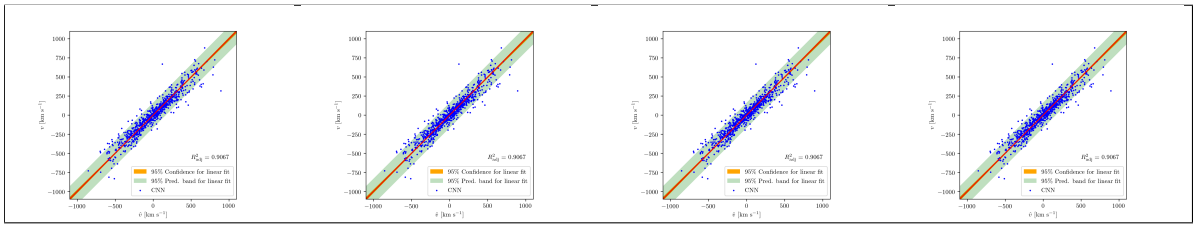


Figure 4.1: Test for redshift dependence on all four redshift slices $\mathbb{V}_{0,z}$. The horizontal axis is the target or true peculiar velocity from the cluster catalogue, whilst the vertical, is the predicted peculiar velocity. The solid red line is the reference ideal line; the yellow shaded area is the 95% confidence interval for the linear fit, where m is the *best* fit line slope; and the green shaded area is the 95% prediction band of the best fit slope.

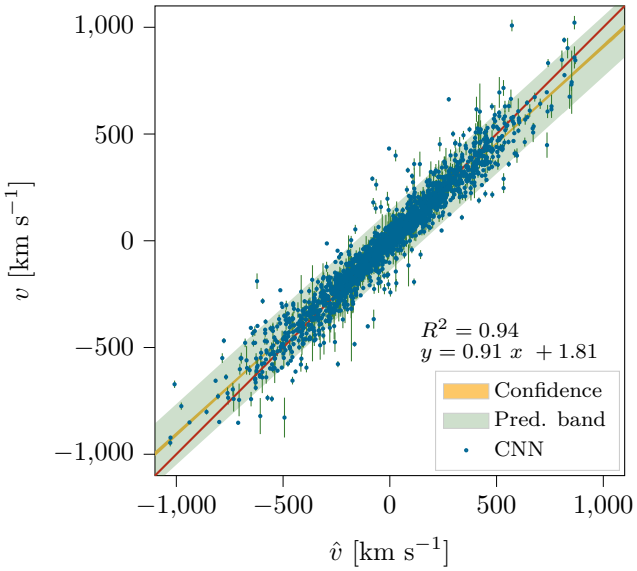


Figure 4.2: Training on full dataset.

4.2 Direct Calculation

Using Eq. (3.18) to estimate individual cluster optical depths, and Eq. (1.38) to calculate its peculiar velocity.

Figure 4.4... Show strong correlation with

4.3 Pairwise Velocity Estimator

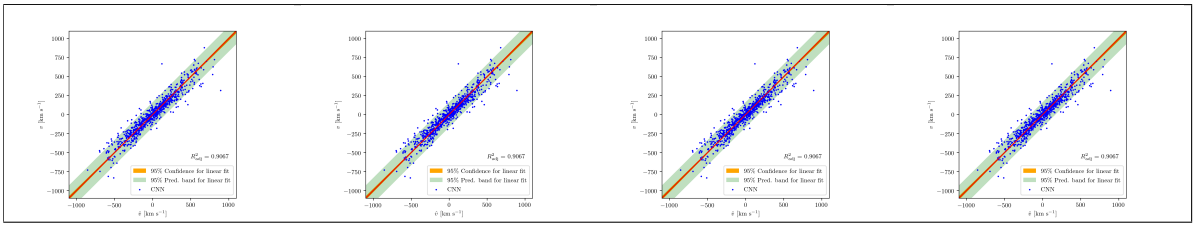


Figure 4.3: Test for systematic distortions.

Model	Slope	R^2	MSPE ($\times 10^n$)	C. I.
$\mathbb{V}_{0,1.04-1.32}$	0.99	0.001		
$\mathbb{V}_{0,1.32-1.59}$	0.99	0.001		
$\mathbb{V}_{0,1.59-1.84}$	0.99	0.001		
$\mathbb{V}_{0,1.84-2.15}$	0.99	0.001		
$\mathbb{V}_{0,1.04-2.15}$	0.99	0.001		
$\mathbb{V}_{0,\mathrm{KSZ+TSZ}}$	0.99	0.001		

Table 4.1: All models results

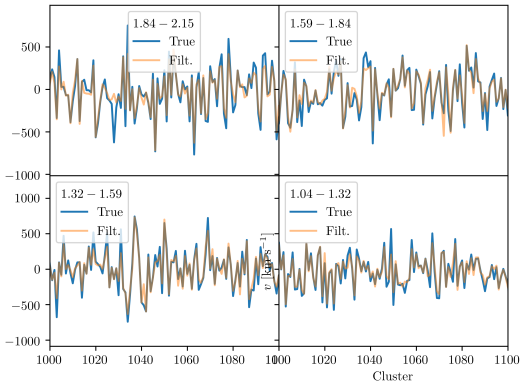


Figure 4.4: Peculiar velocities using the traditional method.

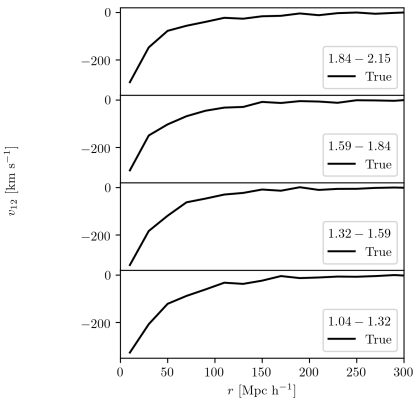


Figure 4.5: Pairwise velocity estimator per redshift slice.

This page intentionally left blank.

Chapter 5 – Conclusions

Appendix A – Artificial Neuron Logic Solver

Let's analyze how a single AN can classify a simple logic classification. In Fig. A.1 are three logic truth tables, each represented as a graph. In this simple case is very easy to identify every possible instance so the complete training dataset is:

$$\mathbb{X} = \{(0,0),(0,1),(1,0),(1,1)\} \qquad \hat{\mathbb{Y}} = \{(0),(1)\}$$

Take $\hat{f} : \mathbb{X} \mapsto \hat{\mathbb{Y}}$ to be an objective function that maps every instance using an arbitrary logic gate (AND, OR, XOR) to a target in the truth dataset; and $f : \mathbb{X} \mapsto \mathbb{Y}$ a guessed map of how the parameters in $\boldsymbol{\theta}$ can be combined in order to approximate $\hat{f}$. A single AN will take only one instance $\mathbf{X}^{(i)} = (x_1^{(i)}, x_2^{(i)})$, compute a weighted sum, and pass it to the step function to output a prediction. If this prediction is correct, then the next instance can be fed, but if its wrong it can decide to update its parameters $\boldsymbol{\theta}$ before feeding the next instance. In the end, a neuron's computation is simple a straight line $z_{\mathbf{W}} = w_0 + w_1 x_1 + w_2 x_2$ which splits space into two regions (see Fig. A.1).

If two neurons are stacked together and receive the same input, each will output a different (in principle) straight line, e.g.: $w_{1,0} + w_{1,1} x_1 + w_{1,2} x_2$ and $w_{2,0} + w_{2,1} x_1 + w_{2,2} x_2$. With two lines, three regions can be defined and so XOR logic properly solved.

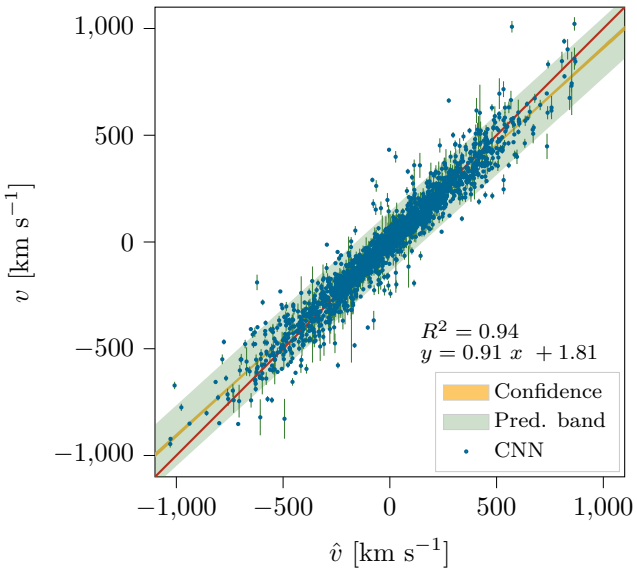


Figure A.1: Logic gates OR (left), AND (center), and XOR (right) where all possible outputs are represented in xy plots. Each colored region delimits a group with the same class.

Appendix B – 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.*

This page intentionally left blank.

Bibliography

Abadi M., et al., 2015, TensorFlow, Google, http://www.tensorflow.org

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

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

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

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

Diamond J., Bellwood P., 2003, Science, 300, 597

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

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

Gamow G., 1946, Phys.˜Rev., 70, 572

Hasanpour S. H., Rouhani M., Fayyaz M., Sabokrou M., 2016, Lets keep it simple, Using simple architectures to outperform deeper and more complex architectures (arXiv:1608.06037)

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

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

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, The Sunyaev-Zel'dovich Effect Spectrum of Abell 2163 (arXiv:astro-ph/0204134)

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

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

Makiya R., Hikage C., Komatsu E., 2019, arXiv e-prints,

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

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 2016, A&A, 594, A1

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, doi:10.1002/9783527618170

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

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

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

Springel V., 2005, The cosmological simulation code GADGET-2 (arXiv:0505010), doi:10.1111/j.1365-2966.2005.09655.x

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

Springel V., et al., 2005, Nature

Sunyaev R. A., Zeldovich Y. B., 1980, Monthly Notices of the Royal Astronomical Society, 190, 413

Switzer E. R., 2016, Lambda-CDM Model of Cosmology, https://lambda.gsfc.nasa.gov/education/graphic_history/univ_evol.cfm

Wang Y., Rooney C., Feldman H. A., Watkins R., 2018, Monthly Notices of the Royal Astronomical Society, 480, 5332

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

White D. M., Zacchei A., Zonca A., 2018, Technical report, Planck 2018 results. VI. Cosmological parameters. (arXiv:1807.06209v1), doi:10.1051/0004-6361/201525830

Wright E. L., 1979, The Astrophysical Journal, 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 Astrophysics and Space Physics, 4, 173

Glossary

activation function	Emulates a neuron's response.
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.
cost	Performance measure. Scores the error when using a model.
feature	Represent some quality or quantity of the instance. Characteristic or attribute of an instance.
forward propagation	The flow of information is directed from input to output. A single instance completes a forward pass through the networ, where cost and prediction values are calculated.
hidden layer	Extra layer hidden between input and output.
hyperparameter	Control the behavior of the learning algorithm and are arbitrarily set by the user.
hypothesis	Target value for the objective function.
instance	A particular case, example or single occurrence of something.
instances dataset	Contains all available examples.
layer	Stack of parallel neurons.
learning	Determination of the parameter set $\boldsymbol{\theta}$.
learning rate	Optimization algorithm step parameter.
loss function	Score function measuring the *goodness* of the current parameter set.
minibatch	Instances subsample dataset
objective function	Holds the true relationship bewteen instance and truth. It is unknown.
parameter set	Represents the knowledge of a task.
prediction	Model output value.
prediction function	Maps a training example to a value y.
predictions dataset	Contains predicted values corresponding to all available examples.
supervised learning	Machine learning task of learning a function that maps an input to an output based on example input-output pairs.
targets dataset	Contains taget values or hypothesis corresponding to all available examples.

test set	Contains a subset of the instances and targets datasets not used during training.
training set	Contains a subset of the instances and targets datasets used during training.